COMPLETE

ENGINEERING CHANGE NOTICE

Page 1 of 2

1. ECN No 620362

Proj. ECN ———

2. ECN Category (mark one)	3. Originator's Name, Organization, MSIN, and Telephone No.		4. Date
Supplemental [] Direct Revision [X] Change ECN [] Temporary [] Standby [] Supersedure [] Cancel/Void []	J. F. Berger / W.T.S.E. / R1-43 / 373-4584		2/28/95
	5. Project Title/No./Work Order No.	6. Bldg./Sys./Fac. No.	7. Approval Designator
	242A Sable & C Programs	242-A	N/A
	8. Document Numbers Changed by this ECN (includes sheet no. and rev.)	9. Related ECN No(s).	10. Related PO No.
	WHC-SD-534-CSWL-001 REV. 1	N/A	N/A

11a. Modification Work	11b. Work Package No.	11c. Modification Work Complete	11d. Restored to Original Condition (Temp. or Standby ECN only)
[] Yes (fill out Blk. 11b) [X] No (NA Blks. 11b, 11c, 11d)	N/A	N/A Cog. Engineer Signature & Date	N/A Cog. Engineer Signature & Date

12. Description of Change

This ECN revises document WHC-SD-534-CSWL-001 Rev. 1 in its entirety and changes the revision to 2. This ECN incorporates all outstanding ECN (610002, 610602, 612396, 614996). The text portion of this document has been revised to reflect the hardware and software upgrade of the MCS system from revision 3.3 to revision 6.3 of the MCS software. In addition all SABL programs were revised to remove the IF SELECT statements out of them. This is now obsolete in the new revision of the MCS software. All C programs were changed to meet VAX C standards from Whitesmith C standards. Additional EPNs were added to programs UKF34.C and UKF43.C. The program ACKFLG.C was incorporated into WATCH.C for better programming. The DOG programs were made more efficient. Changes were made in the String packet for the new system. The rebuild procedure was changed completely. The SKID key configuration was changed to allow labels for the new VAX workstation.

13a. Justification (mark one)							
Criteria Change	[X]	Design Improvement	[]	Environmental	[]	As-Found	[]
Facilitate Const.	[]	Const. Error/Omission	[]	Design Error/Omission	[]		

13b. Justification Details

The MCS system at the 242-A evaporator has been upgraded. This SD document needed revision to reflect the changes in the new system.

14. Distribution (include name, MSIN, and no. of copies)

J. F. Berger	R1-43	Central Files L8-04
B. S. Darling	R2-87	O.S.T.I. (2) L8-07
J. E. Geary	S5-14	
D. S. Haring	R1-43	
R. J. Nicklas	R1-43	
R. M. Persin	R1-43	

RELEASE STAMP

OFFICIAL RELEASE BY WHC DATE MAR 01 1995 13

Sta 4

A-7900-013-2 (06/94) GEF095

OSTI

MAR 27 1995

RECEIVED

A-7900-013-1

ENGINEERING CHANGE NOTICE

1. ECN (use no. from pg. 1): 620362

15. Design Verification Required	16. Cost Impact				17. Schedule Impact (days)	
	ENGINEERING		CONSTRUCTION			
[X] Yes	Additional	[] $ N/A	Additional	[] $ N/A	Improvement	[] N/A
[] No	Savings	[] $ N/A	Savings	[] $ N/A	Delay	[] N/A

18. Change Impact Review: Indicate the related documents (other than the engineering documents identified on Side 1) that will be affected by the change described in Block 12. Enter the affected document number in Block 19.

SDD/DD	[]	Seismic/Stress Analysis	[]	Tank Calibration Manual	[]
Functional Design Criteria	[]	Stress/Design Report	[]	Health Physics Procedure	[]
Operating Specification	[]	Interface Control Drawing	[]	Spares Multiple Unit Listing	[]
Criticality Specification	[]	Calibration Procedure	[]	Test Procedures/Specification	[]
Conceptual Design Report	[]	Installation Procedure	[]	Component Index	[]
Equipment Spec.	[]	Maintenance Procedure	[]	ASME Coded Item	[]
Const. Spec.	[]	Engineering Procedure	[]	Human Factor Consideration	[]
Procurement Spec.	[]	Operating Instruction	[]	Computer Software	[]
Vendor Information	[]	Operating Procedure	[]	Electric Circuit Schedule	[]
OM Manual	[]	Operational Safety Requirement	[]	ICRS Procedure	[]
FSAR/SAR	[]	IEFD Drawing	[]	Process Control Manual/Plan	[]
Safety Equipment List	[]	Cell Arrangement Drawing	[]	Process Flow Chart	[]
Radiation Work Permit	[]	Essential Material Specification	[]	Purchase Requisition	[]
Environmental Impact Statement	[]	Fac. Proc. Samp. Schedule	[]	Tickler File	[]
Environmental Report	[]	Inspection Plan	[]		[]
Environmental Permit	[]	Inventory Adjustment Request	[]	None	[X]

19. Other Affected Documents: (NOTE: Documents listed below will not be revised by this ECN.) Signatures below indicate that the signing organization has been notified of other affected documents listed below.

Document Number/Revision	Document Number/Revision	Document Number Revision
N/A		

20. Approvals

Signature	Date	Signature	Date
OPERATIONS AND ENGINEERING		ARCHITECT-ENGINEER	
Cog. Eng. J.F. Berger	2-28-95	PE	
Cog. Mgr. R.J. Nicklas	2/28/95	QA	
QA		Safety	
Safety		Design	
Environ.		Environ.	
Other		Other	
Informal Reviewer (RM Persin)	2/28/95		
		DEPARTMENT OF ENERGY	
		Signature or a Control Number that tracks the Approval Signature	
		ADDITIONAL	

RELEASE AUTHORIZATION

Document Number:	WHC-SD-534-CSWL-001, REV 0
Document Title:	SEQUENCE AND BATCH LANGUAGE PROGRAMS AND ALARM-RELATED "C" PROGRAMS FOR THE 242-A MCS
Release Date:	2/28/95

This document was reviewed following the procedures described in WHC-CM-3-4 and is:

APPROVED FOR PUBLIC RELEASE

WHC Information Release Administration Specialist:

Kara M. Broz — February 28, 1995

TRADEMARK DISCLAIMER. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof or its contractors or subcontractors.

This report has been reproduced from the best available copy. Available in paper copy and microfiche. Printed in the United States of America. Available to the U.S. Department of Energy and its contractors from:

U.S. Department of Energy
Office of Scientific and Technical Information (OSTI)
P.O. Box 62
Oak Ridge, TN 37831
Telephone: (615) 576-8401

Available to the public from:

U.S. Department of Commerce
National Technical Information Service (NTIS)
5285 Port Royal Road
Springfield, VA 22161
Telephone: (703) 487-4650

MASTER

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DT

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

JKE 3/1/95

SUPPORTING DOCUMENT

1. Total Pages 702 ~~700~~

2. Title	3. Number	4. Rev No.
SEQUENCE AND BATCH LANGUAGE PROGRAMS AND ALARM-RELATED "C" PROGRAMS FOR THE 242-A MCS	WHC-SD-534-CSWL-001	2

5. Key Words

Distributive Control System
Monitoring and Control System
242-A Evaporator
SABL

6. Author

Name: J.F. Berger

Signature

Organization/Charge Code 71730

7. Abstract

A Distributive Process Control system was purchased by Project B-534, "242-A Evaporator/Crystallizer Upgrades". This control system, called the Monitor and Control System (MCS), was installed in the 242-A Evaporator located in the 200 East Area. The purpose of the MCS is to monitor and control the Evaporator and monitor a number of alarms and other signals from various Tank Farm facilities. Applications software for the MCS was developed by the Waste Treatment Systems Engineering (WTSE) group of Westinghouse.

The standard displays and alarm scheme provide for control and monitoring, but do not directly indicate the signal location or depict the overall process. To do this, WTSE developed a second alarm scheme which uses special programs, annunciator keys, and process graphics. The special programs are written in two languages; Sequence and Batch Language (SABL), and "C" language. The WTSE-developed alarm scheme works as described below:

SABL relates signals and alarms to the annunciator keys, called SKID keys. When an alarm occurs, a SABL program causes a SKID key to flash, and if the alarm is of yellow or white priority then a "C" program turns on an audible horn (the D/3 system uses a different audible horn for the red priority alarms). The horn and flashing key draws the attention of the operator.

8. RELEASE STAMP

OFFICIAL RELEASE BY WHC
DATE MAR 01 1995
STA 4

JKB 3/1/95

RECORD OF REVISION	(1) Document Number WHC-SD-534-CSWD-006 L	Page 1

(2) Title
SEQUENCE AND BATCH LANGUAGE PROGRAMS AND ALARM-RELATED "C" PROGRAMS FOR THE 242-A MCS

CHANGE CONTROL RECORD

(3) Revision	(4) Description of Change - Replace, Add, and Delete Pages	Authorized for Release	
		(5) Cog. Engr.	(6) Cog. Mgr. Date
Rev. 0	(7) Initial release EDT 131507, 7-15-91.	M.C. TEATS	R.J. NICKLAS
Rev. 1	Replace Revision 0 with Revision 1 in entirety and changed document title per ECN 190999.	M.C. TEATS	R.J. NICKLAS
Rev. 2 RS	Replace revision 1 with revision 2 in its entirety. Incorporated ECNs 610002, 610602, 612396, 614996 per 620362	J.F. Berger 2-28-95	R.J. Nicklas 2/28/95

SEQUENCE AND BATCH LANGUAGE PROGRAMS

AND ALARM-RELATED 'C' PROGRAMS

FOR THE 242-A MCS

Prepared By:

Waste Treatment Systems Engineering

MASTER

TABLE OF CONTENTS

1.0 INTRODUCTION 5

2.0 PURPOSE 7

3.0 SCOPE 7

4.0 DEFINITIONS 8

5.0 DESCRIPTION 11
5.1 Alarm Schemes 11
5.1.1 TI Alarm Scheme 11
5.1.2 New Alarm Scheme 11
5.2 Sequence Units and IPN Order 12
5.3 RPFs 13
5.4 UPFs 13
5.5 Sequence Database 14
5.6 Pointers, Tables, And How The Programs Work 14
5.6.1 Pointers 14
5.6.2 Procedure Used By All Programs 16
5.6.3 PCM1 DIN Alarms 17
5.6.4 PCMO DIN Alarms 18
5.6.5 PCMO AI Red and Yellow Alarms 19
5.6.6 PCMO AI White Alarms 20
5.6.7 Device Alarms 22
5.7 White Alarm Summary Program 22
5.8 Unit Relative Feature of Programs 24
5.9 Program Statuses, Diagnostics, and Corrective Action 24
5.10 "C" Programs Related To Alarm Scheme 25
FIGURE 1. Program Relationships 27
5.11 Other Alarm-Related "C" Programs 28
5.12 SKID Key Configuration 28

6.0 REBUILDING DATABASES AND PROGRAMS 29

7.0 APPENDICES 30

APPENDIX A - IPN ORDER 31

APPENDIX B -SEQUENCE DATABASE (SDB) 55
PCMOASDB 55
PCM1ASDB 65

APPENDIX C - RECIPE PROTOTYPE FILES (RPF) 69
ALLO 69

242A 83
ALL1 91
FARM 102
EMTY 111

APPENDIX D - UNIT PROTOTYPE FILES (UPF) 122
TOTAL0 122
SHORT0 143
PRCESS 155
OVRVEW 169
TOTAL1 195
TNKFRM 225
BLANKK 248

APPENDIX E - SEQUENCE AND BATCH LANGUAGE (SABL) 258
224BATCH 258
244HVAC 269
WSTXFR 280
350BATCH 290
FEEDBATCH 299
CA1BATCH 313
PB1BATCH 325
PB2BATCH 337
EA1BATCH 350
EC1BATCH 362
RWBATCH 373
URWBATCH 387
VENTBATCH 399
PCBATCH 412
IXBATCH 426
SCBATCH 437
BASINBAT 448
UTILBATCH 458
AMUBATCH 470
LERFBATCH 483
CASSBATCH 494
TAG 498
TIMEINHA 500
PGM0A 502
WHITE1 507
ANBATCH 540
APBATCH 555
AWBATCH 574
XYZBATCH 589
242BATCH 605
PGM1A 615

APPENDIX F - "C" PROGRAMS FOR ALARMS 619
DOG0A 619

DOG0B 622
DOG1A 625
DOG1B 629
WATCH 633
INHA 641
UKF34 645
UKF43 653
UKF44 661

APPENDIX G - STRING PACKET LISTING 665

APPENDIX H - REBUILD PROCEDURE 685

APPENDIX I - SKID KEY CONFIGURATION 694

1.0 INTRODUCTION

A Texas Instruments (TI) D/3 Distributive Process Control system was purchased by Project B-534, "242-A Evaporator/Crystallizer Upgrades". This control system, called the Monitor and Control System (MCS), was installed in the 242-A Evaporator located in the 200 East Area. The purpose of the MCS is to monitor and control the Evaporator and monitor a number of alarms and other signals from various Tank Farm facilities. Applications software for the MCS was developed by the Waste Treatment Systems Engineering (WTSE) group of Westinghouse. TI's D/3 division has been purchased by GSE Process Solutions Inc. (GSE). In this document TI and GSE are synonymous.

The standard TI displays and alarm scheme provide for control and monitoring, but do not directly indicate the signal location or depict the overall process. To do this, WTSE developed a second alarm scheme which uses special programs, annunciator keys, and process graphics. The special programs are written in two languages; Sequence and Batch Language (SABL), which is a proprietary language of TI, and "C" language. The WTSE-developed alarm scheme works as described below:

- SABL relates signals and alarms to the annunciator keys, called SKID keys. When an alarm occurs, a SABL program causes a SKID key to flash, and if the alarm is of yellow or white priority then a "C" program turns on an audible horn (the D/3 system uses a different audible horn for the red priority alarms). The horn and flashing key draws the attention of the operator.

- The SKID keys relate the alarms to custom graphic displays, which depict the process much like the Process and Instrumentation Engineering Diagrams (P&IDs). When an operator presses a flashing SKID key, a graphic showing an alarm message is displayed.

- Each graphic contains a menu, which may be used to direct the operator to other more detailed displays.

- When an alarm is acknowledged from a graphic display, user-written "C" code manages the acknowledgement of all alarm messages in that graphic and sequence unit throughout the system (on other D/3 displays such as faceplates, P1, and P2 displays). Likewise, when an alarm is acknowledged from a faceplate display or other D/3 displays the corresponding alarm messages are acknowledged in the appropriate graphics.

The engineering tools used to develop the alarm scheme includes following TI D/3 utilities; the Management of Database Elements (MDL) utility, the Recipe Management Subsystem (RMS), the Unit Prototype File (UPF) utility, the Sequence Database (SDB) utility, the Sequence Instruction Compiler (SIC) utility, the String Editor (SED), and the Sequence Key Interactive Definer Utility (SKID). Additionally, the source code of user-written "C" programs

are compiled to create task images. Finally, a custom graphics development package installed on a personal computer is used to develop the graphics for the D/3 system. Development of the WTSE alarm scheme from scratch is summarized below. Once developed, modifications will involve some combination of the steps below depending the extent of the modification.

- Signals are defined in the continuous database using the MDL utility. Signals in the continuous database are grouped into process areas or systems by declaring the Internal Point Number (IPN) for each signal. Usually, the IPN of a signal should not be defaulted.

- Variables used in sequence programs are defined using the RMS and UPF utilities. The unit prototype files declare the associated recipe prototype files.

- The sequence databases are built using the SDB utility, which defines the sequence units, the first External Point Name (EPN) of each signal type (analog or digital), the total number of each signal type in a sequence unit, and the associated UPF.

- SABL programs are written and compiled.

- String packets are created using SED. The string packets contain the custom alarm messages for the graphics.

- The annunciator key layout is defined.

- Graphics are created on a personal computer, down-loaded to the D/3, and compiled.

- "C" programs are written and compiled.

- All databases are loaded and all programs are installed and run. At this point, the alarm scheme may be tested to verify functionality.

2.0 PURPOSE

The purpose of this document is to provide a hardcopy of the software for the alarm scheme developed by WTSE. Revision 1 of this document contains listings of software contained in revision 0 of the magnetic media (WHC-SD-534-CSWD-008).

3.0 SCOPE

This document provides the following information:

- A list of the IPN order of the signals required to construct the sequence databases, and lists the EPNs contained in each sequence unit.
- A list of the sequence databases.
- A list of the sequence program variables. These variables are defined in the unit prototype files (UPF) and recipe prototype files (RPF).
- Listings of SABL and "C" programs that manage alarms in the WTSE alarm scheme and control the MCS graphics.
- The custom alarm messages (string packets) created with SED for the graphics.
- The layout of the annunciator keys.
- Listings of "C" programs for inhibiting and de-inhibiting certain evaporator process signals.
- Discussion of programs and databases where appropriate.

The Tag List (WHC-SD-534-CSCM-001) defines any alarms associated with a signal.

It is assumed that the reader has some programming background and has a good understanding of the process of developing applications for the MCS. It is assumed that the reader is already familiar with the TI continuous control package for D/3 version 3.3, is familiar with the operating system of the host computer, and understands SABL coding. Additionally, the reader should be familiar with "C" language programming. Although not necessary for this document, the reader should be familiar with the development of graphics, the string packets, and annunciator key layout.

4.0 DEFINITIONS

Analog Input (AI):
A signal indicting analog values such as temperature, pressure, or liquid flows. Usually a 4 to 20 mA or 0 to 100 mV signal which is converted by the MCS to engineering units whose range represents 0 to 100% of the calibrated range of a transmitter. Sometimes used to control flow rates, pump speeds, liquid levels, or temperatures by analog loop which provides a 4 - 20 mA output signal. For thermocouples (T/Cs) or resistance temperature devices (RTDs), the input range is roughly -10 to 100 mV, where negative base volts corresponds to freezing temperatures.

Analog chain:
An arrangement of software process control blocks, or database blocks, to handle analog signal addressing, scaling, alarming, and process control.

Continuous control database:
An arrangement of digital and analog signals into AI inputs, digital inputs (DINs), master devices (MDVs), and device (DEVs) which are named by EPN and ordered by IPN. Includes alarm limits for the critical (red) and non-critical (yellow) alarms. Alarm limits for informational supervisory (white) alarms are imbedded in SABL programs.

Device (DEV):
For control of valves and motors by means of fixed logic, including interlock logic (see also master device). Uses some combination of analog inputs, digital inputs, and command (operator) keyboard inputs. Provides one or more digital output (DO) signals, and/or provides a software input to another logic device or analog chain. Devices usually have more than two statuses.

Digital Input (DIN):
A discrete (on or off) signal using 24 VDC or 110 VAC. Used for leak detectors, radiation alarms, valve positions, motor statuses, and so on.

Display Control Modules (DCM):
A computer which communicates with the process control modules and creates all displays appearing on the operator workstations.

External Point Name (EPN):
A name, up to nine characters long, assigned to each AI, DIN, MDV, and DEV defined in the continuous control database.

Faceplate:
A standard TI display that shows the primary indications of a single EPN. Up to 16 EPNs can be grouped together and displayed simultaneously on one operator workstation screen. A grouping of up to 16 EPNs in a single display is sometimes called a faceplate group, or just faceplate. The faceplate group display is also a standard TI display.

Graphic:
A display created by engineering using a graphics development package installed on a personal computer to show the relationship of signals to each other and their alarms. Usually, the graphic appears much like a process flow diagram.

Internal Point Number (IPN):
A number assigned to each signal type (AI, DIN, MDV, or DEV) which locates the EPN within the continuous database. Sequence programs scan signals of one type in order of the IPN.

Master Device (MDV):
A block of database containing the actual boolean logic programming for control of valves and motors. The boolean logic program statements usually are a series of gates (EQUATE, AND, OR, XOR, NOT, FLIP FLOP, TDE, TDD, or OS).

Operator Control Module (OCM):
An operator console, or workstation, used to monitor and control Evaporator and Tank Farm signals. An OCM displays information such as faceplates, alarm lists, graphics, and trends.

Process Control Module (PCM):
Redundant computers that monitors all signals and runs all SABL programs and certain user-written "C" programs (DOG0A, DOG0B, DOG1A, DOG1B). A PCM performs all process control functions.

Recipe Prototype File:
A file which lists flags, integers, floating points, pointers, timers, and string variables used in sequence programs. The file name is 4 characters long. Each SABL program uses variables found in one UPF or one RPF.

Sequence and Batch Language (SABL):
A TI proprietary language that can access all blocks of the continuous database with puts and fetches.

Sequence Database (SDB):
A utility that relates continuous control database signals to sequence variables and sequence units. The SDB utility defines the sequence units, the first EPN of each signal type, the total number of each signal type in a sequence unit, and the associated UPF.

Sequence Key Interactive Definer Utility (SKID):
Usually relates displays to one of 64 annunciator keys on an OCM. A SKID key can be defined to run a sequence program, run a "C" program, change a sequence variable, command a device, and bring up a display by pressing the key. The light of each key can be off, on, or flashing as commanded by a SABL program.

Sequence Unit:
All signals in one process area or system are ordered into a sequence unit to allow access by the sequence database. Example sequence units are PB1, PB2, CA1, EA1, and EC1.

Signal type:
AI, DIN, MDV, or DEV.

String Editor (SED):
A utility used to create alarm messages, or strings, in graphics that change depending on the value of an integer variable in the sequence database. The application of SED is the only way to create flashing alarm messages in a graphic.

Unit Prototype File (UPF):
A file which lists scratch pad variable flags, integers, floating points, timers, pointers, and string variables used by SABL programs. The file name is 6 characters long. Each UPF must have a RPF assigned to it. Each SABL program uses variables found in one UPF or one RPF.

5.0 DESCRIPTION

5.1 Alarm Schemes

TI provided a standard D/3 alarm scheme and WTSE developed a new alarm scheme. The new scheme adds functionality to the scheme provided by TI. Both schemes are used (the new scheme does not replace the TI scheme) in the complete management of alarms.

5.1.1 TI Alarm Scheme

The MCS has two alarm systems, P1 (critical, red) and P2 (non-critical, yellow), that are part of the TI package. Critical alarms are indicated by a flashing red P1 on all OCM screens and produce an audible alarm. The audible alarm is silenced by pressing the "Silence" key on the OCM keyboard. A red alarm summary list, up to 4 pages long, is displayed by pressing the P1 operator console key. Non-critical alarms are indicated by a flashing yellow P2 on all OCM screens. There is no audible alarm for the P2 alarms as defined in the TI software package. A yellow alarm summary list, up to 4 pages long, is displayed by pressing the P2 operator console key. In the P1 and P2 alarm summary lists, acknowledged alarms appear steady, unacknowledged alarms flash, and unacknowledged alarms which have cleared appear flashing green.

5.1.2 New Alarm Scheme

The TI alarm package was considered inadequate by itself, requiring the development of an additional alarm scheme. The new alarm scheme had to satisfy the following 4 requirements:

- Graphics must roughly identify alarm locations.
- Graphics must display alarms using customized flashing alarm messages that are acknowledgeable from the graphic.
- Include a third alarm priority level called informational, or advisory (white) alarms.
- Provide audible alarms for the yellow and white alarms.

The D/3 system critical alarm horn is activated for red alarms. The new alarm scheme commands a device to turn on a separate horn if a yellow or white alarm occurs, and to flash a SKID key for any alarm that occurs. Both horns are silenced by pressing the "Silence" key on the OCM keyboard. Graphics are displayed by pressing one of 64 annunciator keys. When any alarm occurs, the SKID key associated with the alarm flashes. Pressing this key displays the graphic. The alarm is shown by a flashing message in the appropriate alarm

color; red, yellow, or white. At the same time, an alarm message is printed at system alarm typer. The Tank Farm graphics show a list of the last 5 alarms at the bottom of each screen. All alarms on one graphic are acknowledged system wide by pressing the acknowledge key. Conversely, alarms acknowledged at the faceplate or alarm list level are also acknowledged on the graphic. White alarms also light a White Alarm annunciator key. Pressing this key displays up to three pages of white alarms.

5.2 Sequence Units and IPN Order

Forty SKID keys have been assigned to 40 graphics involving 33 sequence units using the new alarm scheme described in Section 5.1.2. For the Evaporator, EPNs in each graphic are assigned to one sequence unit, are controlled by one SABL program, and are associated with one SKID key. Tank farm graphics differ somewhat in that one SABL program usually controls 4, and up to 5, separate SKID keys.

The continuous database is structured by the order in which EPNs are defined. This order is established by the IPN of each EPN. This structure creates the foundation for the sequence database and sequence units. A table of EPNs, IPNs, and Sequence Units is found in the IPN order list (Appendix A). Appendix A is arranged by sequence unit and IPN. The "First Graph" column shows the graphic number where the EPNs are found. The column "ACAD File Name" indicates the personal computer (PC) file name for the graphic. The PC files are found in directory D:\A386\DRAW with ".DWG" extensions. The column "Number per Seq Unit" shows the total IPNs allowed in each unit for the AI, DIN, and DEV signal types. The total allowable IPNs are defined in the sequence database (see Appendix B, and Section 5.5). The "PCM", "MCS EPN", "SIG TYPE", and "IPN" columns are self explanatory. The remaining contents of Appendix A are discussed as needed in the remainder of this document. The relationship between the IPN order (and EPNs) and graphic alarms through the use of pointers is discussed further in Section 5.6.1.

Usually, the continuous database is setup with spare IPNs (or EPNs not defined) at the end of each sequence unit. Therefore, there are fewer EPNs in a sequence unit than the maximum allowed by the sequence database. Hopefully as signals are added to the continuous database over time, these "spare" IPNs are sufficient in quantity to add new signals without having to reshuffle the entire IPN order, which is a major undertaking. It would be nice to add plenty of spares to the continuous database to handle all future needs. However, there is a tradeoff between the number of spares available and the additional memory consumed by having more spares. Generally, 5 to 10% is a reasonable amount of spare IPNs to last a long time.

5.3 RPFs

Recipe Prototype Files (RPF) define all flags, integers, analog variables, second timers, minute timers, and string variables that can be modified from the OCM. All RPFs are found in Appendix C. SABL requires defining RPFs, but these variables are only used in the new alarm scheme as an extension to the UPF scratch pad variables table.

5.4 UPFs

Unit Prototype Files (UPF) define all scratchpad variables used by the sequence programs. All UPFs are found in Appendix D. Each UPF has one RPF associated with it. The UPF for PCM0 is slightly different from the UPF for PCM1. Because of this, the variables used by the programs running in the two PCMs are slightly different.

The following table lists the sequence unit names, numbers, program names, and their associated PCM, RPF, and UPF. Note the similarity between sequence unit names and sequence program names.

SEQUENCE UNIT NAME	SEQUENCE UNIT NUMBER	SEQUENCE PROGRAM NAME	PCM	RPF	UPF
WHOLE0	0	TAG.SEQ	0	ALL0	TOTAL0
WHOLE0	0	TIMEINHA.SEQ	0	ALL0	TOTAL0
A244	1	244BATCH.SEQ	0	FARM	SHORT0
HVC244	2	244HVAC.SEQ	0	FARM	SHORT0
WASTE	3	WSTXFR.SEQ	0	FARM	SHORT0
A350	4	350BATCH.SEQ	0	FARM	SHORT0
FEED	5	FEEDBATCH.SEQ	0	242A	PRCESS
CA1	6	CA1BATCH.SEQ	0	242A	PRCESS
PB1	7	PB1BATCH.SEQ	0	242A	PRCESS
PB2	8	PB2BATCH.SEQ	0	242A	PRCESS
EA1	9	EA1BATCH.SEQ	0	242A	PRCESS
EC1	10	EC1BATCH.SEQ	0	242A	PRCESS
RWATER	11	RWBATCH.SEQ	0	242A	PRCESS
URW	12	URWBATCH.SEQ	0	242A	PRCESS
VVENT	13	VENTBATCH.SEQ	0	242A	PRCESS
PC	14	PCBATCH.SEQ	0	242A	PRCESS
IX	15	IXBATCH.SEQ	0	242A	PRCESS
SC	16	SCBATCH.SEQ	0	242A	PRCESS
BASINS	17	BASINBAT.SEQ	0	242A	PRCESS
UTIL	18	UTILBATCH.SEQ	0	242A	PRCESS
ROOMS	19	----------	0	242A	PRCESS
AMU	20	AMUBATCH.SEQ	0	242A	PRCESS
LERF	21	LERFBATCH.SEQ	0	242A	PRCESS
EVAP	22	CASSBATCH.SEQ	0	242A	OVRVEW
VAPOR	23	----------	0	242A	OVRVEW

SEQUENCE UNIT NAME	SEQUENCE UNIT NUMBER	SEQUENCE PROGRAM NAME	PCM	RPF	UPF
COND	24	----------	0	242A	OVRVEW
BLANK0	25	PGM0A.SEQ	0	EMTY	BLANKK
WHOLE1	0	WHITE1.SEQ	1	ALL1	TOTAL1
ANFARM	1	ANBATCH.SEQ	1	FARM	TNKFRM
APFARM	2	APBATCH.SEQ	1	FARM	TNKFRM
AWFARM	3	AWBATCH.SEQ	1	FARM	TNKFRM
XYZFRM	4	XYZBATCH.SEQ	1	FARM	TNKFRM
BLNDG	5	242BATCH.SEQ	1	FARM	TNKFRM
BLANK1	6	PGM1A.SEQ	1	EMTY	BLANKK

5.5 Sequence Database

The SDB utility associates a number of AI, DIN, and DEVs with a unit prototype file. SDB output files for PCM0 and PCM1 are found in Appendix B. These listings give the sequence unit number, sequence unit name, UPF name, the total quantity of DEVs and EPN of the first DEV, total quantity of AIs and EPN of the first AI, and total quantity of DINs and EPN of the first DIN.

5.6 Pointers, Tables, And How The Programs Work

5.6.1 Pointers

Analog and digital signal values are all accessed through pointers. The first DEV, DIN, and AI in a sequence unit are assigned to pointers. The first integer and flag, also called variables, of a table are also assigned to pointers. By arranging variables into tables and comparing the value of integers and flags in the tables to the value of the EPNs, alarms are found, acknowledged, or cleared. Pointers start with the first EPN in a sequence unit and are stepped by one through all signals, which are arranged by IPN, in the unit. Therefore, the IPN order and use of pointers determines the relationship between alarm messages in graphics and EPNs associated with the graphics and SKID keys.

Alarm programs work by assigning the first DEV, DIN, or AI in a sequence unit to a pointer, checking its alarm status, and then moving on to the next EPN by stepping the pointer. Each EPN has one integer (TYPEn where n is a number, or pointer PTYPE) assigned to it that defines the type (red, yellow, white, etc.) of alarms it has. Each EPN also has another integer that indicates the status of that alarm (INTn, or pointer PSTAT). It is the INTn variables that are declared in a graphic file in association with an alarm message string (or string packet conversion number) for a DIN. The string packets are defined using the SED utility, which generates the D/3 file STRING.BIN and list file STRING.OUT. The string packet listing is found is Appendix G.

The association of the INTn batch variables to the EPNs within each unit are found in the IPN order list ("SDB Int" column of Appendix A). The association of the TYPEn batch variables to the EPNs within each unit follow the same order. For example, by referring to Appendix A one can see that the DIN LDS-PP, in unit A244 (unit 1 of PCM0), is associated with INT17. Likewise, LDS-PP is also associated with TYPE17. The actual TYPEn designations (0 for none, 1 for white, etc.) are inferred from the Tag List (WHC-SD-534-CSCM-001), which defines all alarms associated with a signal.

Programs running in PCM0 and PCM1 work the same way with a few subtle differences, particularly the use of pointer PSKID. In addition, the names of variables used in PCM0 differ somewhat from the names of variables in PCM1 programs because different UPFs are involved. These differences are presented below and in Sections 5.6.3, 5.6.4, and 5.6.5.

Despite these differences, the meaning of the values of the program variables are identical. The values of some common pointers are described below. Differences in the variable names whose attributes are assigned (ie: ASSIGN PTYPE = TYPE0) to the pointers are also noted.

Values of pointer PTYPE (TYPEn for PCM0, IALMn for PCM1) are:

0 = no alarm is defined
1 = white alarm
2 = yellow alarm
3 = yellow and white alarm combination
4 = red alarm
6 = red and yellow alarm combination

Values of pointer PSTAT (INTn for all programs) are:

0 = no alarm
1 = alarm found but not acknowledged, results in flashing message in graphic
2 = alarm is acknowledged
3 = alarm is inhibited (INHA)
4 = signal is off scan or disabled (OSCN, or DIS)
5 = signal has a bad input (IBAD)

Values of pointer PREV (FRVn for PCM0, FXn for PCM1) when associated with DINs are:

0 = alarm when DIN_VAL is 0
1 = alarm when DIN_VAL is 1

Note: For digital inputs, alarms occur if PREV = PEPN:DIN_VAL

Values for pointer PREV (FRVn for PCM0, FXn for PCM1) when associated with AIs are:

0 = Hi or HiHi alarm
1 = Lo or LoLo alarm

The attribute of variable FPASn (for all programs) is assigned to pointer PPASS. If PPASS equals 1, the SKID light turns off (provided there are no other alarms on the graphic) and graphic alarm message disappears after the alarm is acknowledged.

5.6.2 Procedure Used By All Programs

The same procedure is used by all programs to handle alarms, although subtle variations may exist between programs. The commented code listings of each sequence program are found in Appendix E. The procedure used by the programs for alarm management is summarized below.

1. Does the EPN have an alarm defined? If so, PTYPE is greater than zero. The type (TYPEn, or IALMn) is a constant defined in the initialization section of a program.

2. If an alarm is defined, is the EPN inhibited? If so, PSTAT is less than 3.

3. If not inhibited, is there an alarm present? Check by comparing PDIN:DIN_VAL and PREV.

4. If an alarm is present, does the system know of the alarm and has it been acknowledged (PSTAT equals 2 if true)? If not, then go to the Alarm Subroutine, ALM.

5. What type of alarm is it (red, yellow, or white)? Check the PTYPE.

6. For PCM1 programs, what graphic is it found in? Check the PSKID.

7. If acknowledged, should the alarm message vanish and the SKID lamp turn off? If so, PPASS equals 1. PPASS is a constant defined in the initialization section of a program. Note that if other alarms exist in the graphic, the message for an alarm may vanish but the lamp may remain on.

8. Has the alarm been acknowledged either from graphics or the system? Check by bitwise AND (AND (PDIN:DIN_ALMCT,32)).

9. Summarize all alarms in this unit and flash the skid key accordingly. For example, CURENT = OR (CURENT,NXT).

The primary variations between programs include:

- A few programs handle alarms found in a PCM other than the PCM the program is running in. In these cases, the program must fetch the data from the other PCM. This programming is hard-coded. PCM fetches are performed in RWBATCH, WSTXFR, WHITE1, XYZBATCH, 242BATCH, and CASSBATCH.

- A few programs manage alarms associated with logic devices (see Section 5.6.7).

- Reset CAL block output for totalizers. Programs that do this are RWBATCH, ANBATCH, APBATCH, and AWBATCH.

- A permissive always involves an Input-Output Switch (IOS) block. For loops having a permissive, set the PID output % (the O/C display bar % value) to zero when a permissive is lost (value of the TOGGLE goes to zero) to match the loop output. Programs that do this are FEEDBATCH, PB2BATCH, EA1BATCH, and VENTBATCH.

- Synchronizing of cascades to provide a bumpless transfer of a cascaded loop setpoint. Programs where this is done includes PCBATCH, FEEDBATCH, and PB2BATCH.

- EPN statuses, of DEVs and DINs, for the interlock graphics.

5.6.3 PCM1 DIN Alarms

The pointer PSKID in PCM1 programs, which are mainly for Tank Farms, is used to indicate which graphic has an active alarm, and pointer PSN (variable FSNn) is a flag for the graphic to acknowledge. The reason these pointers are needed is because one Tank Farm sequence program in PCM1 may control 4 or 5 SKID lamps. In PCM0 programs, one sequence program controls one lamp. The following is a sample of the pointer table used in the ANBATCH program for 241-AN Tank Farm signals. Pointer names are in parenthesis. Similar pointer tables are used for the other Tank Farm programs; APBATCH, AWBATCH, and XYZBATCH. There is no column for pointer PSN in the table below because 1) the FSNn variables are assigned to a graphic and not to individual EPNs, and 2) only 4 or 5 FSNn variables (FSN0 to FSN3 or FSN4) are used in each program, one for each graphic.

EPN (PDIN)	TYPE (PTYPE)	STATUS (PSTAT)	DIRECTION (PREV)	VANISH (PPASS)	GRAPHIC (PSKID)
YS-AN-101	IALM0	INT0	FX0	FPAS0	SKID0
RSH-AN101	IALM1	INT1	FX1	FPAS1	SKID1
RSH-AN01C	IALM2	INT2	FX2	FPAS2	SKID2
YS-AN-102	IALM3	INT3	FX3	FPAS3	SKID3
RSH-AN102	IALM4	INT4	FX4	FPAS4	SKID4

EPN (PDIN)	TYPE (PTYPE)	STATUS (PSTAT)	DIRECTION (PREV)	VANISH (PPASS)	GRAPHIC (PSKID)
RSH-AN02C	IALM5	INT5	FX5	FPAS5	SKID5
YS-AN-103	IALM6	INT6	FX6	FPAS6	SKID6
RSH-AN103	IALM7	INT7	FX7	FPAS7	SKID7
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

The first EPN and the attributes of the first integer and flag found on the first row of the above table are assigned to their corresponding pointer. For example, these assignments are made in the code as follows:

```
ASSIGN PDIN  = 'YS-AN-101'
ASSIGN PTYPE = IALM0
ASSIGN PSTAT = INT0
ASSIGN PREV  = FX0
ASSIGN PPASS = FPAS0
ASSIGN PSKID = SKID0
```

Alarms are found by comparing the 'EPN':DIN VAL values to values in the PREV column of the table. The program looks at all DINs, AIs, and sometimes the DEVs in a unit. When a white or yellow alarm occurs, an audible alarm horn (different from the critical alarm horn) is turned on. This is done by setting the flag FALMON in any sequence unit. This flag is scanned by a "C" program running in the PCM. The subsequent actions of this and other "C" programs are discussed in Section 5.10.

Within the sequence program, the SKID key status is the result of performing a logical "OR" on the PSTAT column and converting it to the appropriate SKID key function number where:

0 = SKID lamp off
1 = SKID lamp steady
2 = SKID lamp flash
3 = SKID lamp off-synchronized flash

5.6.4 PCM0 DIN Alarms

PCM0 digital inputs, which are primarily Evaporator signals, are handled the same way as the PCM1 DINs, which are predominately Tank Farm signals, but are associated with a different UPF. Because of this, some variable names change. The following pointer table is a sample of PB1 digital signals showing the different variable names. Pointer names are in parenthesis. Similar pointer tables exist for the other PCM0 sequence units and DINs.

EPN (PDIN)	TYPE (PTYPE)	STATUS (PSTAT)	DIRECTION (PREV)	VANISH (PPASS)	IPN
YS-PC-105	TYPE0	INT0	FRV0	FPAS0	70
YS-PC105A	TYPE1	INT1	FRV1	FPAS1	71
ZS-CA1-7N	TYPE2	INT2	FRV2	FPAS2	72
YS-PB1-1	TYPE3	INT3	FRV3	FPAS3	73
ZS-CA1-7C	TYPE4	INT4	FRV4	FPAS4	74
LDS-SMPL2	TYPE5	INT5	FRV5	FPAS5	75
ZS-CA1-8N	TYPE6	INT6	FRV6	FPAS6	76
ZS-CA1-8C	TYPE7	INT7	FRV7	FPAS7	77
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

5.6.5 PCM0 AI Red and Yellow Alarms

As with PCM0 DINs, the PCM0 analog inputs are primarily Evaporator signals.

The type of analog alarm is always checked for each EPN prior to executing the analog alarm code. If TYPEn for an EPN is initialized as zero, then there are no defined alarms for the EPN and the bulk of the analog alarm coding is bypassed.

A new analog alarm exists if the analog alarm count (AI_ALMCT) number is less than 16 but greater than zero. Evaporator graphics use PSKID to indicate the previous analog alarm status. A new alarm occurs when PSKID does not equal PAI:AI_ALMCT, and the signal in not inhibited, and the alarm is current. In addition, if the PAI:AI_ALMCT is less than 3, then the alarm is non-critical and the yellow-white audible horn is activated. Hi and Lo alarms are generally yellow (non-critical). HiHi and LoLo alarms are always red (critical), meaning an audible alarm will be set off on an OCM by the D/3 software provided by TI. AI_ALMCT values are:

0 = no alarm
1 = Hi alarm
2 = Lo alarm
3 = HiHi alarm
4 = LoLo alarm
16 = all alarms are inhibited
32 = all alarms are acknowledged

A sample of the pointer table for analog signals in sequence unit PB1 is summarized below. Pointer names are in parenthesis. Usually PSTAT, PREV, and PPASS are not used in the code for analog red and yellow alarms. In almost all PCM0 programs PAI, PTYPE, and PSKID are used. Similar pointer tables exist for the other PCM0 sequence units and AIs.

EPN (PAI)	TYPE (PTYPE)	STATUS (PSTAT)	DIRECTION (PREV)	VANISH (PPASS)	AI ALMCT (PSKID)
KY-PB1-1	TYPE10	INT10	FRV10	FPAS10	SKID10
FI-CA1-7	TYPE11	INT11	FRV11	FPAS11	SKID11
II-PB1-1	TYPE12	INT12	FRV12	FPAS12	SKID12
VI-PB1-2A	TYPE13	INT13	FRV13	FPAS13	SKID13
VI-PB1-1A	TYPE14	INT14	FRV14	FPAS14	SKID14
VI-PB1-3A	TYPE15	INT15	FRV15	FPAS15	SKID15
FI-CA1-1	TYPE16	INT16	FRV16	FPAS16	SKID16
PI-CA1-9	TYPE17	INT17	FRV17	FPAS17	SKID17
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

There are a few minor differences in the pointer tables and code used in some of the PCM0 sequence programs. First, in PCBATCH and IXBATCH the pointer PTYPE is not used. Instead, PALARM (or FALMn) is used for the same purpose as PTYPE, but in different direction. In most programs, if PTYPE is set then the section of code is executed. However in PCBATCH and IXBATCH, if PALARM is not set, which is the case for all EPNs in these units, then the section of code is executed. Although a column for PALARM in the above table is not shown, the variable assignments in PCBATCH are similar (ie: for II-A-C100, the attribute of FALM5 is assigned to PALARM, and so on).

Secondly, sequence programs in PCM0 do not all require similar sections of code to handle the different types of alarms; red, yellow, and white. Some units do not have white alarms, so there may not be a section of code in the program for that unit to handle white alarms. Similarly, some units do not have red or yellow alarms, so there may not be a section of code in the program to handle them.

Finally, there are some code inconsistencies between programs. These are relatively minor and it is not the intent of this document to discuss them, but the reader needs to be aware that they exist. Just as there usually are several ways to arrive at the correct solution to a mathematical problem, there are even more possible combinations of code that will result in the same desired outcome. For example, the initialization of variables for analog signals may be done inside the same loop as the digitals, or the analogs may be initialized in a separate loop.

5.6.6 PCM0 AI White Alarms

All white alarms for analog signals are in PCM0. Most of these are Evaporator signals, but several are Tank Farm signals. PCM1 contains some white alarms, but these are only digital inputs. The white alarm summary page program runs in PCM1 and is discussed in Section 5.7.

Coding for white analog alarms use variables not used in the coding for analog red and yellow alarms. The type (TYPEn) of alarm is a constant defined during program initialization. TYPEn equals one (1) for white alarms. As with other programs, the alarm status (INTn) integer variables are related to an EPN and graphic alarm message. An EPN is assigned to pointer PAI before running analog white alarm subroutine, ANLG. The ANLG subroutine consists of two primary loops; one for handling the Hi and HiHi alarms, and second nearly identical loop for the Lo and LoLo alarms. The proper loop for each EPN is defined by the reverse flag variables (FRVn), which are constants established in the initialization section of a program. Values of the FRVn variables follow the convention:

0 = HI or HIHI alarm
1 = LO or LOLO alarm

Some EPNs have more than one white alarm. In such cases, EPNs will be associated with multiple FRV variables. The white alarm setpoints are controlled by the Tag List (WHC-SD-534-SWD-001). The total number of white alarms for each EPN can be determined from the Tag List, from sequence code listings, or from the IPN order (Appendix A). For example, Appendix A shows two white alarms for PI-EA1-13 in unit EA1. If an alarm must disappear after it is acknowledged, then the constant FPASn is set equal to one (1) during initialization. To determine if an alarm exists, analog measured value AI_MEAS is compared to floating point value AIn. For example, the measured value of PI-EA1-13 is compared to AI0, and so on. The white alarm setpoints, AIn, are defined in the initialization section of a program. The pointer table for EA1 unit white alarms is listed below. Similar pointer tables are used for other units.

EPN (PAI)	TYPE (PTYPE)	STATUS (PSTAT)	DIRECTION (PREV)	VANISH (PPASS)	ALARM (PALARM)
PI-EA1-13	TYPE26	INT26	FRV26	FPAS26	AI0
PI-EA1-13	TYPE27	INT27	FRV27	FPAS27	AI1
FIC-EA1-1	TYPE28	INT28	FRV28	FPAS28	AI2
PI-EA1-1	TYPE29	INT29	FRV29	FPAS29	AI3
PI-EA1-1	TYPE30	INT30	FRV30	FPAS30	AI4
TI-EA1-5	TYPE31	INT31	FRV31	FPAS31	AI5

Values of these variables as found in EA1BATCH.SEQ are:

EPN (PAI)	TYPE (PTYPE)	STATUS (PSTAT)	DIRECTION (PREV)	VANISH (PPASS)	ALARM (PALARM)
PI-EA1-13	1	INT26	1 (LoLo)	0	75.0
PI-EA1-13	1	INT27	1 (Lo)	0	80.0
FIC-EA1-1	1	INT28	1 (Lo)	0	3000.0
PI-EA1-1	1	INT29	1 (Lo)	0	17.7
PI-EA1-1	1	INT30	0 (Hi)	0	29.7
TI-EA1-5	1	INT31	0 (Hi)	0	190.0

When one EPN has more than one white alarm, the more extreme alarm has priority. This means that a HiHi has priority over a Hi alarm, and a LoLo has priority over a Lo. When the more extreme alarm resets, the less extreme alarm becomes active again. A Lo alarm resets when the measured value is greater than the Lo alarm limit plus AI scan block deadband, and a Hi alarm resets when the measured value is less than the Hi alarm limit minus the deadband.

5.6.7 Device Alarms

A few programs handle alarms associated with logic devices. The code for device alarms is very similar to the code for the digital alarms, except that a) the code for device alarms is usually hard-coded, whereas the code for most DINs use pointers and tables, and b) device alarm programming uses DEV_STAT and DEV_ALMCT, whereas the code for DINs use DIN_VAL and DIN_ALMCT.

5.7 White Alarm Summary Program

The white alarm summary page program running in PCM1 and found in unit WHOLE1 is called WHITE1. This program runs independently of all other batch programs. The alarm values are hard-coded into this lengthy SABL program. Signals reset as described in the last paragraph of Section 5.6.6. As new alarms occur, they are displayed at the top of the list, much like P1 and P2 alarms, and cause SKID key 60 to flash. Pressing SKID key 60 causes it to return to a dark-board condition. White alarms are eliminated from the list when inhibited or when they clear.

The flags and integers listed below are used differently than other programs. Pointer names are in parenthesis. The maximum of the range, 191, is the total number of each variable in the UPF. Currently, 175 of each variable are being used, with many of these being spares or old, abandoned white alarms. The letter "n" in the variable name indicates a number between 1 and 175.

VARIABLE (POINTER)	TYPE	RANGE	PURPOSE
FXn (PSTAT)	FLAG	0 - 191	For DINs, is the real-time digital input value. For AIs, is set to 1 if AI is in an alarm state.
FRVn (PREV)	FLAG	0 - 191	FRV is the predetermined value of a digital input when in the alarm state. An alarm exists when PREV equals PSTAT. Set equal to 1 for all analog alarms.

FDSn (PTYPE)	FLAG	0 - 191	If equal to 1, then alarm is inhibited.
FALn (PALARM)	FLAG	0 - 191	Set to 1 when an alarm is found to prevent repeat. Is reset to 0 when alarm clears.
INTn (PLINE)	INTEGER	0 - 191	Is the line in the white alarm list display where the alarm message resides. If 0, then no alarm.

The white alarm program first initializes all alarm conditions by setting flags FRVn and FDSn. The inhibit flag, FDSn, for each signal is initialized as a 1. For signals that have white alarms, the program then reads the DIN_ALMCT, AI_ALMCT, or DEV_ALMCT bit values (at the start of the program repeat loop, AGAIN:) to determine if they are really inhibited or not and sets FDSn accordingly. Next, it determines the real-time signal status and sets flags FXn to the value of each DIN or analog alarm. If a signal is not inhibited (if not, PTYPE = FDSn = 0), then the program checks if to see if the signal was previously alarmed (if not, PALARM = FALn = 0). If not inhibited and not previously alarmed, then the program checks to see if a new alarm exists. A new alarm exists when FXn equals FRVn (when PREV = PSTAT). If a new alarm is found, then the program does the following:

- Set PALARM (FALn) equal to 1
- Set PLINE (INTn) equal to ILOOP
- Flash the SKID lamp
- Index, or bump, all messages in the white alarm summary graphics by 1.

If a signal is inhibited (PTYPE = FDSn = 1) and it was previously alarmed (PALARM = FALn = 1) then the program clears the alarm messages for that signal from the alarm summary graphic. To accomplish this the program does the following:

- Set PALARM (FALn) equal to 0
- Step to the alarm message line displayed in the graphic. That is, step down ILOOP lines from first line (STEP PLINE BY ILOOP).
- Step all message lines backwards to erase the alarm message and set PLINE (INTn) equal to 0.

A signal that once had a white alarm which was deleted is handled in a unique fashion. This is because it was felt that some white alarms might be reinstated later. For all signals, FDSn (PTYPE) is a constant set equal to 1 (PTYPE = 1 for inhibited) in the initialization section of the code. When the inhibit flag, PTYPE, is set the program bypasses the alarm status checking code contained in the program loop labeled "LOP:", and alarm messages of inhibited signals are kept out of the white alarm summary graphics. By taking

advantage of this, white alarms are essentially deleted by commenting specific lines of code pertaining to the individual EPNs. Reinstating deleted white alarms is a simple programming matter of removing semi-colons in commented lines. An example of lines of code that have been commented to delete AW Farm pump device white alarms are listed below (see Appendix E for the actual code). Similar lines of code would be commented for deleted analog or digital white alarms.

- In the initialization section, definition of FRV70 through FRV75
- In the AGAIN program loop, current settings of FDS70 through FDS75 which reads the DEV_ALMCT bit values for inhibit checking
- In the AGAIN program loop, current device statuses FX70 through FX75
- In the alarm subroutine, "ALM:", graphic message strings from labels "S70:" through "S75:"

Appendix A indicates the EPNs which have white alarms associated with them. This is indicated in the "White Numb" column. For example, the digital signal PSH-FP in unit A244 is a white alarm. The number "084" refers to the variable number or label (ie: FX84, FRV84, FDS84, FAL84, INT84, and "S84:") in the white alarm program that deals with this signal.

5.8 Unit Relative Feature of Programs

All MCS SABL programs using the same UPF are unit relative, which means several things. First, the code from one program to the next is similar if not identical, which is wonderful from a programmers standpoint. However, the 242-A process does not lend itself to unit relativity because there is no similarity between process units as in a production facility. As a result, there is a down-side to this unit relativity. For example, ANBATCH can run in unit APFARM, or CA1BATCH can run in unit AMU. This is highly undesirable because if this were to happen false alarms would be presented to the operator. To prevent this, a GETUNIT command early in each program determines which sequence unit a program is running in. If a program is running in the wrong unit then the program aborts, a SKID key (key 48 labeled "Program Status") flashes, and a message is printed at the system alarm typer.

5.9 Program Statuses, Diagnostics, and Corrective Action

Two SABL programs, one in each PCM, monitor the status of all sequence programs. PGM0A runs in the PCM0 processors, and PGM1A runs in the PCM1 processors. Program statuses are displayed on graphic 43. Normal or healthy statuses (RUNNING, WAIT, or CRITICAL) are shown in a steady green color. Abnormal statuses (NOT-RUNNING, TRAPPED, or FAULT on AUDALARM0) are displayed in a flashing red color on the graphic and cause the Program Status SKID key (key 48) to flash. The abnormal statuses, NOT-RUNNING and TRAPPED, require corrective action (aborting, unloading, reloading, and running the indicated

program). Command files in directory D3DATA:[100001] of the CDCM are used to correct these problems.

In the Program Status graphic, error lines and error codes are shown in white for diagnostics. Error number 187 (COULD NOT DO REMOTE ACCESS) is the most common error code shown, if any. A few SABL programs must get data from another PCM (ie: PCM0A program RWBATCH must get the status of DIN TSL242A81 from PCM1A). If the requesting PCM cannot communicate with the other PCM for any reason, then error 187 will occur. Error 187 will occur anytime one PCM is offline or in load-request. There are a host of other reasons why communications may be lost. Most are temporary. Temporary loss of communications usually indicates some hardware problem exists, but could be the result of a data highway that is too busy.

Another occasional problem occurs when the yellow-white alarm horn becomes stuck and cannot be shut off for some reasonable duration. To diagnose this, access the Program Diagnostic display, graphic 133, and observe the flags FALMON, ACK, FLGACK, and FSL0. These flags should normally be green or temporarily red.

5.10 "C" Programs Related To Alarm Scheme

There are six alarm-related "C" programs that enhance the alarm management scheme. These all work together to minimize operator time and effort to find and acknowledge alarms. The standard TI D/3 package requires that each alarm in a graphic be selected and acknowledged individually. The six "C" programs get around this feature so that pressing the operator console "ACK" key once acknowledges all alarms in the graphic. The only requirement for this to work is that the alarm acknowledge device in a graphic must be selected. The alarm acknowledge device is automatically selected when a graphic is called up. However, the operator may have to re-select this device (using Quick Select key 0) if another EPN was temporarily selected. The alarms in a graphic are also acknowledged throughout the entire D/3 system.

Four of the "C" programs run in the PCMs. One program runs in each PCM for redundancy. The other two programs run on the CDCM. One of these two tasks watches flags in the PCM sequence databases, and the other acknowledges all alarms in a graphic. The programs are named as follows:

Micro Name	Program Name
PCM0A	DOG0A
PCM0B	DOG0B
PCM1A	DOG1A
PCM1B	DOG1B
CDCM	WATCH (installed as task DOG)

Appendix F contains the code listings of these programs.

The relationships between these programs are illustrated in Figure 1. When a white or yellow alarm occurs, a sequence program sets the flag FALMON to a value of one. This flag is scanned by a "C" program running in the PCM called DOG0A, DOG0B, DOG1A, or DOG1B (hereafter referred to as DOGxx) depending on the PCM. If a DOGxx program finds FALMON set, then DOGxx commands the audible alarm device (AUDALARM0) on to turn on the horn, and resets the FALMON flag. Also, DOG0A and DOG0B look at flag ACK (in PCM0 units), and DOG1A and DOG1B look at the FLGACK flag (in PCM1 units). If ACK is set to a value of one, then either DOG0A or DOG0B (depending on the PCM selected) sets flag FSL0 to a value of one in unit BLANK0. If FLGACK is set to a value of one, then either DOG1A or DOG1B (depending on the PCM selected) sets flag FSL0 to a value of one in unit BLANK1. The WATCH program scans the FSL0 flag in each of these two BLANKx units. If WATCH finds that either flag is set, then WATCH resets the flag and runs function ACKFLG. The function ACKFLG acknowledges all flashing alarms in the displayed graphic from which the operator console "Alarm ACK" key was pressed. Conversely, if the "Alarm ACK" key was pressed from a P1 or P2 display, then those alarms are acknowledged in the graphics where they are found.

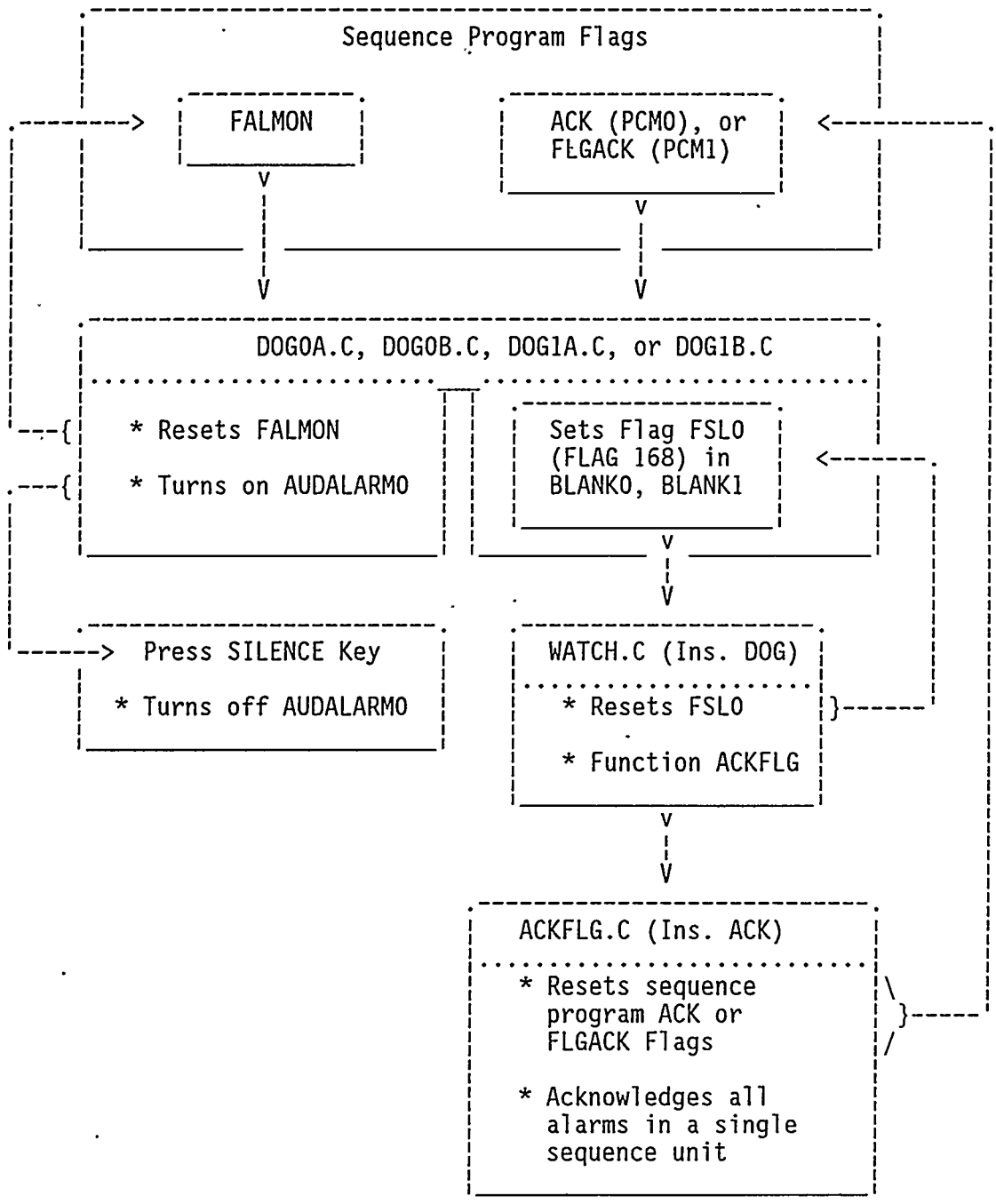


FIGURE 1. Program Relationships

5.11 Other Alarm-Related "C" Programs

There are four other programs that deal with inhibited alarms. The code listings of these programs are found in Appendix F. One of these is INHA.C, which is installed on a DCM as task INHALM. Task INHALM is run from an annunciator SKID panel by pressing key number 53, which is labeled "INHA". This task provides a list of inhibited alarms for graphics 80 through 84. It works in conjunction with SABL program TIMEINHA, which provides the date and time that the program was run. The date and time are displayed on each of the graphics numbered 80 through 84. TIMEINHA is also activated by pressing SKID key 53.

The other three programs are user key functions (UKFs), or General Functions (GFs). General Functions 43 and 44 are activated after the Evaporator is shut down to inhibit select alarms. UKF43 is run only on the CDCM and inhibits the alarms and clears the CDCM alarm summary pages. UKF44 is then run on DCM0 to clear its alarm summary pages. General Function 14 (provided by TI) is then run on the CDCM to recreate the alarms in the alarm summary pages.

Conversely, UKF34 is activated prior to starting the Evaporator up and cancels the inhibited status of the same select EPNs. After UKF34 is run, no other tasks or user key functions need to be run.

5.12 SKID Key Configuration

The annunciator key configuration for each OCM is built from command files in directory D3DATA:[100010] of the CDCM. These files are named CDCMSK0.CMD, CDCMSK1.CMD, DCM0SK0.CMD, DCM0SK1.CMD, and DCM0SK2.CMD. These five files are identical. Appendix I contains a listing of CDCMSK0.CMD.

6.0 REBUILDING DATABASES AND PROGRAMS

A word about rebuilding databases is appropriate at this point. Whenever the continuous database is rebuilt, a new serial number is automatically assigned to that database. This serial number is the fine thread that is associated with the sequence database and graphics. Its purpose is to warn when something has been rebuilt, but something else has not. When such discrepancies occur a large flashing red message, such as "... SERIAL MISMATCH", will appear in a graphic.

When the continuous database is rebuilt, usually most everything else must be rebuilt. The nice thing about the RPFs and UPFs is that unless they have been modified, they do not need to be rebuilt. However, the sequence database, sequence programs, and graphics must be rebuilt. If only one continuous database is rebuilt, it is safest to <u>also</u> rebuild the sequence database, programs, and graphics for the continuous database that was <u>not</u> rebuilt. This will take quite a bit more time, but in the long run it could save some grief. The reason for taking the extra time is that the same cross-reference files (D3DATA:[300001]SDBMASTER.XRF, SDBSHORT.XRF) are written to when the SDB utility is run for PCM0 or PCM1. It is best to start off clean by renaming the "XRF" files to "OLD" files, and let SDB write new files. A procedure for rebuilding the entire applications software is found in Appendix H.

7.0 APPENDICES

APPENDIX A - IPN ORDER

MCS EPN	SIG TYPE	IPN	SDB INT	FIRST GRAPH	ACAD FILE NAME	WHITE NUMB.	NUMBER PER SEQ UNIT	PCM
** - - - S D B U N I T N A M E - - -								
0 - - - W H O L E 0 - - -								
244ALARM	DEV	0	-	-	-		(217)	
YS244AIL	DI1	0	-	-	-		(348)	
II-P244A1	AI1	0	-	-	-		(379)	
1 - - - A 2 4 4 - - -								
244ALARM	DEV	0	-	340	244A		(15)	0
P-244-A1	DEV	1	-	340	244A			0
P-244-A2	DEV	2	-	340	244A			0
P-244-A3	DEV	3	-	340	244A			0
A-244-A	DEV	4	-	340	244A			0
HV-MOV-2	DEV	5	-	340	244A			0
YS244A1IL	DI1	0	INT0	340	244A		(50)	0
YS-P244A1	DI1	1	INT1	340	244A			0
YS244A2IL	DI1	2	INT2	340	244A			0
YS-P244A2	DI1	3	INT3	340	244A			0
YS-P244A3	DI1	4	INT4	340	244A			0
YS-A-244A	DI1	5	INT5	340	244A			0
ZS-MOV-2F	DI1	6	INT6	340	244A			0
ZS-MOV2DR	DI1	7	INT7	340	244A			0
ZS-244A11	DI1	8	INT8	340	244A			0
ZS-244A12	DI1	9	INT9	340	244A			0
ZS-244A13	DI1	10	INT10	340	244A			0
ZS-244A14	DI1	11	INT11	340	244A			0
PS-244A11	DI1	12	INT12	340	244A			0
PS-244A12	DI1	13	INT13	340	244A			0
PS-244A13	DI1	14	INT14	340	244A			0
LDS-PP	DI1	17	INT17	340	244A			0
RSH-SP	DI1	18	INT18	340	244A			0
PSH-FP	DI1	19	INT19	340	244A	084		0
II-P244A1	AI1	0	AI0	340	244A		(30)	0
II-P244A2	AI1	1	AI1	340	244A			0
II-P244A3	AI1	2	AI2	340	244A			0
DI-244-SP	AI1	3	AI3	340	244A			0
WFI-244SP	AI1	4	AI4	340	244A			0
II-A-244A	AI1	5	AI5	340	244A			0
LI-244TK	AI	6	AI6	340	244A			0
DI-244-TK	AI1	7	AI7	340	244A			0
WFI-244TK	AI1	8	AI8	340	244A			0
TI-244-TK	AI1	9	AI9	340	244A	090		0
PI-244-TK	AI1	10	AI10	340	244A			0

MCS EPN	SIG TYPE	IPN	SDB INT	FIRST GRAPH	ACAD FILE NAME	WHITE NUMB.	NUMBER PER SEQ UNIT	PCM
2 - - - H V C 2 4 4 - - -								
HVAC244	DEV	10	-	346	244AHVAC1		(1)	0
PDSL-EF-2	DI1	20	INT0	341	244AHVAC		(5)	0
PDSH-EF-1	DI1	21	INT1	341	244AHVAC			0
RSH-A	DI1	22	INT2	341	244AHVAC			0
RSH-EXH-1	DI1	23	INT3	341	244AHVAC			0
QXS-EXH-1	DI1	24	INT4	341	244AHVAC			0
RI-244ANN	AI1	11	AI0	341	244AHVAC		(5)	0
FI-244EXH	AI1	12	AI1	341	244AHVAC	091,092		0
RI-244EXH	AI1	13	AI2	341	244AHVAC			0
3 - - - W A S T E - - -								
WSTXFR	DEV	11	-	349	WSTXFR1		(1)	0
LDS-AR151	DI1	25	INT0	344	WSTXFR		(18)	0
LDS-ER151	DI1	26	INT1	344	WSTXFR			0
PSHSNV229	DI1	27	INT2	344	WSTXFR			0
LDS-ER152	DI1	28	INT3	344	WSTXFR			0
ZSER15212	DI1	29	INT4	344	WSTXFR			0
ZSER15211	DI1	30	INT5	344	WSTXFR			0
PSER15212	DI1	31	INT6	344	WSTXFR			0
PSER15211	DI1	32	INT7	344	WSTXFR			0
LDS-ER153	DI1	33	INT8	344	WSTXFR			0
PSHPSN232	DI1	34	INT9	344	WSTXFR			0
PSHPSN234	DI1	35	INT10	344	WSTXFR			0
ZSER15312	DI1	36	INT11	344	WSTXFR			0
ZSER15311	DI1	37	INT12	344	WSTXFR			0
PSER15312	DI1	38	INT13	344	WSTXFR			0
PSER15311	DI1	39	INT14	344	WSTXFR			0
LDS-AX155	DI1	40	INT15	344	WSTXFR			0
LDS-DR713	DI1	41	INT16	344	WSTXFR			0
LDS-DR717	DI1	42	INT17	344	WSTXFR			0
LDS216A40	DI1	243	INT18	344	WSTXFR			1
4 - - - A 3 5 0 - - -								
350ALARM	DEV	12	-	343	350A		(3)	0
P-350-1	DEV	13	-	343	350A			0
LSH-350-2	DI1	44	INT1	343	350A		(5)	0
LDS-350-2	DI1	45	INT2	343	350A			0
LDS-PW481	DI1	46	INT3	343	350A			0

MCS EPN	SIG TYPE	IPN	SDB INT	FIRST GRAPH	ACAD FILE NAME	WHITE NUMB.	NUMBER PER SEQ UNIT	PCM
II-350-1	AI1	14	AI0	343	350A		(17)	0
LI-350TK	AI	15	AI1	343	350A			0
DI-350-TK	AI1	16	AI2	343	350A			0
WFI-350TK	AI1	17	AI3	343	350A			0
FQI-350-2	AI	18	AI4	343	350A			0
FQI-350-1	AI	19	AI5	343	350A			0
FI-350-1	AI	20	AI6	343	350A			0
5 - - - F E E D - - -								
PAW102ALM	DEV	15	DEV0	301	AWFEED		(10)	0
P-AW-102	DEV	16	DEV1	301	AWFEED			0
HV-CA1-1	DEV	17	DEV2	301	AWFEED			0
VCA1-INLK	DEV	18	DEV3	301	AWFEED			0
YS-AW102	DI1	50	INT0	301	AWFEED		(10)	0
YS-CA1-1	DI1	51	INT1	301	AWFEED			0
LDS-SMPL1	DI1	52	INT2	301	AWFEED			0
PSH-102-3	DI1	53	INT3	301	AWFEED			0
YSAW102IL	DI1	54	INT4	301	AWFEED			0
ZS-CA1-1C	DI2	55	INT5	301	AWFEED			0
ZS-CA1-1N	DI2	56	INT6	301	AWFEED			0
II-AW-102	AI1	30	AI0	301	AWFEED	094	(15)	0
DI-AW-102	AI1	31	AI1	301	AWFEED			0
FIC-CA1-1	AI2	32	AI2	301	AWFEED			0
FQI-CA1-1	AI	33	AI3	301	AWFEED			0
KY-CA1-1	AI	34	AI4	301	AWFEED			0
CI-WVR-2	AI	35	AI5	301	AWFEED			0
CI-WVR-3	AI	36	AI6	301	AWFEED			0
CI-WVR-1	AI	37	AI7	301	AWFEED			0
PI-102-AW	AI1	38	AI8	301	AWFEED			0
KY-102-3	AI	39	-	301	AWFEED			0
6 - - - C A 1 - - -								
CA1-ALARM	DEV	25	DEV0	010	CA1		(10)	0
SELECT-WF	DEV	26	DEV1	010	CA1			0
SELECT-DI	DEV	27	DEV2	010	CA1			0
HV-PDSPRY	DEV	28	DEV3	010	CA1			0
PSH-CA111	DI1	60	INT0	010	CA1		(10)	0
WFSH-CA11	DI1	61	INT1	010	CA1			0
WFSH-CA12	DI1	62	INT2	010	CA1			0
PDSH-CA1	DI1	63	INT3	010	CA1			0
PI-CA1-11	AI1	45	AI0	010	CA1		(35)	0
PIC-CA1-7	AI1	46	AI1	010	CA1			0

MCS EPN	SIG TYPE	IPN	SDB INT	FIRST GRAPH	ACAD FILE NAME	WHITE NUMB.	NUMBER PER SEQ UNIT	PCM
TI-CA1-6S	AI5	47	AI2	010	CA1			0
TI-CA1-6	AI5	48	AI3	010	CA1			0
TDI-BPR-1	AI	49	AI4	010	CA1			0
WFI-CA1-1	AI1	50	AI5	010	CA1			0
LI-CA1-1	AI	51	AI6	010	CA1			0
LI-CA1-1G	AI	52	AI7	010	CA1			0
LIC-CA1-1	AI	53	AI8	010	CA1			0
WFI-CA1-2	AI1	54	AI9	010	CA1			0
LI-CA1-2	AI	55	AI10	010	CA1			0
LI-CA1-2G	AI	56	AI11	010	CA1			0
LIC-CA1-2	AI	57	AI12	010	CA1			0
DIC-CA1-1	AI	58	AI13	010	CA1			0
DI-CA1-1	AI1	59	AI14	010	CA1			0
DI-CA1-2	AI1	60	AI15	010	CA1			0
DDI-CA1-1	AI	61	AI16	010	CA1			0
WFI-CA1-3	AI1	62	AI17	010	CA1			0
LI-CA1-3	AI	63	AI18	010	CA1			0
LI-CA1-3G	AI	64	AI19	010	CA1			0
PDI-FH1-1	AI1	65	AI20	010	CA1	098		0
PDI-FH2-1	AI1	66	AI21	010	CA1	099		0
PDI-FH3-1	AI1	67	AI22	010	CA1	100		0
FIC-CA1-6	AI2	68	AI23	010	CA1			0
FQI-CA1-6	AI	69	AI24	010	CA1			0
PDI-CA1-2	AI1	70	AI25	010	CA1			0
HIC-CA114	AO1	71	AI26	010	CA1			0
HIC-CA115	AO1	72	AI27	010	CA1			0
PDI-CA1-1	AI1	73	AI28	010	CA1			0
TI-CA1-10	AI	74	AI29	010	CA1			0
7 - - - P B 1 - - -								
PB1-ALARM	DEV	35	DEV0	012	PB1		(15)	0
PB-1	DEV	36	DEV1	012	PB1			0
PB1WFINLK	DEV	37	DEV2	012	PB1			0
PB1-BYPAS	DEV	38	DEV3	012	PB1			0
HV-CA1-7	DEV	39	DEV4	012	PB1			0
HV-CA1-9	DEV	40	DEV5	012	PB1			0
BOT-DUMP	DEV	41	DEV6	012	PB1			0
HV-SUMP-1	DEV	42	DEV7	012	PB1			0
JGV-SUMP	DEV	43	DEV8	012	PB1			0
P-C-105A	DEV	44	DEV9	012	PB1			0
P-C-105	DEV	45	DEV10	012	PB1			0
DUMP-INLK	DEV	46	-	012	PB1			0
YS-PC-105	DI1	70	INT0	012	PB1		(15)	0
YS-PC105A	DI1	71	INT1	012	PB1			0
ZS-CA1-7N	DI1	72	INT2	012	PB1			0
YS-PB1-1	DI1	73	INT3	012	PB1			0
ZS-CA1-7C	DI1	74	INT4	012	PB1			0

MCS EPN	SIG TYPE	IPN	SDB INT	FIRST GRAPH	ACAD FILE NAME	WHITE NUMB.	NUMBER PER SEQ UNIT	PCM
LDS-SMPL2	DI1	75	INT5	012	PB1			0
ZS-CA1-8N	DI1	76	INT6	012	PB1			0
ZS-CA1-8C	DI1	77	INT7	012	PB1			0
ZS-CA1-9C	DI1	78	INT8	012	PB1			0
ZS-CA1-9N	DI1	79	INT9	012	PB1			0
ZS-SUMP1C	DI1	80	INT10	012	PB1			0
ZS-SUMP1N	DI1	81	INT11	012	PB1			0
KY-PB1-1	AI	80	AI0	012	PB1		(20)	0
FI-CA1-7	AI2	81	AI1	012	PB1			0
II-PB1-1	AI1	82	AI2	012	PB1			0
VI-PB1-2A	AI1	83	AI3	012	PB1			0
VI-PB1-1A	AI1	84	AI4	012	PB1			0
VI-PB1-3A	AI1	85	AI5	012	PB1			0
FI-CA1-1	AI2	86	AI6	012	PB1			0
PI-CA1-9	AI1	87	AI7	012	PB1			0
FQI-PB1-W	AI	88	AI8	012	PB1			0
FI-CA1-3	AI2	89	AI9	012	PB1			0
TI-CA1-7	AI6	90	AI10	012	PB1			0
WFI-SUMP1	AI1	91	AI11	012	PB1			0
DI-SUMP-1	AI1	92	AI12	012	PB1			0
LI-SUMP1	AI	93	AI13	012	PB1			0
PI-SUMP-1	AI1	94	AI14	012	PB1			0
8 - - - P B 2 - - -								
PB2-ALARM	DEV	50	DEV0	015	PB2		(15)	0
PB-2	DEV	51	DEV1	015	PB2			0
HV-CA1-2	DEV	52	DEV2	015	PB2			0
HV-CA1-2S	DEV	53	DEV3	015	PB2			0
HV-CA1-5	DEV	54	DEV4	015	PB2			0
HV-CA1-6	DEV	55	DEV5	015	PB2			0
JGV-STMFL	DEV	56	DEV6	015	PB2			0
CONTROL	DEV	57	DEV7	015	PB2			0
PB2SLFILK	DEV	58	DEV8	015	PB2			0
PB2WFINLK	DEV	59	DEV9	015	PB2			0
YS-PB2-1	DI1	85	INT0	015	PB2		(20)	0
YS-PB2-IL	DI1	86	INT1	015	PB2			0
TSH-PB2-1	DI1	87	INT2	015	PB2			0
PSH-CA1-3	DI1	88	INT3	015	PB2			0
PSH-PB2-1	DI1	89	INT4	015	PB2			0
SXS-PB2-1	DI1	90	INT5	015	PB2			0
YS-VFD-1	DI1	91	INT6	015	PB2			0
SSL-PB2-1	DI1	92	INT7	015	PB2			0
FQA-RW-1	DI	93	INT8	015	PB2			0
ZS-CA1-21	DI2	94	-	015	PB2			0
ZS-CA1-22	DI2	95	-	015	PB2			0
ZS-CA12A1	DI2	96	-	015	PB2			0

MCS EPN	SIG TYPE	IPN	SDB INT	FIRST GRAPH	ACAD FILE NAME	WHITE NUMB.	NUMBER PER SEQ UNIT	PCM
ZS-CA12A2	DI2	97	-	015	PB2			0
ZS-CA1-6N	DI1	98	-	015	PB2			0
ZS-CA1-6C	DI1	99	-	015	PB2			0
ZS-CA1-54	DI2	100	-	015	PB2			0
ZS-CA1-55	DI2	101	-	015	PB2			0
HI-WVR-1	AI	100	AI0	015	PB2		(20)	0
II-PB2-1	AI1	101	AI1	015	PB2			0
SIC-PB2-1	AI1	102	AI2	015	PB2	101		0
VI-PB2-1A	AI1	I03	AI3	015	PB2			0
PI-CA1-8	AI1	104	AI4	015	PB2			0
FQI-PB2-W	AI	105	AI5	015	PB2			0
FI-CA1-2	AI2	106	AI6	015	PB2			0
PI-CA1-10	AI1	107	AI7	015	PB2			0
KY-CA1-2	AI	108	AI8	015	PB2			0
KY-CA1-2F	AI	109	AI9	015	PB2			0
FIC-CA1-4	AI2	110	AI10	015	PB2			0
FQI-CA1-4	AI	111	AI11	015	PB2			0
PI-CA1-5	AI1	112	AI12	015	PB2			0
FQI-RW-1	AI	113	AI13	015	PB2			0
FQI-RW-1B	AI	114	AI14	015	PB2			0
9 - - - E A 1 - - -								
EA1-ALARM	DEV	65	DEV0	013	EA1		(15)	0
FV-EA1-1	DEV	66	DEV1	013	EA1			0
EA1-INLK1	DEV	67	DEV3	013	EA1			0
EA1-INLK2	DEV	68	DEV4	013	EA1			0
TV-DSH1/2	DEV	69	DEV5	013	EA1			0
SELECT-TI	DEV	70	DEV6	013	EA1			0
PSL-EA1-1	DI1	105	INT0	013	EA1	102	(5)	0
TSH-DSH-3	DI1	106	INT1	013	EA1			0
ZS-EA1-1C	DI1	107	INT2	013	EA1			0
TI-EA1-7	AI6	120	AI0	013	EA1		(35)	0
TI-EA1-7S	AI6	121	AI1	013	EA1			0
TI-EA1-1	AI6	122	AI2	013	EA1			0
TI-EA1-1S	AI6	123	AI3	013	EA1			0
TDI-EA1-1	AI	124	AI4	013	EA1			0
PI-EA1-12	AI1	125	AI5	013	EA1			0
PI-EA1-2	AI1	126	AI6	013	EA1			0
PDI-EA1-1	AI	127	AI7	013	EA1			0
PI-STM-1	AI1	128	AI8	013	EA1			0
PI-EA1-13	AI1	129	AI9	013	EA1	103,104		0
PI-EA1-14	AI1	130	AI10	013	EA1			0
FIC-EA1-1	AI1	131	AI11	013	EA1	105		0
ZI-EA1-1	AI1	132	AI12	013	EA1			0
TI-DSH-2	AI6	133	AI13	013	EA1			0

MCS EPN	SIG TYPE	IPN	SDB INT	FIRST GRAPH	ACAD FILE NAME	WHITE NUMB.	NUMBER PER SEQ UNIT	PCM
PI-EA1-1	AI1	134	AI14	013	EA1	106,107		0
PI-EA1-11	AI1	135	AI15	013	EA1			0
PDI-EA1-2	AI	136	AI16	013	EA1			0
TI-DSH-3	AI1	137	AI17	013	EA1			0
TI-EA1-5	AI5	138	AI18	013	EA1	108		0
CI-EA1-U2	AI	139	AI19	013	EA1			0
CI-EA1-U1	AI	140	AI20	013	EA1			0
10 - - - E C 1 - - -								
EC1-ALARM	DEV	80	DEV0	016	EC1		(10)	0
HV-EC1-1	DEV	81	DEV1	016	EC1			0
HVEC2/3-1	DEV	82	DEV2	016	EC1			0
RSH-CA1-1	DI1	110	INT0	016	EC1		(10)	0
ZS-EC1-1N	DI1	111	-	016	EC1			0
ZS-EC1-1C	DI1	112	-	016	EC1			0
ZSEC2/31N	DI1	113	-	016	EC1			0
ZSEC2/31C	DI1	114	-	016	EC1			0
TI-CA1-9	AI5	150	AI0	016	EC1		(25)	0
TI-EC1-3	AI4	151	AI1	016	EC1			0
TI-EC2-3	AI4	152	AI2	016	EC1			0
PI-EC1-4	AI1	153	AI3	016	EC1			0
PI-EC2-2	AI1	154	AI4	016	EC1			0
CI-EC2-Q1	AI	155	AI5	016	EC1			0
CI-EC3-Q1	AI	156	AI6	016	EC1			0
ZI-CA1-7	AI1	157	AI7	016	EC1			0
FI-EC2/3	AI1	158	AI8	016	EC1			0
TI-EC1-4	AI5	159	AI9	016	EC1			0
PI-EC1-5	AI1	160	AI10	016	EC1	109		0
TI-EC2-1	AI4	161	AI11	016	EC1			0
FIC-EC3-1	AI1	162	AI12	016	EC1	110		0
TI-EC3-1	AI4	163	AI13	016	EC1			0
TI-EC3-2	AI4	164	AI14	016	EC1			0
TI-EC2-2	AI4	165	AI15	016	EC1			0
PI-EC3-2	AI1	166	AI16	016	EC1			0
TI-EC3-3	AI4	167	AI17	016	EC1			0
RI-CA1-1	AI3	168	AI18	016	EC1			0
FI-EC1-2	AI2	169	AI19	016	EC1			0
FQI-EC1-2	AI	170	AI20	016	EC1			0
TIC-EC1-2	AI4	171	AI21	016	EC1	111		0
FQI-EC2/3	AI	172	AI22	016	EC1			0
CI-EC1-U1	AI	173	-	016	EC1			0

MCS EPN	SIG TYPE	IPN	SDB INT	FIRST GRAPH	ACAD FILE NAME	WHITE NUMB.	NUMBER PER SEQ UNIT	PCM
11 - - - R W A T E R - - -								
RW-ALARM	DEV	90	DEV0	026	RW1		(10)	0
HV-RW-1	DEV	91	DEV1	011	RW		RWATER	0
HV-RW-2	DEV	92	DEV2	011	RW			0
LDS-BFP-1	DI1	120	INT0	011	RW	112	(10)	0
LSL-RWSMP	DI1	121	INT1	011	RW	113	RWATER	0
LDS-BFP-2	DI1	122	INT2	011	RW	114		0
RSH-RW-1	DI1	123	INT3	011	RW			0
RXS-RW-1	DI1	124	INT4	011	RW			0
ZS-RW-1C	DI1	125	-	011	RW			0
ZS-RW-1N	DI1	126	-	011	RW			0
ZS-RW-2C	DI1	127	-	011	RW			0
ZS-RW-2N	DI1	128	-	011	RW			0
PDI-RW-1	AI1	180	AI0	011	RW	115	(10)	0
PI-RW-1	AI1	181	AI1	011	RW		RWATER	0
PDI-RW-2	AI1	182	AI2	011	RW	116		0
RI-RW-1	AI3	183	AI3	011	RW			0
PI-RW-2	AI1	184	AI4	011	RW	117		0
TSL242A81	DI1	242	INT13	011	RW			1
12 - - - U R W - - -								
URW-ALARM	DEV	100	DEV0	017	URW		(10)	0
RC2-SAMP	DEV	101	DEV1	017	URW			0
RC2-PIG	DEV	102	DEV2	017	URW			0
FSL-RC2-1	DI1	130	INT0	017	URW		(10)	0
RSH-RC2-1	DI1	131	INT1	017	URW			0
RXS-RC2-1	DI1	132	INT2	017	URW			0
YS-RC2-6	DI1	133	INT3	017	URW			0
FIC-EC1-1	AI1	190	AI0	017	URW	119	(15)	0
TI-EC1-1	AI4	191	AI1	017	URW			0
ZI-EC1-1	AI1	192	AI2	017	URW			0
FI-RC2-1	AI1	193	AI3	017	URW			0
FQI-RC2-1	AI	194	AI4	017	URW			0
FQIC-RC2	AI	195	AI5	017	URW			0
RI-RC2-1	AI3	196	AI6	017	URW			0
PIC-EC1-2	AI	197	AI7	017	URW			0
PI-EC1-9	AI1	198	AI8	017	URW			0
ZI-EC1-2	AI1	199	AI9	017	URW			0
KY-RC2-1	AI	200	AI10	017	URW			0

MCS EPN	SIG TYPE	IPN	SDB INT	FIRST GRAPH	ACAD FILE NAME	WHITE NUMB.	NUMBER PER SEQ UNIT	PCM
13 - - - V V E N T - - -								
VV-ALARM	DEV	110	-	020	VVENT		(10)	0
EX-C-1	DEV	111	-	020	VVENT			0
EXC1-INLK	DEV	112	-	020	VVENT			0
PDSH-FC51	DI1	140	INT0	020	VVENT		(20)	0
PDSL-FC51	DI1	141	INT1	020	VVENT			0
ZS-FC5-1	DI1	142	INT2	020	VVENT			0
PDSH-FC52	DI1	143	INT3	020	VVENT			0
ZS-FC5-2	DI1	144	INT4	020	VVENT			0
YS-EXC-1	DI1	145	INT5	020	VVENT			0
YS-EXC3RL	DI1	146	INT6	020	VVENT			0
TSHH-HC13	DI1	147	INT7	020	VVENT			0
RXS-VVALP	DI1	148	INT8	020	VVENT			0
RSH-VVALP	DI1	149	INT9	020	VVENT			0
RXS-VVB/G	DI1	150	INT10	020	VVENT			0
RSH-VVB/G	DI1	151	INT11	020	VVENT			0
QXS-VV-1	DI1	152	INT12	020	VVENT			0
(SPARE)	-	153	INT13	-	-			0
PDSH-AS-1	DI1	154	INT14	020	VVENT			0
PDSL-AS-2	DI1	155	INT15	020	VVENT			0
AXS-NH3-1	DI1	156	INT16	020	VVENT			0
PDI-FC5-2	AI1	205	AI0	020	VVENT		(20)	0
PDI-FC5-1	AI1	206	AI1	020	VVENT			0
PDI-DUC1	AI1	207	AI2	020	VVENT			0
II-EXC-1	AI1	208	AI3	020	VVENT	121		0
PDI-FC6-1	AI1	209	AI4	020	VVENT			0
FQI-AS-5	AI	210	AI5	020	VVENT			0
TI-HC1-1	AI5	211	AI6	020	VVENT			0
TI-HC1-2	AI5	212	AI7	020	VVENT			0
TI-FC5-1	AI5	213	AI8	020	VVENT			0
TDIC-HC11	AI	214	AI9	020	VVENT			0
FI-AS-5	AI1	215	AI10	020	VVENT			0
PDI-FC512	AI	216	AI11	020	VVENT			0
RI-VVALP	AI3	217	AI12	020	VVENT			0
RI-VVB/G	AI3	218	AI13	020	VVENT			0
II-AS-1	AI1	219	AI14	020	VVENT	126		0
AI-NH3-1	AI2	220	AI15	020	VVENT			0
14 - - - P C - - -								
PC-ALARM	DEV	120	-	018	PC		(10)	0
A-C100	DEV	121	-	018	PC			0
P-C100	DEV	122	-	018	PC			0
PC-INLK1	DEV	123	-	018	PC			0

MCS EPN	SIG TYPE	IPN	SDB INT	FIRST GRAPH	ACAD FILE NAME	WHITE NUMB.	NUMBER PER SEQ UNIT	PCM
YS-A-C100	DI1	160	INT0	018	PC	128	(10)	0
LSL-C1001	DI1	161	INT1	018	PC			0
LSL-SP-1	DI1	162	INT2	018	PC			0
YS-P-C100	DI1	163	INT3	018	PC	129		0
YS-RC3-2	DI1	164	INT4	018	PC			0
II-A-C100	AI1	225	AI0	018	PC		(15)	0
WFIC-C100	AI1	226	AI1	018	PC			0
LI-C100-G	AI	227	AI2	018	PC			0
LI-C100-1	AI1	228	AI3	018	PC	135		0
LI-C100-2	AI1	229	AI4	018	PC			0
TI-C100-1	AI4	230	AI5	018	PC			0
II-P-C100	AI1	231	AI6	018	PC	136		0
FIC-C1005	AI2	232	AI7	018	PC			0
FQI-C100	AI	233	AI8	018	PC			0
PI-FC1-3	AI1	234	AI9	018	PC			0
PDI-FC1-1	AI	235	AI10	018	PC			0
PI-FC1-4	AI1	236	AI11	018	PC			0
15 - - - I X - - -								
IX-ALARM	DEV	130	-	044	IX			0
HV-RC3-3	DEV	131	-	044	IX		(10)	0
RC3-SAMP	DEV	132	-	044	IX			0
RC3-PIG	DEV	133	-	044	IX			0
HV-2-34	DEV	134	-	044	IX			0
HV-2-37	DEV	135	-	044	IX			0
ZS-IDX-2	DI1	170	INT0	044	IX	130	(20)	0
PSH-ILF-1	DI1	171	INT1	044	IX	131		0
YS-RC3-6	DI1	172	INT2	044	IX			0
RXS-RC3-1	DI1	173	INT3	044	IX			0
RSH-RC3-1	DI1	174	INT4	044	IX			0
FSL-RC3-1	DI1	175	INT5	044	IX			0
ZS-RC3-4	DI1	176	INT6	044	IX	132		0
ZS-RC3-5	DI1	177	INT7	044	IX	133		0
ZS-IDX1NM	DI1	178	-	044	IX			0
ZS-IDX-1B	DI1	179	-	044	IX			0
ZS-IDX-3B	DI1	180	-	044	IX			0
ZS-IDX3NM	DI1	181	-	044	IX			0
ZS-RC3-3D	DI1	182	-	044	IX			0
ZS-RC33NM	DI1	183	-	044	IX			0
PI-IDX-13	AI1	240	AI0	044	IX		(25)	0
PI-IDX-14	AI1	241	AI1	044	IX			0
PI-IDX-15	AI1	242	AI2	044	IX			0
PI-IDX-16	AI1	243	AI3	044	IX			0
PDI-IDX-2	AI	244	AI4	044	IX			0
PDI-IDX-4	AI	245	AI5	044	IX			0

MCS EPN	SIG TYPE	IPN	SDB INT	FIRST GRAPH	ACAD FILE NAME	WHITE NUMB.	NUMBER PER SEQ UNIT	PCM
PDI-IDX-3	AI	246	AI6	044	IX			0
PI-IDX-12	AI1	247	AI7	044	IX			0
PI-IDX-17	AI1	248	AI8	044	IX			0
PDI-IDX-5	AI	249	AI9	044	IX			0
PDI-IDX-1	AI	250	AI10	044	IX			0
PI-IDX-18	AI1	251	AI11	044	IX			0
PI-ILS-5	AI1	252	AI12	044	IX			0
PI-ILS-6	AI1	253	AI13	044	IX			0
FI-RC3-1	AI2	254	AI14	044	IX			0
PDI-ILS-1	AI	255	AI15	044	IX			0
FQIC-RC3	AI	256	AI16	044	IX			0
FQI-RC3NM	AI	257	AI17	044	IX			0
FQI-RC3-D	AI	258	AI18	044	IX			0
KY-RC3-1	AI	259	AI19	044	IX			0
RI-RC3-1	AI3	260	AI20	044	IX			0
16 - - - S C - - -								
SC-ALARM	DEV	140	-	014	SC		(10)	0
HV-RC1-3	DEV	141	-	014	SC			0
RC1-SAMP	DEV	142	-	014	SC			0
RC1-PIG	DEV	143	-	014	SC			0
HV-EA1-2	DEV	144	-	014	SC			0
RSH-EA1-1	DI1	190	INT0	014	SC		(20)	0
LSL-RC1-1	DI1	191	INT1	014	SC			0
LSL-C1031	DI1	192	INT2	014	SC			0
YS-P-RC1	DI1	193	INT3	014	SC			0
YS-RC1-6	DI1	194	INT4	014	SC			0
FSL-RC1-1	DI1	195	INT5	014	SC			0
RSH-RC1-1	DI1	196	INT6	014	SC			0
RXS-RC1-1	DI1	197	INT7	014	SC			0
LSL-C1032	DI1	198	INT8	014	SC			0
ZS-EA12NM	DI1	199	-	014	SC			0
ZS-EA1-2D	DI1	200	-	014	SC			0
ZS-RC13NM	DI1	201	-	014	SC			0
ZS-RC1-3D	DI1	202	-	014	SC			0
FI-EA1-3	AI1	265	AI0	014	SC		(15)	0
FQI-EA1-D	AI	266	AI1	014	SC			0
FQI-EA1NM	AI	267	AI2	014	SC			0
RI-EA1-1	AI3	268	AI3	014	SC			0
KY-RC1-1	AI	269	AI4	014	SC			0
RI-RC1-1	AI3	270	AI5	014	SC			0
FI-RC1-1	AI1	271	AI6	014	SC			0
FQI-RC1-D	AI	272	AI7	014	SC			0
FQI-RC1NM	AI	273	AI8	014	SC			0
FQIC-RC1	AI	274	AI9	014	SC			0
CI-EA1-SE	AI	275	AI10	014	SC			0

MCS EPN	SIG TYPE	IPN	SDB INT	FIRST GRAPH	ACAD FILE NAME	WHITE NUMB.	NUMBER PER SEQ UNIT	PCM
17 - - - B A S I N S - - -								
BASIN-ALM	DEV	150	-	022	SCBASIN		(10)	0
P-207A-S	DEV	151	INT10	022	SCBASIN			0
207-A-P3	DEV	152	-	022	SCBASIN			0
HV-SC-1A	DEV	153	-	022	SCBASIN			0
HV-SC-2A	DEV	154	-	022	SCBASIN			0
HV-SC-3A	DEV	155	-	022	SCBASIN			0
HV-SC-1B	DEV	156	-	022	SCBASIN			0
HV-SC-2B	DEV	157	-	022	SCBASIN			0
HV-SC-3B	DEV	158	-	022	SCBASIN			0
YS-SCP3IL	DI1	210	INT0	022	SCBASIN		(20)	0
LSH-P-1	DI1	211	INT1	022	SCBASIN			0
YS-P1-1	DI1	212	INT2	022	SCBASIN			0
LSHH-SC-1	DI1	213	INT3	022	SCBASIN			0
LSH-SC-1	DI1	214	INT4	022	SCBASIN			0
LSL-SC-1	DI1	215	INT5	022	SCBASIN	138		0
LSH-SC-2	DI1	216	INT6	022	SCBASIN			0
LSL-SC-2	DI1	217	INT7	022	SCBASIN	139		0
LSH-SC-3	DI1	218	INT8	022	SCBASIN			0
LSL-SC-3	DI1	219	INT9	022	SCBASIN	140		0
YS-SC-P3	DI1	220	INT10	022	SCBASIN			0
ZS-SC-1DC	DI1	221	-	022	SCBASIN			0
ZS-SC-1FC	DI1	222	-	022	SCBASIN			0
ZS-SC-2DC	DI1	223	-	022	SCBASIN			0
ZS-SC-2FC	DI1	224	-	022	SCBASIN			0
ZS-SC-3DC	DI1	225	-	022	SCBASIN			0
ZS-SC-3FC	DI1	226	-	022	SCBASIN			0
II-SUMPP1	AI1	280	AI0	022	SCBASIN		(10)	0
II-SC-P3	AI1	281	AI1	022	SCBASIN			0
18 - - - U T I L - - -								
UTILITIES	DEV	160	-	036	UTIL1		(10)	0
PSLCPE1/2	DI1	230	INT0	021	UTIL		(20)	0
PSL-IA-2	DI1	231	INT1	021	UTIL			0
YS-CPE1-1	DI1	232	INT2	021	UTIL	143		0
YS-CPE2-1	DI1	233	INT3	021	UTIL	144		0
JXS-PWR-1	DI1	234	INT4	021	UTIL			0
YS-DIEGEN	DI1	235	INT5	021	UTIL			0
YS-GEN-1	DI1	236	INT6	021	UTIL			0
ZS-PWR-1	DI1	237	INT7	021	UTIL			0
JXS-PWR-3	DI1	238	INT8	021	UTIL			0
YSL-UPS-1	DI1	239	INT9	021	UTIL			0
ZS-UPS-1	DI1	240	INT10	021	UTIL			0

MCS EPN	SIG TYPE	IPN	SDB INT	FIRST GRAPH	ACAD FILE NAME	WHITE NUMB.	NUMBER PER SEQ UNIT	PCM
PI-IA-1	AI1	290	AI0	021	UTIL	145,146	(20)	0
PI-PA-1	AI1	291	AI1	021	UTIL	148		0
TI-AMBS-1	AI4	292	AI2	021	UTIL	149-152		0
TI-AMBS-2	AI5	293	AI3	021	UTIL	153-156		0
19 - - - R O O M S - - -								
ROOM-ALM	DEV	170	-	023	242A		(10) ROOMS	0
SIMDIN1	DI	250	-	-	-		(10) ROOMS	0
BUILDING	DEV	75	-	038	BLDNG1		(5)	1
BLDG-EXH	DEV	76	-	-	-		BLDNG	1
RSH-ARAMU	DI1	250	INT0	023	242A		(30)	1
RSH-ASAMU	DI1	251	INT1	023	242A		BLDNG	1
RSH-K1-12	DI1	252	INT2	023	242A			1
RSH-K1-11	DI1	253	INT3	023	242A			1
QXS-K1	DI1	254	INT4	023	242A			1
PDSH-K161	DI1	255	INT5	023	242A			1
PDSL-K161	DI1	256	INT6	023	242A			1
PDSH-K162	DI1	257	INT7	023	242A			1
PDSL-K162	DI1	258	INT8	023	242A			1
PDSL-K163	DI1	259	INT9	023	242A			1
PDSL-K164	DI1	260	INT10	023	242A			1
YS-K1-1	DI1	261	INT11	023	242A			1
RSH-ASSUR	DI1	262	INT12	023	242A			1
RSH-ASCON	DI1	263	INT13	023	242A			1
RSHARHVAC	DI1	264	INT14	023	242A			1
RSH-ARC4	DI1	265	INT15	023	242A			1
RSH-ARCB	DI1	266	INT16	023	242A			1
RSHAREVAP	DI1	267	INT17	023	242A			1
RSH-ARHES	DI1	268	INT18	023	242A			1
RXS-ASCON	DI1	269	INT19	023	242A			1
RXS-ASAMU	DI1	270	INT20	023	242A			1
RXS-ASSUR	DI1	271	INT21	023	242A			1
RXSAREVAP	DI1	272	INT22	023	242A			1
RXS-ARAMU	DI1	273	INT23	023	242A			1
RXSARHVAC	DI1	274	INT24	023	242A			1
RXS-ARHES	DI1	275	INT25	023	242A			1
RXS-ARC4	DI1	276	INT26	023	242A			1
RXS-ARCB	DI1	277	INT27	023	242A			1
RI-AR-AMU	AI2	80	AI0	023	242A		(20)	1
RI-ASAMU	AI2	81	AI1	023	242A		BLDNG	1
RI-ASSUR	AI2	82	AI2	023	242A			1
RI-ASCON	AI2	83	AI3	023	242A			1
RI-ARHVAC	AI2	84	AI4	023	242A			1
RI-AR-C4	AI2	85	AI5	023	242A			1

MCS EPN	SIG TYPE	IPN	SDB INT	FIRST GRAPH	ACAD FILE NAME	WHITE NUMB.	NUMBER PER SEQ UNIT	PCM
RI-AR-CB	AI2	86	AI6	023	242A			1
TI-K1-11	AI7	87	AI7	023	242A			1
TI-K1-4	AI2	88	AI8	023	242A			1
TI-K1-12A	AI2	89	AI9	023	242A			1
TI-K1-12B	AI2	90	AI10	023	242A			1
TI-K1-2	AI2	91	AI11	023	242A			1
RI-AREVAP	AI2	92	AI12	023	242A			0
RI-ARHES	AI2	93	AI13	023	242A			0
RI-K1-1-1	AI3	300	AI14	023	242A		(10)	0
RI-K1-1-2	AI3	301	AI15	023	242A		ROOMS	0
FI-K1-1	AI1	302	AI16	023	242A			0
20 - - - A M U - - -								
AMU-ALARM	DEV	180	-	019	AMU		(10)	0
P-E-101	DEV	181	-	019	AMU			0
A-E-101	DEV	182	-	019	AMU			0
A-E-102	DEV	183	-	019	AMU			0
P-E-102	DEV	184	-	019	AMU			0
P-E-104	DEV	185	-	019	AMU			0
A-E-104	DEV	186	-	019	AMU			0
YS-A-E101	DI1	260	INT0	019	AMU	157	(10)	0
YS-P-E101	DI1	261	INT1	019	AMU	158		0
YS-A-E102	DI1	262	INT2	019	AMU	159		0
YS-P-E102	DI1	263	INT3	019	AMU	160		0
YS-A-E104	DI1	264	INT4	019	AMU	161		0
YS-P-E104	DI1	265	INT5	019	AMU	162		0
TI-E101-2	AI5	310	AI0	019	AMU	163	(20)	0
WFI-E101	AI1	311	AI1	019	AMU	164,165		0
DI-E101-2	AI1	312	AI2	019	AMU			0
LI-E101-1	AI	313	AI3	019	AMU			0
FI-E102-1	AI2	314	AI4	019	AMU			0
WFI-E102	AI1	315	AI5	019	AMU	166,167		0
FQI-E102	AI	316	AI6	019	AMU			0
TI-E104-2	AI5	317	AI7	019	AMU	168		0
WFI-E104	AI1	318	AI8	019	AMU	169,170		0
DI-E104-2	AI1	319	AI9	019	AMU			0
LI-E104-1	AI	320	AI10	019	AMU			0
FQI-E104	AI	321	AI11	019	AMU			0
FI-E104-2	AI2	322	AI12	019	AMU			0
FIC-E102	AI	323	AI13	019	AMU			0

MCS EPN	SIG TYPE	IPN	SDB INT	FIRST GRAPH	ACAD FILE NAME	WHITE NUMB.	NUMBER PER SEQ UNIT	PCM
21 - - - L E R F - - -								
LERF-ALM	DEV	190	-	024	LERF3		(10)	0
LERFILK	DEV	191	-	024	LERF3			0
YS-P-43	DI1	270	INT0	024	LERF3		(20)	0
LDS-A1	DI1	271	INT1	024	LERF3			0
YS-P-44	DI1	272	INT2	024	LERF3			0
ZS-105PS1	DI1	273	INT3	024	LERF3	171		0
ZS-105PS2	DI1	274	INT4	024	LERF3	172		0
YS-P-42	DI1	275	INT5	024	LERF3			0
LDS-BSN-42	DI1	276	INT6	024	LERF3			0
LDS-BSN-43	DI1	277	INT7	024	LERF3			0
LDS-BSN-44	DI1	278	INT8	024	LERF3			0
LXS-LCH-42	DI1	279	INT9	024	LERF3			0
LXS-BSN-42	DI1	280	INT10	024	LERF3			0
LXS-LCH-43	DI1	281	INT11	024	LERF3			0
LXS-BSN-43	DI1	282	INT12	024	LERF3			0
LXS-LCH-44	DI1	283	INT13	024	LERF3			0
LXS-BSN-44	DI1	284	INT14	024	LERF3			0
YS-PLC-1	DI1	285	INT15	024	LERF3			0
LDI-A1	AI2	330	AI0	024	LERF3		(20)	0
LI-LCH42	AI1	331	AI1	024	LERF3			0
LI-BSN42	AI1	332	AI2	024	LERF3			0
LI-LCH43	AI1	333	AI3	024	LERF3			0
LI-BSN43	AI1	334	AI4	024	LERF3			0
LI-LCH44	AI1	335	AI5	024	LERF3			0
LI-BSN44	AI1	336	AI6	024	LERF3			0
22 - - - E V A P - - -								
PAW102ALM	DEV	15	-	040	OVER1		(80)	0
YS-AW102	DI1	50	-	040	OVER1		(110)	0
II-AW-102	AI1	30	-	040	OVER1		(155)	0
23 - - - V A P O R - - -								
EC1-ALARM	DEV	80	-	041	OVER2		(30)	0
RSH-CA1-1	DI1	110	-	041	OVER2		(60)	0
TI-CA1-9	AI5	155	-	041	OVER2		(90)	0
24 - - - C O N D - - -								
PC-ALARM	DEV	120	-	042	OVER3		(85)	0
YS-A-C100	DI1	160	-	042	OVER3		(100)	0
II-A-C100	AI1	225	-	042	OVER3		(135)	0

MCS EPN	SIG TYPE	IPN	SDB INT	FIRST GRAPH	ACAD FILE NAME	WHITE NUMB.	NUMBER PER SEQ UNIT	PCM
25 - - - B L A N K 0 - - -							(0)	
** - - - M I S C E L L A N E O U S - - -								
PB2GLOBAL	DEV	60	-	-	-			0
AUDALARM0	DEV	171	-	-	-			0
CASS0AL1A	DEV	200	-	090	CASS			0
CASS0AL2A	DEV	202	-	090	CASS			0
CASS0AL2B	DEV	203	-	090	CASS			0
GBIT0-0/1	DEV	204	-	090	CASS			0
HS-AB1ACK	DEV	208	-	-	-			0
0 - - - W H O L E 1 - - -								
P-AN-101	DEV	0					(90)	1
YS-AN-101	DI1	0					(280)	1
II-AN1011	AI1	0					(130)	1
1 - - - A N F A R M - - -								
P-AN-101	DEV	0	-	270	ANFRM		(15)	1
P-AN-102	DEV	1	-	270	ANFRM			1
P-AN-103	DEV	2	-	270	ANFRM			1
P-AN-104	DEV	3	-	270	ANFRM			1
P-AN-105	DEV	4	-	270	ANFRM			1
P-AN-106	DEV	5	-	270	ANFRM			1
P-AN-107	DEV	6	-	270	ANFRM			1
HS-AN-SIL	DEV	7	-	-	-			1
ANFARM	DEV	8	-	270	ANFRM			1
AN-SERPIT	DEV	9	-	-	-			1
AN-VLVPIT	DEV	10	-	272	ANVLV			1
AN-HVAC	DEV	11	-	273	ANHVC			1
AN-271	DEV	12	-	274	AN271			1
HS-ARM-AN	DEV	13	-	274	AN271			1
YS-AN-101	DI1	0	INT0	270	ANFRM	001	(60)	1
RSH-AN101	DI1	1	INT1	270	ANFRM			1
RSH-AN01C	DI1	2	INT2	270	ANFRM			1
YS-AN-102	DI1	3	INT3	270	ANFRM	002		1
RSH-AN102	DI1	4	INT4	270	ANFRM			1
RSH-AN02C	DI1	5	INT5	270	ANFRM			1
YS-AN-103	DI1	6	INT6	270	ANFRM	003		1
RSH-AN103	DI1	7	INT7	270	ANFRM			1
RSH-AN03C	DI1	8	INT8	270	ANFRM			1
YS-AN-104	DI1	9	INT9	270	ANFRM	004		1
RSH-AN104	DI1	10	INT10	270	ANFRM			1
RSH-AN04C	DI1	11	INT11	270	ANFRM			1

MCS EPN	SIG TYPE	IPN	SDB INT	FIRST GRAPH	ACAD FILE NAME	WHITE NUMB.	NUMBER PER SEQ UNIT	PCM
YS-AN-105	DI1	12	INT12	270	ANFRM	005		1
RSH-AN105	DI1	13	INT13	270	ANFRM			1
RSH-AN05C	DI1	14	INT14	270	ANFRM			1
YS-AN-106	DI1	15	INT15	270	ANFRM	006		1
RSH-AN106	DI1	16	INT16	270	ANFRM			1
RSH-AN06C	DI1	17	INT17	270	ANFRM			1
YS-AN-107	DI1	18	INT18	270	ANFRM	007		1
RSH-AN107	DI1	19	INT19	270	ANFRM			1
RSH-AN07C	DI1	20	INT20	270	ANFRM			1
LDS-SP-AN	DI1	21	INT21	272	ANVLV			1
RSH-SP-AN	DI1	22	INT22	272	ANVLV			1
LDS-AN-PP	DI1	23	INT23	272	ANVLV			1
LDS-ANSL1	DI1	24	INT24	272	ANVLV			1
LDS-ANSN2	DI1	25	INT25	272	ANVLV			1
LDS-COBAN	DI1	26	INT26	272	ANVLV			1
PSH-ANSN1	DI1	27	INT27	272	ANVLV			1
LDS-DR368	DI1	28	INT28	272	ANVLV			1
RSH-AN-K1	DI1	29	INT29	273	ANHVC			1
PSH-ANK11	DI1	30	INT30	273	ANHVC			1
PSH-ANK12	DI1	31	INT31	273	ANHVC			1
RXS-AN-K1	DI1	32	INT32	273	ANHVC			1
RSH-AN-K2	DI1	33	INT33	273	ANHVC			1
PSH-ANK21	DI1	34	INT34	273	ANHVC			1
PSH-ANK22	DI1	35	INT35	273	ANHVC			1
RXS-AN-K2	DI1	36	INT36	273	ANHVC			1
JXS-AN271	DI1	37	INT37	274	AN271			1
RSHAN2711	DI1	38	INT38	274	AN271			1
RSHAN2712	DI1	39	INT39	274	AN271			1
RSH-AN-12	DI1	40	INT40	274	AN271			1
QS-AN-271	DI1	41	INT41	274	AN271			1
TXS-AN-2	DI1	42	INT42	274	AN271	008		1
YS-AN-ARM	DI1	43	INT43	274	AN271			1
YS-AN-MSD	DI1	44	INT44	274	AN271			1
II-AN1011	AI1	0	AI0	270	ANFRM		(20)	1
II-AN1021	AI1	1	AI1	270	ANFRM			1
II-AN1031	AI1	2	AI2	270	ANFRM			1
II-AN1041	AI1	3	AI3	270	ANFRM			1
II-AN1051	AI1	4	AI4	270	ANFRM			1
II-AN1061	AI1	5	AI5	270	ANFRM			1
II-AN1071	AI1	6	AI6	270	ANFRM			1
FQI-AN-1	DI3	7	AI7	272	ANVLV			1
KY-ARM-AN	AI	8	AI8	274	AN271			1
SIMAI-1	AI	9	-	-	-			1

MCS EPN	SIG TYPE	IPN	SDB INT	FIRST GRAPH	ACAD FILE NAME	WHITE NUMB.	NUMBER PER SEQ UNIT	PCM
2 - - - A P F A R M - - -								
P-AP-101	DEV	20	-	290	APFRM	060	(20)	1
P-AP-102	DEV	21	-	290	APFRM	061		1
P-AP-103	DEV	22	-	290	APFRM	062		1
P-AP-104	DEV	23	-	290	APFRM	063		1
P-AP-105	DEV	24	-	290	APFRM	064		1
P-AP-106	DEV	25	-	290	APFRM	065		1
P-AP-107	DEV	26	-	290	APFRM	066		1
P-AP-108	DEV	27	-	290	APFRM	067		1
P-AP-102M	DEV	28	-	291	AP102	068		1
P-AP-102R	DEV	29	-	291	AP102	069		1
EXH-AP-K1	DEV	30	-	-	-			1
EXH-AP-K2	DEV	31	-	-	-			1
HS-AP-SIL	DEV	32	-	-	-			1
APFARM	DEV	33	-	290	APFRM			1
AP-SERPIT	DEV	34	-	-	-			1
AP-VLVPIT	DEV	35	-	292	APVLV			1
AP-HVAC	DEV	36	-	294	APHVC			1
AP-271	DEV	37	-	295	AP271			1
AP1023	DEV	38	-	291	AP102			1
HS-ARM-AP	DEV	39	-	295	AP271			1
YS-AP-101	DI2	60	INT0	290	APFRM	009	(90)	1
RSH-101AP	DI2	61	INT1	290	APFRM			1
RXS-101AP	DI2	62	INT2	290	APFRM			1
YS-AP-102	DI2	63	INT3	291	AP102	010		1
RSH-102AP	DI2	64	INT4	290	APFRM			1
RXS-102AP	DI2	65	INT5	290	APFRM			1
YS-AP-103	DI2	66	INT6	290	APFRM	011		1
RSH-103AP	DI2	67	INT7	290	APFRM			1
RXS-103AP	DI2	68	INT8	290	APFRM			1
RSH-AP03C	DI2	69	INT9	290	APFRM			1
RXS-AP03C	DI2	70	INT10	290	APFRM			1
LDS-AP03D	DI2	71	INT11	290	APFRM	012		1
YS-AP-104	DI2	72	INT12	290	APFRM	013		1
RSH-104AP	DI2	73	INT13	290	APFRM			1
RXS-104AP	DI2	74	INT14	290	APFRM			1
YS-AP-105	DI2	75	INT15	290	APFRM	014		1
RSH-105AP	DI2	76	INT16	290	APFRM			1
RXS-105AP	DI2	77	INT17	290	APFRM			1
RSH-AP05C	DI2	78	INT18	290	APFRM			1
RXS-AP05C	DI2	79	INT19	290	APFRM			1
YS-AP-106	DI2	80	INT20	290	APFRM	015		1
RSH-106AP	DI2	81	INT21	290	APFRM			1
RXS-106AP	DI2	82	INT22	290	APFRM			1
YS-AP-107	DI2	83	INT23	290	APFRM	016		1

MCS EPN	SIG TYPE	IPN	SDB INT	FIRST GRAPH	ACAD FILE NAME	WHITE NUMB.	NUMBER PER SEQ UNIT	PCM
RSH-107AP	DI2	84	INT24	290	APFRM			1
RXS-107AP	DI2	85	INT25	290	APFRM			1
YS-AP-108	DI2	86	INT26	290	APFRM	017		1
RSH-108AP	DI2	87	INT27	290	APFRM			1
RXS-108AP	DI2	88	INT28	290	APFRM			1
LDSAP02DC	DI2	89	INT29	291	AP102			1
YS-AP102M	DI2	90	INT30	291	AP102	018		1
PSHAP102M	DI2	91	INT31	291	AP102	019		1
PSLAP102M	DI2	92	INT32	291	AP102	020		1
LDSAP102M	DI2	93	INT33	291	AP102			1
YS-AP102R	DI2	94	INT34	291	AP102	021		1
WSHAP102M	DI2	95	INT35	291	AP102	022		1
TSHAP102M	DI2	96	INT36	291	AP102	023		1
VSHAP102M	DI2	97	INT37	291	AP102	024		1
YS-AP-MSD	DI1	98	INT38	295	AP271			1
YS-AP-ARM	DI1	99	INT39	295	AP271			1
RSH-AP-2	DI2	100	INT40	295	AP271			1
RXS-AP-2	DI2	101	INT41	295	AP271			1
QS-AP-1	DI2	102	INT42	295	AP271			1
TXS-AP-2	DI2	103	INT43	295	AP271	025		1
LDS-AP-A	DI2	104	INT44	295	AP271			1
LDS-AP-B	DI2	105	INT45	295	AP271			1
LDS-AP-C	DI2	106	INT46	295	AP271			1
JXS-AP-1	DI2	107	INT47	295	AP271			1
FSLAP2711	DI2	108	INT48	295	AP271	026		1
RSH-AP271	DI2	109	INT49	295	AP271			1
RXS-AP271	DI2	110	INT50	295	AP271			1
PSH-APFP1	DI2	111	INT51	292	APVLV	027		1
ZS-AP-FP1	DI2	112	INT52	292	APVLV	028		1
LDS-AP-FP	DI2	113	INT53	292	APVLV			1
YS-AP-K1	DI2	114	INT54	294	APHVC			1
FSL-APK13	DI2	115	INT55	294	APHVC	029		1
RSH-APK11	DI2	116	INT56	294	APHVC			1
RXS-APK11	DI2	117	ING57	294	APHVC			1
FSL-APK12	DI2	118	INT58	294	APHVC	030		1
FSL-APK11	DI2	119	INT59	294	APHVC	031		1
LDS-AP-K1	DI2	120	INT60	294	APHVC			1
YS-AP-K2	DI2	121	INT61	294	APHVC			1
FSL-APK23	DI2	122	INT62	294	APHVC	032		1
RSH-APK21	DI2	123	INT63	294	APHVC			1
RXS-APK21	DI2	124	INT64	294	APHVC			1
FSL-APK22	DI2	125	INT65	294	APHVC	033		1
FSL-APK21	DI2	126	INT66	294	APHVC	034		1
LDS-AP-SP	DI2	127	INT67	292	APVLV			1
RSH-AP-SP	DI2	128	INT68	292	APVLV			1
RXS-AP-SP	DI2	129	INT69	292	APVLV			1

MCS EPN	SIG TYPE	IPN	SDB INT	FIRST GRAPH	ACAD FILE NAME	WHITE NUMB.	NUMBER PER SEQ UNIT	PCM
LDS-AP-SL	DI2	130	INT70	292	APVLV			1
LDS-AP-SN	DI2	131	INT71	292	APVLV			1
LDSAP621A	DI2	132	INT72	291	AP102			1
LDSAP6502	DI2	133	INT73	291	AP102			1
PSH-AP-2	DI2	134	INT74	295	AP271			1
PSL-AP-2	DI2	135	INT75	295	AP271			1
LDS-AP-VP	DI2	136	INT76	292	APVLV			1
FSH-DR712	DI2	137	INT77	292	APVLV	035		1
FXS-DR712	DI2	138	INT78	292	APVLV	036		1
PSH-APFL1	DI2	139	INT79	292	APVLV	037		1
ZS-AP-DR1	DI2	140	INT80	292	APVLV	038		1
ZS-AP-FL1	DI2	141	INT81	292	APVLV	039		1
YS-1023BY	DI2	142	INT82	291	AP102			1
II-AP1011	AI1	20	AI0	290	APFRM		(20)	1
II-AP1021	AI1	21	AI1	291	AP102			1
II-AP1031	AI1	22	AI2	290	APFRM			1
II-AP1041	AI1	23	AI3	290	APFRM			1
II-AP1051	AI1	24	AI4	290	APFRM			1
II-AP1061	AI1	25	AI5	290	APFRM			1
II-AP1071	AI1	26	AI6	290	APFRM			1
II-AP1081	AI1	27	AI7	290	APFRM			1
II-AP102P	AI1	28	AI9	291	AP102			1
II-AP102R	AI1	29	AI10	291	AP102			1
WI-AP102M	AI1	30	AI11	291	AP102			1
PI-AP102M	AI1	31	AI12	291	AP102			1
TI-AP102M	AI1	32	AI13	291	AP102			1
FQI-APSP1	AI	33	AI8	292	APVLV			1
KY-ARM-AP	AI	34	-	295	AP271			1
3 - - - A W F A R M - - -								
P-AW-101	DEV	40	-	300	AWFRM		(20)	1
P-AW-103	DEV	41	-	300	AWFRM			1
P-AW-104	DEV	42	-	300	AWFRM			1
P-AW-105	DEV	43	-	300	AWFRM			1
P-AW-106	DEV	44	-	300	AWFRM			1
HS-AW-SIL	DEV	45	-	-	-			1
AWFARM	DEV	46	-	300	AWFRM			1
AW-SERPIT	DEV	47	-	-	-			1
AW-VLVPIT	DEV	48	-	303	AWVLV			1

MCS EPN	SIG TYPE	IPN	SDB INT	FIRST GRAPH	ACAD FILE NAME	WHITE NUMB.	NUMBER PER SEQ UNIT	PCM
AW-HVAC	DEV	49	-	304	AWHVC			1
AW-271	DEV	50	-	302	AW271			1
AW-102	DEV	51	-	-	-			1
HS-ARM-AW	DEV	52	-	302	AW271			1
YS-AW-101	DI1	150	INT0	300	AWFRM	040	(60)	1
RSH-AW101	DI1	151	INT1	300	AWFRM			1
RSH-AW01C	DI1	152	INT2	300	AWFRM			1
RSH-AW102	DI1	153	INT3	300	AWFRM			1
RSH-AW02C	DI1	154	INT4	300	AWFRM			1
YS-AW-103	DI1	155	INT5	300	AWFRM	042		1
RSH-AW103	DI1	156	INT6	300	AWFRM			1
RSH-AW03C	DI1	157	INT7	300	AWFRM			1
YS-AW-104	DI1	158	INT8	300	AWFRM	043		1
RSH-AW104	DI1	159	INT9	300	AWFRM			1
RSH-AW04C	DI1	160	INT10	300	AWFRM			1
YS-AW-105	DI1	161	INT11	300	AWFRM	044		1
RSH-AW105	DI1	162	INT12	300	AWFRM			1
RSH-AW05C	DI1	163	INT13	300	AWFRM			1
YS-AW-106	DI1	164	INT14	300	AWFRM	045		1
RSH-AW106	DI1	165	INT15	300	AWFRM			1
RSH-AW06C	DI1	166	INT16	300	AWFRM			1
LDS-TL021	DI1	167	INT17	303	AWVLV			1
LDSTL2223	DI1	168	INT18	303	AWVLV			1
LDS-AW-DR	DI1	169	INT19	303	AWVLV			1
LDS-AW-PP	DI1	170	INT20	303	AWVLV			1
LDS-AW-SL	DI1	171	INT21	303	AWVLV			1
LDS-AW-SN	DI1	172	INT22	303	AWVLV			1
LDS-SN269	DI1	173	INT23	303	AWVLV			1
LDS-SN270	DI1	174	INT24	303	AWVLV			1
LDS-COBAW	DI1	175	INT25	303	AWVLV			1
PSH-AWSN1	DI1	176	INT26	303	AWVLV			1
PSH-AWSN2	DI1	177	INT27	303	AWVLV			1
QS-AW-271	DI1	178	INT28	302	AW271			1
RSH-SP-AW	DI1	179	INT29	303	AWVLV			1
LDS-SP-AW	DI1	180	INT30	303	AWVLV			1
RSH-AW-K1	DI1	181	INT31	304	AWHVC			1
QXS-AWK15	DI1	186	INT36	304	AWHVC			1
RSH-AW-K2	DI1	187	INT37	304	AWHVC			1
QXSAWK1A2	DI1	188	INT38	304	AWHVC			1
RSH-AW-12	DI1	189	INT39	302	AW271			1
RSHAW2711	DI1	190	INT40	302	AW271			1
RSHAW2712	DI1	191	INT41	302	AW271			1
YS-AW-ARM	DI1	192	INT42	302	AW271			1
JXS-AW271	DI1	193	INT43	302	AW271			1

MCS EPN	SIG TYPE	IPN	SDB INT	FIRST GRAPH	ACAD FILE NAME	WHITE NUMB.	NUMBER PER SEQ UNIT	PCM
YS-AW-MSD	DI1	195	INT44	302	AW271			1
TXS-AW-2	DI1	196	INT45	302	AW271	046		1
II-AW1011	AI1	40	AI0	300	AWFRM		(20)	1
II-AW1031	AI1	41	AI1	300	AWFRM			1
II-AW1041	AI1	42	AI2	300	AWFRM			1
II-AW1051	AI1	43	AI3	300	AWFRM			1
II-AW1061	AI1	44	AI4	300	AWFRM			1
FQI-AW-1	DI3	45	AI5	303	AWVLV			1
KY-ARM-AW	AI	46	-	302	AW271			1
4 - - - X Y Z F R M - - -								
P-AY-101	DEV	60	INT30	322	AYFRM	076	(15)	1
P-AY-102	DEV	61	INT31	322	AYFRM	077		1
P-AZ-101	DEV	62	INT32	323	AZFRM	078		1
P-AZ-102	DEV	63	INT33	323	AZFRM	079		1
AFARM	DEV	64	-	-	-			1
AYFARM	DEV	65	-	-	-			1
AZFARM	DEV	66	-	-	-			1
AXYZ	DEV	67	-	-	-			1
XYZLEAK	DEV	68	-	-	-			1
HSAXYZSIL	DEV	69	-	-	-			1
HS-ARMAAX	DEV	70	-	-	-			1
LXS-AXPOT	DI1	229	INT19	320	AXYZFRM		(60)	1
LSH-AX-1	DI1	230	INT20	320	AXYZFRM	051		1
LSL-AX-1	DI1	231	INT21	320	AXYZFRM	052		1
LDSCOBAAX	DI1	232	INT22	320	AXYZFRM			1
YS-AAXARM	DI1	233	INT23	320	AXYZFRM			1
YS-AAXMSD	DI1	234	INT24	320	AXYZFRM			1
RSHAAX3/4	DI1	235	INT25	320	AXYZFRM			1
RSH-AXZY1	DI1	236	INT26	320	AXYZFRM			1
LDSPPCOB1	DI1	237	INT27	320	AXYZFRM			1
RXSAAX3/4	DI1	238	INT28	320	AXYZFRM			1
PSH-SL100	DI1	214	INT4	321	AFRM			1
PSH-SL101	DI1	215	INT5	321	AFRM			1
PSH-SL113	DI1	216	INT6	321	AFRM			1
PSH-SL114	DI1	217	INT7	321	AFRM			1
PSH200213	DI1	218	INT8	321	AFRM			1
PSH201214	DI1	219	INT9	321	AFRM			1
LDS-SL500	DI1	220	INT10	323	AZFRM			1
LDS-SL501	DI1	221	INT11	323	AZFRM			1

MCS EPN	SIG TYPE	IPN	SDB INT	FIRST GRAPH	ACAD FILE NAME	WHITE NUMB.	NUMBER PER SEQ UNIT	PCM
LDS-SL502	DI1	222	INT12	322	AYFRM			1
LDS-SL503	DI1	223	INT13	322	AYFRM			1
LDS-SL504	DI1	224	INT14	322	AYFRM			1
LDS-SL505	DI1	225	INT15	322	AYFRM			1
LDS-SN600	DI1	226	INT16	323	AZFRM			1
LDS-SN601	DI1	227	INT17	323	AZFRM			1
LDS-DR700	DI1	228	INT18	322	AYFRM			1
PSH-SN215	DI1	15	INT34	321	AFRM			0
PSH-SN216	DI1	16	INT35	321	AFRM			0
II-AY-101	AI1	60	AI0	322	AYFRM		(60)	1
II-AY-102	AI1	61	AI1	322	AYFRM			1
II-AZ-101	AI1	62	AI2	323	AZFRM			1
II-AZ-102	AI1	63	AI3	323	AZFRM			1
KY-ARMAAX	AI	67	-	320	AXYZFRM			1
5 - - - B U I L D I N G - - -	(SEE PCM0 UNIT #19, ROOMS)							
6 - - - B L A N K 1 - - -								
GBIT1-0/1	DEV	85	-	-	-		(5)	1
CASS1AL1A	DEV	86	-	090	CASS			1
CASS1AL2A	DEV	87	-	090	CASS			1
WHITE	DEV	88	-	-	-			1
** - - - M I S C E L L A N E O U S - - -								
SIMDEV1	DEV	14	-	-	-			1
HS-AB2ACK	DEV	15	-	-	-			1
CNDSR-RM	DEV	77	-	-	-			1
QS-204AR1	DI1	240	-	FP-81	-			1
QS-204AR2	DI1	241	-	FP-81	-			1
YS-241A1A	DI1	244	-	FP-81	-			1

APPENDIX B -SEQUENCE DATABASE (SDB)

PCMOASDB

```
PCM name for which to create the SDB?> = pcm0a
Number of sequence units for PCM?> = 26


Unit 0
      Enter unit 0 sequence unit name?> = whole0
      Enter unit 0 console enable index?> = 0
      Read/Write unit 0 from other sequence units[Y/N]?> = y
      Enter unit 0 UPF name?> = total0
      Enter number of DEV for unit 0?> = 217
            Enter EPN of first DEV?> = 244alarm
      Enter number of AIN for unit 0?> = 379
            Enter EPN of first AIN?> = ii-p244al
      Enter number of DIN for unit 0?> = 348
            Enter EPN of first DIN?> = ys244alil
      Enter number of DOT for unit 0?> = 0
      Enter number of DGR for unit 0?> = 5
            Enter EPN of first DGR?> = dgr0


Unit 1
      Enter unit 1 sequence unit name?> = a244
      Enter unit 1 console enable index?> = 0
      Read/Write unit 1 from other sequence units[Y/N]?> = y
      Enter unit 1 UPF name?> = short0
      Enter number of DEV for unit 1?> = 15
            Enter EPN of first DEV?> = 244alarm
      Enter number of AIN for unit 1?> = 30
            Enter EPN of first AIN?> = ii-p244al
      Enter number of DIN for unit 1?> = 50
            Enter EPN of first DIN?> = ys244alil
      Enter number of DOT for unit 1?> = 0
      Enter number of DGR for unit 1?> = 0


Unit 2
      Enter unit 2 sequence unit name?> = hvc244
      Enter unit 2 console enable index?> = 0
      Read/Write unit 2 from other sequence units[Y/N]?> = y
      Enter unit 2 UPF name?> = short0
      Enter number of DEV for unit 2?> = 1
            Enter EPN of first DEV?> = hvac244
      Enter number of AIN for unit 2?> = 5
            Enter EPN of first AIN?> = ri-244ann
      Enter number of DIN for unit 2?> = 5
            Enter EPN of first DIN?> = pdsl-ef-2
      Enter number of DOT for unit 2?> = 0
      Enter number of DGR for unit 2?> = 0
```

```
Unit 3
      Enter unit 3 sequence unit name?> = waste
      Enter unit 3 console enable index?> = 0
      Read/Write unit 3 from other sequence units[Y/N]?> = y
      Enter unit 3 UPF name?> = short0
      Enter number of DEV for unit 3?> = 1
            Enter EPN of first DEV?> = wstxfr
      Enter number of AIN for unit 3?> = 0
      Enter number of DIN for unit 3?> = 18
            Enter EPN of first DIN?> = lds-ar151
      Enter number of DOT for unit 3?> = 0
      Enter number of DGR for unit 3?> = 0


Unit 4
      Enter unit 4 sequence unit name?> = a350
      Enter unit 4 console enable index?> = 0
      Read/Write unit 4 from other sequence units[Y/N]?> = y
      Enter unit 4 UPF name?> = short0
      Enter number of DEV for unit 4?> = 3
            Enter EPN of first DEV?> = 350alarm
      Enter number of AIN for unit 4?> = 17
            Enter EPN of first AIN?> = ii-350-1
      Enter number of DIN for unit 4?> = 5
            Enter EPN of first DIN?> = lsh-350-2
      Enter number of DOT for unit 4?> = 0
      Enter number of DGR for unit 4?> = 0


Unit 5
      Enter unit 5 sequence unit name?> = feed
      Enter unit 5 console enable index?> = 0
      Read/Write unit 5 from other sequence units[Y/N]?> = y
      Enter unit 5 UPF name?> = prcess
      Enter number of DEV for unit 5?> = 10
            Enter EPN of first DEV?> = paw102alm
      Enter number of AIN for unit 5?> = 15
            Enter EPN of first AIN?> = ii-aw-102
      Enter number of DIN for unit 5?> = 10
            Enter EPN of first DIN?> = ys-aw102
      Enter number of DOT for unit 5?> = 0
      Enter number of DGR for unit 5?> = 0
```

```
Unit 6
      Enter unit 6 sequence unit name?> = cal
      Enter unit 6 console enable index?> = 0
      Read/Write unit 6 from other sequence units[Y/N]?> = y
      Enter unit 6 UPF name?> = prcess
      Enter number of DEV for unit 6?> = 10
            Enter EPN of first DEV?> = cal-alarm
      Enter number of AIN for unit 6?> = 35
            Enter EPN of first AIN?> = pi-cal-11
      Enter number of DIN for unit 6?> = 10
            Enter EPN of first DIN?> = psh-calll
      Enter number of DOT for unit 6?> = 0
      Enter number of DGR for unit 6?> = 0

Unit 7
      Enter unit 7 sequence unit name?> = pbl
      Enter unit 7 console enable index?> = 0
      Read/Write unit 7 from other sequence units[Y/N]?> = y
      Enter unit 7 UPF name?> = prcess
      Enter number of DEV for unit 7?> = 15
            Enter EPN of first DEV?> = pbl-alarm
      Enter number of AIN for unit 7?> = 20
            Enter EPN of first AIN?> = ky-pbl-l
      Enter number of DIN for unit 7?> = 15
            Enter EPN of first DIN?> = ys-pc-105
      Enter number of DOT for unit 7?> = 0
      Enter number of DGR for unit 7?> = 0

Unit 8
      Enter unit 8 sequence unit name?> = pb2
      Enter unit 8 console enable index?> = 0
      Read/Write unit 8 from other sequence units[Y/N]?> = y
      Enter unit 8 UPF name?> = prcess
      Enter number of DEV for unit 8?> = 15
            Enter EPN of first DEV?> = pb2-alarm
      Enter number of AIN for unit 8?> = 20
            Enter EPN of first AIN?> = hi-wvr-1
      Enter number of DIN for unit 8?> = 20
            Enter EPN of first DIN?> = ys-pb2-1
      Enter number of DOT for unit 8?> = 0
      Enter number of DGR for unit 8?> = 0
```

```
Unit 9
      Enter unit 9 sequence unit name?> = eal
      Enter unit 9 console enable index?> = 0
      Read/Write unit 9 from other sequence units[Y/N]?> = y
      Enter unit 9 UPF name?> = prcess
      Enter number of DEV for unit 9?> = 15
            Enter EPN of first DEV?> = eal-alarm
      Enter number of AIN for unit 9?> = 30
            Enter EPN of first AIN?> = ti-eal-7
      Enter number of DIN for unit 9?> = 5
            Enter EPN of first DIN?> = psl-eal-1
      Enter number of DOT for unit 9?> = 0
      Enter number of DGR for unit 9?> = 0


Unit 10
      Enter unit 10 sequence unit name?> = ecl
      Enter unit 10 console enable index?> = 0
      Read/Write unit 10 from other sequence units[Y/N]?> = y
      Enter unit 10 UPF name?> = prcess
      Enter number of DEV for unit 10?> = 10
            Enter EPN of first DEV?> = ecl-alarm
      Enter number of AIN for unit 10?> = 30
            Enter EPN of first AIN?> = ti-cal-9
      Enter number of DIN for unit 10?> = 10
            Enter EPN of first DIN?> = rsh-cal-1
      Enter number of DOT for unit 10?> = 0
      Enter number of DGR for unit 10?> = 0


Unit 11
      Enter unit 11 sequence unit name?> = rwater
      Enter unit 11 console enable index?> = 0
      Read/Write unit 11 from other sequence units[Y/N]?> = y
      Enter unit 11 UPF name?> = prcess
      Enter number of DEV for unit 11?> = 10
            Enter EPN of first DEV?> = rw-alarm
      Enter number of AIN for unit 11?> = 10
            Enter EPN of first AIN?> = pdi-rw-1
      Enter number of DIN for unit 11?> = 10
            Enter EPN of first DIN?> = lds-bfp-1
      Enter number of DOT for unit 11?> = 0
      Enter number of DGR for unit 11?> = 0
```

```
Unit 12
      Enter unit 12 sequence unit name?> = urw
      Enter unit 12 console enable index?> = 0
      Read/Write unit 12 from other sequence units[Y/N]?> = y
      Enter unit 12 UPF name?> = prcess
      Enter number of DEV for unit 12?> = 10
            Enter EPN of first DEV?> = urw-alarm
      Enter number of AIN for unit 12?> = 15
            Enter EPN of first AIN?> = fic-ec1-1
      Enter number of DIN for unit 12?> = 10
            Enter EPN of first DIN?> = fs1-rc2-1
      Enter number of DOT for unit 12?> = 0
      Enter number of DGR for unit 12?> = 0


Unit 13
      Enter unit 13 sequence unit name?> = vvent
      Enter unit 13 console enable index?> = 0
      Read/Write unit 13 from other sequence units[Y/N]?> = y
      Enter unit 13 UPF name?> = prcess
      Enter number of DEV for unit 13?> = 10
            Enter EPN of first DEV?> = vv-alarm
      Enter number of AIN for unit 13?> = 20
            Enter EPN of first AIN?> = pdi-fc5-2
      Enter number of DIN for unit 13?> = 20
            Enter EPN of first DIN?> = pdsh-fc51
      Enter number of DOT for unit 13?> = 0
      Enter number of DGR for unit 13?> = 0


Unit 14
      Enter unit 14 sequence unit name?> = pc
      Enter unit 14 console enable index?> = 0
      Read/Write unit 14 from other sequence units[Y/N]?> = y
      Enter unit 14 UPF name?> = prcess
      Enter number of DEV for unit 14?> = 10
            Enter EPN of first DEV?> = pc-alarm
      Enter number of AIN for unit 14?> = 15
            Enter EPN of first AIN?> = ii-a-c100
      Enter number of DIN for unit 14?> = 10
            Enter EPN of first DIN?> = ys-a-c100
      Enter number of DOT for unit 14?> = 0
      Enter number of DGR for unit 14?> = 0
```

```
Unit 15
      Enter unit 15 sequence unit name?> = ix
      Enter unit 15 console enable index?> = 0
      Read/Write unit 15 from other sequence units[Y/N]?> = y
      Enter unit 15 UPF name?> = prcess
      Enter number of DEV for unit 15?> = 10
            Enter EPN of first DEV?> = ix-alarm
      Enter number of AIN for unit 15?> = 25
            Enter EPN of first AIN?> = pi-idx-13
      Enter number of DIN for unit 15?> = 20
            Enter EPN of first DIN?> = zs-idx-2
      Enter number of DOT for unit 15?> = 0
      Enter number of DGR for unit 15?> = 0


Unit 16
      Enter unit 16 sequence unit name?> = sc
      Enter unit 16 console enable index?> = 0
      Read/Write unit 16 from other sequence units[Y/N]?> = y
      Enter unit 16 UPF name?> = prcess
      Enter number of DEV for unit 16?> = 10
            Enter EPN of first DEV?> = sc-alarm
      Enter number of AIN for unit 16?> = 15
            Enter EPN of first AIN?> = fi-eal-3
      Enter number of DIN for unit 16?> = 20
            Enter EPN of first DIN?> = rsh-eal-1
      Enter number of DOT for unit 16?> = 0
      Enter number of DGR for unit 16?> = 0


Unit 17
      Enter unit 17 sequence unit name?> = basins
      Enter unit 17 console enable index?> = 0
      Read/Write unit 17 from other sequence units[Y/N]?> = y
      Enter unit 17 UPF name?> = prcess
      Enter number of DEV for unit 17?> = 10
            Enter EPN of first DEV?> = basin-alm
      Enter number of AIN for unit 17?> = 10
            Enter EPN of first AIN?> = ii-sumppl
      Enter number of DIN for unit 17?> = 20
            Enter EPN of first DIN?> = ys-scp3il
      Enter number of DOT for unit 17?> = 0
      Enter number of DGR for unit 17?> = 0
```

```
Unit 18
      Enter unit 18 sequence unit name?> = util
      Enter unit 18 console enable index?> = 0
      Read/Write unit 18 from other sequence units[Y/N]?> = y
      Enter unit 18 UPF name?> = prcess
      Enter number of DEV for unit 18?> = 10
            Enter EPN of first DEV?> = utilities
      Enter number of AIN for unit 18?> = 20
            Enter EPN of first AIN?> = pi-ia-1
      Enter number of DIN for unit 18?> = 20
            Enter EPN of first DIN?> = pslcpe1/2
      Enter number of DOT for unit 18?> = 0
      Enter number of DGR for unit 18?> = 0


Unit 19
      Enter unit 19 sequence unit name?> = rooms
      Enter unit 19 console enable index?> = 0
      Read/Write unit 19 from other sequence units[Y/N]?> = y
      Enter unit 19 UPF name?> = prcess
      Enter number of DEV for unit 19?> = 10
            Enter EPN of first DEV?> = room-alm
      Enter number of AIN for unit 19?> = 10
            Enter EPN of first AIN?> = ri-k1-1-1
      Enter number of DIN for unit 19?> = 10
            Enter EPN of first DIN?> = simdin1
      Enter number of DOT for unit 19?> = 0
      Enter number of DGR for unit 19?> = 0


Unit 20
      Enter unit 20 sequence unit name?> = amu
      Enter unit 20 console enable index?> = 0
      Read/Write unit 20 from other sequence units[Y/N]?> = y
      Enter unit 20 UPF name?> = prcess
      Enter number of DEV for unit 20?> = 10
            Enter EPN of first DEV?> = amu-alarm
      Enter number of AIN for unit 20?> = 20
            Enter EPN of first AIN?> = ti-e101-2
      Enter number of DIN for unit 20?> = 10
            Enter EPN of first DIN?> = ys-a-e101
      Enter number of DOT for unit 20?> = 0
      Enter number of DGR for unit 20?> = 0
```

```
Unit 21
      Enter unit 21 sequence unit name?> = lerf
      Enter unit 21 console enable index?> = 0
      Read/Write unit 21 from other sequence units[Y/N]?> = y
      Enter unit 21 UPF name?> = prcess
      Enter number of DEV for unit 21?> = 10
            Enter EPN of first DEV?> = lerf-alm
      Enter number of AIN for unit 21?> = 20
            Enter EPN of first AIN?> = ldi-a1
      Enter number of DIN for unit 21?> = 20
            Enter EPN of first DIN?> = ys-p-43
      Enter number of DOT for unit 21?> = 0
      Enter number of DGR for unit 21?> = 0


Unit 22
      Enter unit 22 sequence unit name?> = evap
      Enter unit 22 console enable index?> = 0
      Read/Write unit 22 from other sequence units[Y/N]?> = y
      Enter unit 22 UPF name?> = ovrvew
      Enter number of DEV for unit 22?> = 80
            Enter EPN of first DEV?> = paw102alm
      Enter number of AIN for unit 22?> = 155
            Enter EPN of first AIN?> = ii-aw-102
      Enter number of DIN for unit 22?> = 110
            Enter EPN of first DIN?> = ys-aw102
      Enter number of DOT for unit 22?> = 0
      Enter number of DGR for unit 22?> = 0


Unit 23
      Enter unit 23 sequence unit name?> = vapor
      Enter unit 23 console enable index?> = 0
      Read/Write unit 23 from other sequence units[Y/N]?> = y
      Enter unit 23 UPF name?> = ovrvew
      Enter number of DEV for unit 23?> = 30
            Enter EPN of first DEV?> = ecl-alarm
      Enter number of AIN for unit 23?> = 90
            Enter EPN of first AIN?> = ti-cal-9
      Enter number of DIN for unit 23?> = 60
            Enter EPN of first DIN?> = rsh-cal-1
      Enter number of DOT for unit 23?> = 0
      Enter number of DGR for unit 23?> = 0
```

```
Unit 24
      Enter unit 24 sequence unit name?> = cond
      Enter unit 24 console enable index?> = 0
      Read/Write unit 24 from other sequence units[Y/N]?> = y
      Enter unit 24 UPF name?> = ovrvew
      Enter number of DEV for unit 24?> = 85
            Enter EPN of first DEV?> = pc-alarm
      Enter number of AIN for unit 24?> = 135
            Enter EPN of first AIN?> = ii-a-c100
      Enter number of DIN for unit 24?> = 100
            Enter EPN of first DIN?> = ys-a-c100
      Enter number of DOT for unit 24?> = 0
      Enter number of DGR for unit 24?> = 0


Unit 25
      Enter unit 25 sequence unit name?> = blank0
      Enter unit 25 console enable index?> = 0
      Read/Write unit 25 from other sequence units[Y/N]?> = y
      Enter unit 25 UPF name?> = blankk
      Enter number of DEV for unit 25?> = 0
      Enter number of AIN for unit 25?> = 0
      Enter number of DIN for unit 25?> = 0
      Enter number of DOT for unit 25?> = 0
      Enter number of DGR for unit 25?> = 0


Build PCM0A.SDB? [Y/N]> = y
Build Completed -- Database requires 3295 pages of memory
Total Errors = 0
```

PCM1ASDB

```
PCM name for which to create the SDB?> = pcm1a
Number of sequence units for PCM?> = 7


Unit 0
      Enter unit 0 sequence unit name?> = whole1
      Enter unit 0 console enable index?> = 0
      Read/Write unit 0 from other sequence units[Y/N]?> = y
      Enter unit 0 UPF name?> = total1
      Enter number of DEV for unit 0?> = 90
            Enter EPN of first DEV?> = p-an-101
      Enter number of AIN for unit 0?> = 130
            Enter EPN of first AIN?> = ii-an1011
      Enter number of DIN for unit 0?> = 280
            Enter EPN of first DIN?> = ys-an-101
      Enter number of DOT for unit 0?> = 0
      Enter number of DGR for unit 0?> = 0


Unit 1
      Enter unit 1 sequence unit name?> = anfarm
      Enter unit 1 console enable index?> = 0
      Read/Write unit 1 from other sequence units[Y/N]?> = y
      Enter unit 1 UPF name?> = tnkfrm
      Enter number of DEV for unit 1?> = 15
            Enter EPN of first DEV?> = p-an-101
      Enter number of AIN for unit 1?> = 20
            Enter EPN of first AIN?> = ii-an1011
      Enter number of DIN for unit 1?> = 60
            Enter EPN of first DIN?> = ys-an-101
      Enter number of DOT for unit 1?> = 0
      Enter number of DGR for unit 1?> = 0


Unit 2
      Enter unit 2 sequence unit name?> = apfarm
      Enter unit 2 console enable index?> = 0
      Read/Write unit 2 from other sequence units[Y/N]?> = y
      Enter unit 2 UPF name?> = tnkfrm
      Enter number of DEV for unit 2?> = 20
            Enter EPN of first DEV?> = p-ap-101
      Enter number of AIN for unit 2?> = 20
            Enter EPN of first AIN?> = ii-ap1011
      Enter number of DIN for unit 2?> = 90
            Enter EPN of first DIN?> = ys-ap-101
      Enter number of DOT for unit 2?> = 0
      Enter number of DGR for unit 2?> = 0
```

```
Unit 3
      Enter unit 3 sequence unit name?> = awfarm
      Enter unit 3 console enable index?> = 0
      Read/Write unit 3 from other sequence units[Y/N]?> = y
      Enter unit 3 UPF name?> = tnkfrm
      Enter number of DEV for unit 3?> = 20
            Enter EPN of first DEV?> = p-aw-101
      Enter number of AIN for unit 3?> = 20
            Enter EPN of first AIN?> = ii-awl011
      Enter number of DIN for unit 3?> = 60
            Enter EPN of first DIN?> = ys-aw-101
      Enter number of DOT for unit 3?> = 0
      Enter number of DGR for unit 3?> = 0


Unit 4
      Enter unit 4 sequence unit name?> = xyzfrm
      Enter unit 4 console enable index?> = 0
      Read/Write unit 4 from other sequence units[Y/N]?> = y
      Enter unit 4 UPF name?> = tnkfrm
      Enter number of DEV for unit 4?> = 15
            Enter EPN of first DEV?> = p-ay-101
      Enter number of AIN for unit 4?> = 60
            Enter EPN of first AIN?> = ii-ay-101
      Enter number of DIN for unit 4?> = 60
            Enter EPN of first DIN?> = psh-sl100
      Enter number of DOT for unit 4?> = 0
      Enter number of DGR for unit 4?> = 0


Unit 5
      Enter unit 5 sequence unit name?> = bldng
      Enter unit 5 console enable index?> = 0
      Read/Write unit 5 from other sequence units[Y/N]?> = y
      Enter unit 5 UPF name?> = tnkfrm
      Enter number of DEV for unit 5?> = 5
            Enter EPN of first DEV?> = building
      Enter number of AIN for unit 5?> = 20
            Enter EPN of first AIN?> = ri-ar-amu
      Enter number of DIN for unit 5?> = 30
            Enter EPN of first DIN?> = rsh-aramu
      Enter number of DOT for unit 5?> = 0
      Enter number of DGR for unit 5?> = 0
```

```
Unit 6
      Enter unit 6 sequence unit name?> = blank1
      Enter unit 6 console enable index?> = 0
      Read/Write unit 6 from other sequence units[Y/N]?> = y
      Enter unit 6 UPF name?> = blankk
      Enter number of DEV for unit 6?> = 5
            Enter EPN of first DEV?> = gbit1-0/1
      Enter number of AIN for unit 6?> = 0
      Enter number of DIN for unit 6?> = 0
      Enter number of DOT for unit 6?> = 0
      Enter number of DGR for unit 6?> = 0


Build PCM1A.SDB? [Y/N]> = y
Build Completed -- Database requires 1791 pages of memory
Total Errors = 0
```

APPENDIX C - RECIPE PROTOTYPE FILES (RPF)

ALLO

```
RMS Version 6.3-2                    05-Dec-94 09:31:35
RMS?> = rpf de all0
RPF?> = rpf de all0
Invalid Command
Open error file SV0:[300,1]U.UPF ,error = -1
RPF?> = cr all0,*
MRPF?>  = add 32 fg

Enter new 6 char. name of fg 0?> = flag0
Enter new 18 char. descriptor of fg 0?> = flag 0

Enter new 6 char. name of fg 1?> = flag1
Enter new 18 char. descriptor of fg 1?> = flag 1

Enter new 6 char. name of fg 2?> = flag2
Enter new 18 char. descriptor of fg 2?> = flag 2

Enter new 6 char. name of fg 3?> = flag3
Enter new 18 char. descriptor of fg 3?> = flag 3

Enter new 6 char. name of fg 4?> = flag4
Enter new 18 char. descriptor of fg 4?> = flag 4

Enter new 6 char. name of fg 5?> = flag5
Enter new 18 char. descriptor of fg 5?> = flag 5

Enter new 6 char. name of fg 6?> = flag6
Enter new 18 char. descriptor of fg 6?> = flag 6

Enter new 6 char. name of fg 7?> = flag7
Enter new 18 char. descriptor of fg 7?> = flag 7

Enter new 6 char. name of fg 8?> = flag8
Enter new 18 char. descriptor of fg 8?> = flag 8

Enter new 6 char. name of fg 9?> = flag9
Enter new 18 char. descriptor of fg 9?> = flag 9

Enter new 6 char. name of fg 10?> = flag10
Enter new 18 char. descriptor of fg 10?> = flag 10

Enter new 6 char. name of fg 11?> = flag11
Enter new 18 char. descriptor of fg 11?> = flag 11

Enter new 6 char. name of fg 12?> = flag12
Enter new 18 char. descriptor of fg 12?> = flag 12

Enter new 6 char. name of fg 13?> = flag13
Enter new 18 char. descriptor of fg 13?> = flag 13
```

```
Enter new 6 char. name of fg 14?> = flag14
Enter new 18 char. descriptor of fg 14?> = flag 14

Enter new 6 char. name of fg 15?> = flag15
Enter new 18 char. descriptor of fg 15?> = flag 15

Enter new 6 char. name of fg 16?> = flag16
Enter new 18 char. descriptor of fg 16?> = flag 16

Enter new 6 char. name of fg 17?> = flag17
Enter new 18 char. descriptor of fg 17?> = flag 17

Enter new 6 char. name of fg 18?> = flag18
Enter new 18 char. descriptor of fg 18?> = flag 18

Enter new 6 char. name of fg 19?> = flag19
Enter new 18 char. descriptor of fg 19?> = flag 19

Enter new 6 char. name of fg 20?> = flag20
Enter new 18 char. descriptor of fg 20?> = flag 20

Enter new 6 char. name of fg 21?> = flag21
Enter new 18 char. descriptor of fg 21?> = flag 21

Enter new 6 char. name of fg 22?> = flag22
Enter new 18 char. descriptor of fg 22?> = flag 22

Enter new 6 char. name of fg 23?> = flag23
Enter new 18 char. descriptor of fg 23?> = flag 23

Enter new 6 char. name of fg 24?> = flag24
Enter new 18 char. descriptor of fg 24?> = flag 24

Enter new 6 char. name of fg 25?> = flag25
Enter new 18 char. descriptor of fg 25?> = flag 25

Enter new 6 char. name of fg 26?> = flag26
Enter new 18 char. descriptor of fg 26?> = flag 26

Enter new 6 char. name of fg 27?> = flag27
Enter new 18 char. descriptor of fg 27?> = flag 27

Enter new 6 char. name of fg 28?> = flag28
Enter new 18 char. descriptor of fg 28?> = flag 28

Enter new 6 char. name of fg 29?> = flag29
Enter new 18 char. descriptor of fg 29?> = flag 29
```

```
Enter new 6 char. name of fg 30?> = flag30
Enter new 18 char. descriptor of fg 30?> = flag 30

Enter new 6 char. name of fg 31?> = flag31
Enter new 18 char. descriptor of fg 31?> = 31
MRPF?>  = add 40 fl

Enter new 6 char. name of fl 0?> = val0
Enter new 18 char. descriptor of fl 0?> = amps
Enter low limit for fl 0?> = 0
Enter high limit for fl 0?> = 13.6

Enter new 6 char. name of fl 1?> = val1
Enter new 18 char. descriptor of fl 1?> = pmp amps
Enter low limit for fl 1?> = 0
Enter high limit for fl 1?> = 13.6

Enter new 6 char. name of fl 2?> = val2
Enter new 18 char. descriptor of fl 2?> = pmp amps
Enter low limit for fl 2?> = 0
Enter high limit for fl 2?> = 13.6

Enter new 6 char. name of fl 3?> = val3
Enter new 18 char. descriptor of fl 3?> = agt amps
Enter low limit for fl 3?> = 0
Enter high limit for fl 3?> = 14.0

Enter new 6 char. name of fl 4?> = val4
Enter new 18 char. descriptor of fl 4?> = temp
Enter low limit for fl 4?> = 0
Enter high limit for fl 4?> = 200.0

Enter new 6 char. name of fl 5?> = val5
Enter new 18 char. descriptor of fl 5?> = flow
Enter low limit for fl 5?> = 375
Enter high limit for fl 5?> = 500

Enter new 6 char. name of fl 6?> = val6
Enter new 18 char. descriptor of fl 6?> = pmp amps
Enter low limit for fl 6?> = 0.0
Enter high limit for fl 6?> = 50.00

Enter new 6 char. name of fl 7?> = val7
Enter new 18 char. descriptor of fl 7?> = pmp amps
Enter low limit for fl 7?> = 0
Enter high limit for fl 7?> = 34.00
```

```
Enter new 6 char. name of fl 8?> = val8
Enter new 18 char. descriptor of fl 8?> = dlta spg
Enter low limit for fl 8?> = -0.1
Enter high limit for fl 8?> = 00.1

Enter new 6 char. name of fl 9?> = val9
Enter new 18 char. descriptor of fl 9?> = rw dp
Enter low limit for fl 9?> = 0
Enter high limit for fl 9?> = 30.00

Enter new 6 char. name of fl 10?> = val10
Enter new 18 char. descriptor of fl 10?> = rw dp
Enter low limit for fl 10?> = 0
Enter high limit for fl 10?> = 30.00

Enter new 6 char. name of fl 11?> = val11
Enter new 18 char. descriptor of fl 11?> = rw dp
Enter low limit for fl 11?> = 0
Enter high limit for fl 11?> = 30.00

Enter new 6 char. name of fl 12?> = val12
Enter new 18 char. descriptor of fl 12?> = slry spd
Enter low limit for fl 12?> = 0
Enter high limit for fl 12?> = 100.0

Enter new 6 char. name of fl 13?> = val13
Enter new 18 char. descriptor of fl 13?> = 90# stm
Enter low limit for fl 13?> = 0
Enter high limit for fl 13?> = 100.00

Enter new 6 char. name of fl 14?> = val14
Enter new 18 char. descriptor of fl 14?> = stm flw
Enter low limit for fl 14?> = 0.0
Enter high limit for fl 14?> = 30000.0

Enter new 6 char. name of fl 15?> = val15
Enter new 18 char. descriptor of fl 15?> = stm inlet
Enter low limit for fl 15?> = 15.00
Enter high limit for fl 15?> = 40.00

Enter new 6 char. name of fl 16?> = val16
Enter new 18 char. descriptor of fl 16?> = sc temp
Enter low limit for fl 16?> = 100.0
Enter high limit for fl 16?> = 200.0

Enter new 6 char. name of fl 17?> = val17
Enter new 18 char. descriptor of fl 17?> = rw inlet
Enter low limit for fl 17?> = 0.0
Enter high limit for fl 17?> = 100.00
```

```
Enter new 6 char. name of fl 18?> = val18
Enter new 18 char. descriptor of fl 18?> = urw flow
Enter low limit for fl 18?> = 0.0
Enter high limit for fl 18?> = 150.0

Enter new 6 char. name of fl 19?> = val19
Enter new 18 char. descriptor of fl 19?> = filter dp
Enter low limit for fl 19?> = 0.0
Enter high limit for fl 19?> = 30.0

Enter new 6 char. name of fl 20?> = val20
Enter new 18 char. descriptor of fl 20?> = filter dp
Enter low limit for fl 20?> = 0.0
Enter high limit for fl 20?> = 30.0

Enter new 6 char. name of fl 21?> = val21
Enter new 18 char. descriptor of fl 21?> = urw flow
Enter low limit for fl 21?> = 0.0
Enter high limit for fl 21?> = 4000.0

Enter new 6 char. name of fl 22?> = val22
Enter new 18 char. descriptor of fl 22?> = vv amps
Enter low limit for fl 22?> = 0
Enter high limit for fl 22?> = 4.8

Enter new 6 char. name of fl 23?> = val23
Enter new 18 char. descriptor of fl 23?> = vv flow
Enter low limit for fl 23?> = 0.0
Enter high limit for fl 23?> = 1000.0

Enter new 6 char. name of fl 24?> = val24
Enter new 18 char. descriptor of fl 24?> = pmp amps
Enter low limit for fl 24?> = 0.0
Enter high limit for fl 24?> = 15.20

Enter new 6 char. name of fl 25?> = val25
Enter new 18 char. descriptor of fl 25?> = pc amps
Enter low limit for fl 25?> = 0.0
Enter high limit for fl 25?> = 18.33

Enter new 6 char. name of fl 26?> = val26
Enter new 18 char. descriptor of fl 26?> = lo iface
Enter low limit for fl 26?> = 0.8
Enter high limit for fl 26?> = 1.00

Enter new 6 char. name of fl 27?> = val27
Enter new 18 char. descriptor of fl 27?> = pc amps
Enter low limit for fl 27?> = 0.0
Enter high limit for fl 27?> = 18.67
```

```
Enter new 6 char. name of fl 28?> = val28
Enter new 18 char. descriptor of fl 28?> = float value 28
Enter low limit for fl 28?> = -9999
Enter high limit for fl 28?> = 99999

Enter new 6 char. name of fl 29?> = val29
Enter new 18 char. descriptor of fl 29?> = float value 29
Enter low limit for fl 29?> = -9999
Enter high limit for fl 29?> = 99999

Enter new 6 char. name of fl 30?> = val30
Enter new 18 char. descriptor of fl 30?> = float value 30
Enter low limit for fl 30?> = -9999
Enter high limit for fl 30?> = 99999

Enter new 6 char. name of fl 31?> = val31
Enter new 18 char. descriptor of fl 31?> = float value 31
Enter low limit for fl 31?> = -9999
Enter high limit for fl 31?> = 99999

Enter new 6 char. name of fl 32?> = val32
Enter new 18 char. descriptor of fl 32?> = float value 32
Enter low limit for fl 32?> = -9999
Enter high limit for fl 32?> = 99999

Enter new 6 char. name of fl 33?> = val33
Enter new 18 char. descriptor of fl 33?> = float value 33
Enter low limit for fl 33?> = -9999
Enter high limit for fl 33?> = 99999

Enter new 6 char. name of fl 34?> = val34
Enter new 18 char. descriptor of fl 34?> = float value 34
Enter low limit for fl 34?> = -9999
Enter high limit for fl 34?> = 99999

Enter new 6 char. name of fl 35?> = val35
Enter new 18 char. descriptor of fl 35?> = float value 35
Enter low limit for fl 35?> = -9999
Enter high limit for fl 35?> = 99999

Enter new 6 char. name of fl 36?> = val36
Enter new 18 char. descriptor of fl 36?> = float value 36
Enter low limit for fl 36?> = -9999
Enter high limit for fl 36?> = 99999

Enter new 6 char. name of fl 37?> = val37
Enter new 18 char. descriptor of fl 37?> = float value 37
Enter low limit for fl 37?> = -9999
Enter high limit for fl 37?> = 99999
```

```
Enter new 6 char. name of fl 38?> = val38
Enter new 18 char. descriptor of fl 38?> = float value 38
Enter low limit for fl 38?> = -9999
Enter high limit for fl 38?> = 99999

Enter new 6 char. name of fl 39?> = val39
Enter new 18 char. descriptor of fl 39?> = float value 39
Enter low limit for fl 39?> = -9999
Enter high limit for fl 39?> = 99999
MRPF?>  = add 50 in

Enter new 6 char. name of in 0?> = in0
Enter new 18 char. descriptor of in 0?> = integer number 0
Enter low limit for in 0?> = 0
Enter high limit for in 0?> = 100

Enter new 6 char. name of in 1?> = in1
Enter new 18 char. descriptor of in 1?> = integer number 1
Enter low limit for in 1?> = 0
Enter high limit for in 1?> = 100

Enter new 6 char. name of in 2?> = in2
Enter new 18 char. descriptor of in 2?> = integer number 2
Enter low limit for in 2?> = 0
Enter high limit for in 2?> = 100

Enter new 6 char. name of in 3?> = in3
Enter new 18 char. descriptor of in 3?> = integer number 3
Enter low limit for in 3?> = 0
Enter high limit for in 3?> = 100

Enter new 6 char. name of in 4?> = in4
Enter new 18 char. descriptor of in 4?> = integer number 4
Enter low limit for in 4?> = 0
Enter high limit for in 4?> = 100

Enter new 6 char. name of in 5?> = in5
Enter new 18 char. descriptor of in 5?> = integer number 5
Enter low limit for in 5?> = 0
Enter high limit for in 5?> = 100

Enter new 6 char. name of in 6?> = in6
Enter new 18 char. descriptor of in 6?> = integer number 6
Enter low limit for in 6?> = 0
Enter high limit for in 6?> = 100

Enter new 6 char. name of in 7?> = in7
Enter new 18 char. descriptor of in 7?> = integer number 7
Enter low limit for in 7?> = 0
Enter high limit for in 7?> = 100
```

```
Enter new 6 char. name of in 8?> = in8
Enter new 18 char. descriptor of in 8?> = integer number 8
Enter low limit for in 8?> = 0
Enter high limit for in 8?> = 7

Enter new 6 char. name of in 9?> = in9
Enter new 18 char. descriptor of in 9?> = integer number 9
Enter low limit for in 9?> = 0
Enter high limit for in 9?> = 7

Enter new 6 char. name of in 10?> = in10
Enter new 18 char. descriptor of in 10?> = integer number 10
Enter low limit for in 10?> = 0
Enter high limit for in 10?> = 7

Enter new 6 char. name of in 11?> = in11
Enter new 18 char. descriptor of in 11?> = integer number 11
Enter low limit for in 11?> = 0
Enter high limit for in 11?> = 7

Enter new 6 char. name of in 12?> = in12
Enter new 18 char. descriptor of in 12?> = integer number 12
Enter low limit for in 12?> = 0
Enter high limit for in 12?> = 7

Enter new 6 char. name of in 13?> = in13
Enter new 18 char. descriptor of in 13?> = integer number 13
Enter low limit for in 13?> = 0
Enter high limit for in 13?> = 7

Enter new 6 char. name of in 14?> = in14
Enter new 18 char. descriptor of in 14?> = integer number 14
Enter low limit for in 14?> = 0
Enter high limit for in 14?> = 7

Enter new 6 char. name of in 15?> = in15
Enter new 18 char. descriptor of in 15?> = integer number 15
Enter low limit for in 15?> = 0
Enter high limit for in 15?> = 7

Enter new 6 char. name of in 16?> = in16
Enter new 18 char. descriptor of in 16?> = integer number 16
Enter low limit for in 16?> = 0
Enter high limit for in 16?> = 7

Enter new 6 char. name of in 17?> = in17
Enter new 18 char. descriptor of in 17?> = integer number 17
Enter low limit for in 17?> = 0
Enter high limit for in 17?> = 7
```

```
Enter new 6 char. name of in 18?> = in18
Enter new 18 char. descriptor of in 18?> = integer number 18
Enter low limit for in 18?> = 0
Enter high limit for in 18?> = 7

Enter new 6 char. name of in 19?> = in19
Enter new 18 char. descriptor of in 19?> = integer number 19
Enter low limit for in 19?> = 0
Enter high limit for in 19?> = 7

Enter new 6 char. name of in 20?> = in20
Enter new 18 char. descriptor of in 20?> = integer number 20
Enter low limit for in 20?> = 0
Enter high limit for in 20?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 21?> = in21
Enter new 18 char. descriptor of in 21?> = integer number 21
Enter low limit for in 21?> = 0
Enter high limit for in 21?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 22?> = in22
Enter new 18 char. descriptor of in 22?> = integer number 22
Enter low limit for in 22?> = 0
Enter high limit for in 22?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 23?> = in23
Enter new 18 char. descriptor of in 23?> = integer number 23
Enter low limit for in 23?> = 0
Enter high limit for in 23?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 24?> = in24
Enter new 18 char. descriptor of in 24?> = integer number 24
Enter low limit for in 24?> = 0
Enter high limit for in 24?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 25?> = in25
Enter new 18 char. descriptor of in 25?> = integer number 25
Enter low limit for in 25?> = 0
Enter high limit for in 25?> = 100000
High limit is less than low - setting high to 32767
```

```
Enter new 6 char. name of in 26?> = in26
Enter new 18 char. descriptor of in 26?> = integer number 26
Enter low limit for in 26?> = 0
Enter high limit for in 26?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 27?> = in27
Enter new 18 char. descriptor of in 27?> = integer number 27
Enter low limit for in 27?> = 0
Enter high limit for in 27?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 28?> = in28
Enter new 18 char. descriptor of in 28?> = integer number 28
Enter low limit for in 28?> = 0
Enter high limit for in 28?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 29?> = in29
Enter new 18 char. descriptor of in 29?> = integer number 29
Enter low limit for in 29?> = 0
Enter high limit for in 29?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 30?> = in30
Enter new 18 char. descriptor of in 30?> = integer number 30
Enter low limit for in 30?> = 0
Enter high limit for in 30?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 31?> = in31
Enter new 18 char. descriptor of in 31?> = integer number 31
Enter low limit for in 31?> = 0
Enter high limit for in 31?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 32?> = in32
Enter new 18 char. descriptor of in 32?> = integer number 32
Enter low limit for in 32?> = 0
Enter high limit for in 32?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 33?> = in33
Enter new 18 char. descriptor of in 33?> = integer number 33
Enter low limit for in 33?> = 0
Enter high limit for in 33?> = 100000
High limit is less than low - setting high to 32767
```

```
Enter new 6 char. name of in 34?> = in34
Enter new 18 char. descriptor of in 34?> = integer number 34
Enter low limit for in 34?> = 0
Enter high limit for in 34?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 35?> = in35
Enter new 18 char. descriptor of in 35?> = integer number 35
Enter low limit for in 35?> = 0
Enter high limit for in 35?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 36?> = in36
Enter new 18 char. descriptor of in 36?> = integer number 36
Enter low limit for in 36?> = 0
Enter high limit for in 36?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 37?> = in37
Enter new 18 char. descriptor of in 37?> = integer number 37
Enter low limit for in 37?> = 0
Enter high limit for in 37?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 38?> = in38
Enter new 18 char. descriptor of in 38?> = integer number 38
Enter low limit for in 38?> = 0
Enter high limit for in 38?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 39?> = in39
Enter new 18 char. descriptor of in 39?> = integer number 39
Enter low limit for in 39?> = 0
Enter high limit for in 39?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 40?> = imisc1
Enter new 18 char. descriptor of in 40?> = misc1 val1
Enter low limit for in 40?> = 0
Enter high limit for in 40?> = 128

Enter new 6 char. name of in 41?> = imisc2
Enter new 18 char. descriptor of in 41?> = misc2 val2
Enter low limit for in 41?> = 0
Enter high limit for in 41?> = 128

Enter new 6 char. name of in 42?> = adi
Enter new 18 char. descriptor of in 42?> = alm des index
Enter low limit for in 42?> = 0
Enter high limit for in 42?> = 80
```

```
Enter new 6 char. name of in 43?> = lmpvar
Enter new 18 char. descriptor of in 43?> = lamp var
Enter low limit for in 43?> = 0
Enter high limit for in 43?> = 4

Enter new 6 char. name of in 44?> = led0
Enter new 18 char. descriptor of in 44?> = led 0
Enter low limit for in 44?> = 0
Enter high limit for in 44?> = 63

Enter new 6 char. name of in 45?> = led1
Enter new 18 char. descriptor of in 45?> = led 1
Enter low limit for in 45?> = 0
Enter high limit for in 45?> = 63

Enter new 6 char. name of in 46?> = led2
Enter new 18 char. descriptor of in 46?> = led 2
Enter low limit for in 46?> = 0
Enter high limit for in 46?> = 63

Enter new 6 char. name of in 47?> = ispar0
Enter new 18 char. descriptor of in 47?> = spare int 0
Enter low limit for in 47?> = 0
Enter high limit for in 47?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 48?> = ispar1
Enter new 18 char. descriptor of in 48?> = spare int 1
Enter low limit for in 48?> = 0
Enter high limit for in 48?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 49?> = ispar2
Enter new 18 char. descriptor of in 49?> = spare int 2
Enter low limit for in 49?> = 0
Enter high limit for in 49?> = 37000
High limit is less than low - setting high to 32767
MRPF?>  = add 5 st

Enter new 6 char. name of st 0?> = stmr0
Enter new 18 char. descriptor of st 0?> = second timer 0
Enter low limit for st 0?> = 0
Enter high limit for st 0?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of st 1?> = stmr1
Enter new 18 char. descriptor of st 1?> = second timer 1
Enter low limit for st 1?> = 0
Enter high limit for st 1?> = 300
```

```
Enter new 6 char. name of st 2?> = stmr2
Enter new 18 char. descriptor of st 2?> = second timer 2
Enter low limit for st 2?> = 0
Enter high limit for st 2?> = 255

Enter new 6 char. name of st 3?> = stmr3
Enter new 18 char. descriptor of st 3?> = second timer 3
Enter low limit for st 3?> = 0
Enter high limit for st 3?> = 75

Enter new 6 char. name of st 4?> = stmr4
Enter new 18 char. descriptor of st 4?> = second timer 4
Enter low limit for st 4?> = 0
Enter high limit for st 4?> = 600
MRPF?>  = add 5 sg

Enter new 6 char. name of sg 0?> = note0
Enter new 18 char. descriptor of sg 0?> = total note 0
Enter maximum size for sg 0?> = 30

Enter new 6 char. name of sg 1?> = note1
Enter new 18 char. descriptor of sg 1?> = total note 1
Enter maximum size for sg 1?> = 30

Enter new 6 char. name of sg 2?> = note2
Enter new 18 char. descriptor of sg 2?> = total note 2
Enter maximum size for sg 2?> = 30

Enter new 6 char. name of sg 3?> = note3
Enter new 18 char. descriptor of sg 3?> = total note 3
Enter maximum size for sg 3?> = 30

Enter new 6 char. name of sg 4?> = note4
Enter new 18 char. descriptor of sg 4?> = total note 4
Enter maximum size for sg 4?> = 30
MRPF?>  =
RPF?>
 end of file encountered
RMS?>
 end of file encountered
```

242A

```
RMS Version 6.3-2                    05-Dec-94 09:32:03
RMS?> = rpf de 242a
RPF?> = rpf de 242a
Invalid Command
Open error file SV0:[300,1]U.UPF ,error = -1
Open error file SV0:[300,1]U.UPF ,error = -1
RPF?> = cr 242a,*
MRPF?>  = add 16 fg

Enter new 6 char. name of fg 0?> = flg0
Enter new 18 char. descriptor of fg 0?> = unit flag 0

Enter new 6 char. name of fg 1?> = flg1
Enter new 18 char. descriptor of fg 1?> = unit flag 1

Enter new 6 char. name of fg 2?> = flg2
Enter new 18 char. descriptor of fg 2?> = unit flag 2

Enter new 6 char. name of fg 3?> = flg3
Enter new 18 char. descriptor of fg 3?> = unit flag 3

Enter new 6 char. name of fg 4?> = flg4
Enter new 18 char. descriptor of fg 4?> = unit flag 4

Enter new 6 char. name of fg 5?> = flg5
Enter new 18 char. descriptor of fg 5?> = unit flag 5

Enter new 6 char. name of fg 6?> = flg6
Enter new 18 char. descriptor of fg 6?> = unit flag 6

Enter new 6 char. name of fg 7?> = flg7
Enter new 18 char. descriptor of fg 7?> = unit flag 7

Enter new 6 char. name of fg 8?> = flg8
Enter new 18 char. descriptor of fg 8?> = unit flag 8

Enter new 6 char. name of fg 9?> = flg9
Enter new 18 char. descriptor of fg 9?> = unit flag 9

Enter new 6 char. name of fg 10?> = flg10
Enter new 18 char. descriptor of fg 10?> = unit flag 10

Enter new 6 char. name of fg 11?> = flg11
Enter new 18 char. descriptor of fg 11?> = unit flag 11

Enter new 6 char. name of fg 12?> = flg12
Enter new 18 char. descriptor of fg 12?> = unit flag 12
```

```
Enter new 6 char. name of fg 13?> = flg13
Enter new 18 char. descriptor of fg 13?> = unit flag 13

Enter new 6 char. name of fg 14?> = flg14
Enter new 18 char. descriptor of fg 14?> = unit flag 14

Enter new 6 char. name of fg 15?> = flg15
Enter new 18 char. descriptor of fg 15?> = unit flag 15
MRPF?>  = add 20 fl

Enter new 6 char. name of fl 0?> = wf0
Enter new 18 char. descriptor of fl 0?> = wf float 0
Enter low limit for fl 0?> = 0
Enter high limit for fl 0?> = 100

Enter new 6 char. name of fl 1?> = wf1
Enter new 18 char. descriptor of fl 1?> = wf float 1
Enter low limit for fl 1?> = 0
Enter high limit for fl 1?> = 100

Enter new 6 char. name of fl 2?> = wf2
Enter new 18 char. descriptor of fl 2?> = wf float 2
Enter low limit for fl 2?> = 0
Enter high limit for fl 2?> = 100

Enter new 6 char. name of fl 3?> = wf3
Enter new 18 char. descriptor of fl 3?> = wf float 3
Enter low limit for fl 3?> = 0
Enter high limit for fl 3?> = 100

Enter new 6 char. name of fl 4?> = wf4
Enter new 18 char. descriptor of fl 4?> = wf float 4
Enter low limit for fl 4?> = 0
Enter high limit for fl 4?> = 100

Enter new 6 char. name of fl 5?> = psi0
Enter new 18 char. descriptor of fl 5?> = pressure 0
Enter low limit for fl 5?> = 0
Enter high limit for fl 5?> = 100

Enter new 6 char. name of fl 6?> = psi1
Enter new 18 char. descriptor of fl 6?> = pressure 1
Enter low limit for fl 6?> = 0
Enter high limit for fl 6?> = 100

Enter new 6 char. name of fl 7?> = psi2
Enter new 18 char. descriptor of fl 7?> = pressure 2
Enter low limit for fl 7?> = 0
Enter high limit for fl 7?> = 100
```

```
Enter new 6 char. name of fl 8?> = psi3
Enter new 18 char. descriptor of fl 8?> = pressure 3
Enter low limit for fl 8?> = 0
Enter high limit for fl 8?> = 100

Enter new 6 char. name of fl 9?> = psi4
Enter new 18 char. descriptor of fl 9?> = pressure 4
Enter low limit for fl 9?> = 0
Enter high limit for fl 9?> = 100

Enter new 6 char. name of fl 10?> = di0
Enter new 18 char. descriptor of fl 10?> = density 0
Enter low limit for fl 10?> = 0
Enter high limit for fl 10?> = 100

Enter new 6 char. name of fl 11?> = di1
Enter new 18 char. descriptor of fl 11?> = density 1
Enter low limit for fl 11?> = 0
Enter high limit for fl 11?> = 100

Enter new 6 char. name of fl 12?> = di2
Enter new 18 char. descriptor of fl 12?> = density 2
Enter low limit for fl 12?> = 0
Enter high limit for fl 12?> = 100

Enter new 6 char. name of fl 13?> = di3
Enter new 18 char. descriptor of fl 13?> = density 3
Enter low limit for fl 13?> = 0
Enter high limit for fl 13?> = 100

Enter new 6 char. name of fl 14?> = di4
Enter new 18 char. descriptor of fl 14?> = density 4
Enter low limit for fl 14?> = 0
Enter high limit for fl 14?> = 100

Enter new 6 char. name of fl 15?> = vl0
Enter new 18 char. descriptor of fl 15?> = float spare 0
Enter low limit for fl 15?> = 0
Enter high limit for fl 15?> = 100

Enter new 6 char. name of fl 16?> = vl1
Enter new 18 char. descriptor of fl 16?> = float spare 1
Enter low limit for fl 16?> = 0
Enter high limit for fl 16?> = 100

Enter new 6 char. name of fl 17?> = vl2
Enter new 18 char. descriptor of fl 17?> = float spare 2
Enter low limit for fl 17?> = 0
Enter high limit for fl 17?> = 100
```

```
Enter new 6 char. name of fl 18?> = vl3
Enter new 18 char. descriptor of fl 18?> = float spare 3
Enter low limit for fl 18?> = 0
Enter high limit for fl 18?> = 100

Enter new 6 char. name of fl 19?> = vl4
Enter new 18 char. descriptor of fl 19?> = float spare 4
Enter low limit for fl 19?> = 0
Enter high limit for fl 19?> = 100
MRPF?>  = add 20 in

Enter new 6 char. name of in 0?> = in0
Enter new 18 char. descriptor of in 0?> = integer number 0
Enter low limit for in 0?> = 0
Enter high limit for in 0?> = 1000

Enter new 6 char. name of in 1?> = in1
Enter new 18 char. descriptor of in 1?> = integer number 1
Enter low limit for in 1?> = 0
Enter high limit for in 1?> = 1000

Enter new 6 char. name of in 2?> = in2
Enter new 18 char. descriptor of in 2?> = integer number 2
Enter low limit for in 2?> = 0
Enter high limit for in 2?> = 1000

Enter new 6 char. name of in 3?> = in3
Enter new 18 char. descriptor of in 3?> = integer number 3
Enter low limit for in 3?> = 0
Enter high limit for in 3?> = 1000

Enter new 6 char. name of in 4?> = in4
Enter new 18 char. descriptor of in 4?> = integer number 4
Enter low limit for in 4?> = 0
Enter high limit for in 4?> = 1000

Enter new 6 char. name of in 5?> = in5
Enter new 18 char. descriptor of in 5?> = integer number 5
Enter low limit for in 5?> = 0
Enter high limit for in 5?> = 100

Enter new 6 char. name of in 6?> = in6
Enter new 18 char. descriptor of in 6?> = integer number 6
Enter low limit for in 6?> = 0
Enter high limit for in 6?> = 100

Enter new 6 char. name of in 7?> = in7
Enter new 18 char. descriptor of in 7?> = integer number 7
Enter low limit for in 7?> = 0
Enter high limit for in 7?> = 100
```

```
Enter new 6 char. name of in 8?> = in8
Enter new 18 char. descriptor of in 8?> = integer number 8
Enter low limit for in 8?> = 0
Enter high limit for in 8?> = 100

Enter new 6 char. name of in 9?> = in9
Enter new 18 char. descriptor of in 9?> = integer number 9
Enter low limit for in 9?> = 0
Enter high limit for in 9?> = 100

Enter new 6 char. name of in 10?> = in10
Enter new 18 char. descriptor of in 10?> = integer number 10
Enter low limit for in 10?> = 0
Enter high limit for in 10?> = 7

Enter new 6 char. name of in 11?> = in11
Enter new 18 char. descriptor of in 11?> = integer number 11
Enter low limit for in 11?> = 0
Enter high limit for in 11?> = 7

Enter new 6 char. name of in 12?> = in12
Enter new 18 char. descriptor of in 12?> = integer number 12
Enter low limit for in 12?> = 0
Enter high limit for in 12?> = 7

Enter new 6 char. name of in 13?> = in13
Enter new 18 char. descriptor of in 13?> = integer number 13
Enter low limit for in 13?> = 0
Enter high limit for in 13?> = 7

Enter new 6 char. name of in 14?> = in14
Enter new 18 char. descriptor of in 14?> = integer number 14
Enter low limit for in 14?> = 0
Enter high limit for in 14?> = 7

Enter new 6 char. name of in 15?> = in15
Enter new 18 char. descriptor of in 15?> = integer number 15
Enter low limit for in 15?> = 0
Enter high limit for in 15?> = 7

Enter new 6 char. name of in 16?> = in16
Enter new 18 char. descriptor of in 16?> = integer number 16
Enter low limit for in 16?> = 0
Enter high limit for in 16?> = 7

Enter new 6 char. name of in 17?> = in17
Enter new 18 char. descriptor of in 17?> = integer number 17
Enter low limit for in 17?> = 0
Enter high limit for in 17?> = 7
```

```
Enter new 6 char. name of in 18?> = in18
Enter new 18 char. descriptor of in 18?> = integer number 18
Enter low limit for in 18?> = 0
Enter high limit for in 18?> = 7

Enter new 6 char. name of in 19?> = in19
Enter new 18 char. descriptor of in 19?> = integer number 19
Enter low limit for in 19?> = 0
Enter high limit for in 19?> = 7
MRPF?>  = add 8 st

Enter new 6 char. name of st 0?> = minst
Enter new 18 char. descriptor of st 0?> = one minute timer
Enter low limit for st 0?> = 0
Enter high limit for st 0?> = 60

Enter new 6 char. name of st 1?> = fivest
Enter new 18 char. descriptor of st 1?> = five minute timer
Enter low limit for st 1?> = 0
Enter high limit for st 1?> = 300

Enter new 6 char. name of st 2?> = tenst
Enter new 18 char. descriptor of st 2?> = ten minute timer
Enter low limit for st 2?> = 0
Enter high limit for st 2?> = 600

Enter new 6 char. name of st 3?> = pigtmr
Enter new 18 char. descriptor of st 3?> = rc pig timer
Enter low limit for st 3?> = 0
Enter high limit for st 3?> = 255

Enter new 6 char. name of st 4?> = ostmr
Enter new 18 char. descriptor of st 4?> = one shot timer
Enter low limit for st 4?> = 0
Enter high limit for st 4?> = 75

Enter new 6 char. name of st 5?> = stime0
Enter new 18 char. descriptor of st 5?> = pgm error timer
Enter low limit for st 5?> = 0
Enter high limit for st 5?> = 10000

Enter new 6 char. name of st 6?> = stime1
Enter new 18 char. descriptor of st 6?> = pgm utility timer
Enter low limit for st 6?> = 0
Enter high limit for st 6?> = 10000

Enter new 6 char. name of st 7?> = sprtmr
Enter new 18 char. descriptor of st 7?> = spare sec timer
Enter low limit for st 7?> = 0
Enter high limit for st 7?> = 10000
```

```
MRPF?>  = add 5 sg

Enter new 6 char. name of sg 0?> = note0
Enter new 18 char. descriptor of sg 0?> = total note 0
Enter maximum size for sg 0?> = 30

Enter new 6 char. name of sg 1?> = note1
Enter new 18 char. descriptor of sg 1?> = total note 1
Enter maximum size for sg 1?> = 30

Enter new 6 char. name of sg 2?> = note2
Enter new 18 char. descriptor of sg 2?> = total note 2
Enter maximum size for sg 2?> = 30

Enter new 6 char. name of sg 3?> = note3
Enter new 18 char. descriptor of sg 3?> = total note 3
Enter maximum size for sg 3?> = 30

Enter new 6 char. name of sg 4?> = note4
Enter new 18 char. descriptor of sg 4?> = total note 4
Enter maximum size for sg 4?> = 30
MRPF?>  =
RPF?>
 end of file encountered
RMS?>
 end of file encountered
```

ALL1

```
RMS Version 6.3-2                    05-Dec-94 09:33:03
RMS?> = rpf de all
RPF?> = rpf de all
Invalid Command
Open error file SV0:[300,1]U.UPF ,error = -1
RPF?> = cr all,*
MRPF?>  = add 25 fl

Enter new 6 char. name of fl 0?> = val0
Enter new 18 char. descriptor of fl 0?> = value number 0
Enter low limit for fl 0?> = 0
Enter high limit for fl 0?> = 10000

Enter new 6 char. name of fl 1?> = val1
Enter new 18 char. descriptor of fl 1?> = value number 1
Enter low limit for fl 1?> = 0
Enter high limit for fl 1?> = 10000

Enter new 6 char. name of fl 2?> = val2
Enter new 18 char. descriptor of fl 2?> = value number 2
Enter low limit for fl 2?> = 0
Enter high limit for fl 2?> = 10000

Enter new 6 char. name of fl 3?> = val3
Enter new 18 char. descriptor of fl 3?> = value number 3
Enter low limit for fl 3?> = 0
Enter high limit for fl 3?> = 10000

Enter new 6 char. name of fl 4?> = val4
Enter new 18 char. descriptor of fl 4?> = value number 4
Enter low limit for fl 4?> = 0
Enter high limit for fl 4?> = 10000

Enter new 6 char. name of fl 5?> = val5
Enter new 18 char. descriptor of fl 5?> = value number 5
Enter low limit for fl 5?> = 0
Enter high limit for fl 5?> = 10000

Enter new 6 char. name of fl 6?> = val6
Enter new 18 char. descriptor of fl 6?> = value number 6
Enter low limit for fl 6?> = 0
Enter high limit for fl 6?> = 10000

Enter new 6 char. name of fl 7?> = val7
Enter new 18 char. descriptor of fl 7?> = value number 7
Enter low limit for fl 7?> = 0
Enter high limit for fl 7?> = 10000
```

```
Enter new 6 char. name of fl 8?> = val8
Enter new 18 char. descriptor of fl 8?> = value number 8
Enter low limit for fl 8?> = 0
Enter high limit for fl 8?> = 10000

Enter new 6 char. name of fl 9?> = val9
Enter new 18 char. descriptor of fl 9?> = value number 9
Enter low limit for fl 9?> = 0
Enter high limit for fl 9?> = 10000

Enter new 6 char. name of fl 10?> = val10
Enter new 18 char. descriptor of fl 10?> = value number 10
Enter low limit for fl 10?> = 0
Enter high limit for fl 10?> = 10000

Enter new 6 char. name of fl 11?> = val11
Enter new 18 char. descriptor of fl 11?> = value number 11
Enter low limit for fl 11?> = 0
Enter high limit for fl 11?> = 10000

Enter new 6 char. name of fl 12?> = val12
Enter new 18 char. descriptor of fl 12?> = value number 12
Enter low limit for fl 12?> = 0
Enter high limit for fl 12?> = 10000

Enter new 6 char. name of fl 13?> = val13
Enter new 18 char. descriptor of fl 13?> = value number 13
Enter low limit for fl 13?> = 0
Enter high limit for fl 13?> = 10000

Enter new 6 char. name of fl 14?> = val14
Enter new 18 char. descriptor of fl 14?> = value number 14
Enter low limit for fl 14?> = 0
Enter high limit for fl 14?> = 10000

Enter new 6 char. name of fl 15?> = val15
Enter new 18 char. descriptor of fl 15?> = value number 15
Enter low limit for fl 15?> = 0
Enter high limit for fl 15?> = 10000

Enter new 6 char. name of fl 16?> = val16
Enter new 18 char. descriptor of fl 16?> = value number 16
Enter low limit for fl 16?> = 0
Enter high limit for fl 16?> = 10000

Enter new 6 char. name of fl 17?> = val17
Enter new 18 char. descriptor of fl 17?> = value number 17
Enter low limit for fl 17?> = 0
Enter high limit for fl 17?> = 10000
```

```
Enter new 6 char. name of fl 18?> = val18
Enter new 18 char. descriptor of fl 18?> = value number 18
Enter low limit for fl 18?> = 0
Enter high limit for fl 18?> = 10000

Enter new 6 char. name of fl 19?> = val19
Enter new 18 char. descriptor of fl 19?> = value number 19
Enter low limit for fl 19?> = 0
Enter high limit for fl 19?> = 10000

Enter new 6 char. name of fl 20?> = val20
Enter new 18 char. descriptor of fl 20?> = value number 20
Enter low limit for fl 20?> = 0
Enter high limit for fl 20?> = 10000

Enter new 6 char. name of fl 21?> = val21
Enter new 18 char. descriptor of fl 21?> = value number 21
Enter low limit for fl 21?> = 0
Enter high limit for fl 21?> = 10000

Enter new 6 char. name of fl 22?> = val22
Enter new 18 char. descriptor of fl 22?> = value number 22
Enter low limit for fl 22?> = 0
Enter high limit for fl 22?> = 10000

Enter new 6 char. name of fl 23?> = val23
Enter new 18 char. descriptor of fl 23?> = value number 23
Enter low limit for fl 23?> = 0
Enter high limit for fl 23?> = 10000

Enter new 6 char. name of fl 24?> = val24
Enter new 18 char. descriptor of fl 24?> = value number 24
Enter low limit for fl 24?> = 0
Enter high limit for fl 24?> = 10000
MRPF?>  = add 50 in

Enter new 6 char. name of in 0?> = in0
Enter new 18 char. descriptor of in 0?> = integer number 0
Enter low limit for in 0?> = 0
Enter high limit for in 0?> = 7

Enter new 6 char. name of in 1?> = in1
Enter new 18 char. descriptor of in 1?> = integer number 1
Enter low limit for in 1?> = 0
Enter high limit for in 1?> = 7

Enter new 6 char. name of in 2?> = in2
Enter new 18 char. descriptor of in 2?> = integer number 2
Enter low limit for in 2?> = 0
Enter high limit for in 2?> = 7
```

```
Enter new 6 char. name of in 3?> = in3
Enter new 18 char. descriptor of in 3?> = integer number 3
Enter low limit for in 3?> = 0
Enter high limit for in 3?> = 7

Enter new 6 char. name of in 4?> = in4
Enter new 18 char. descriptor of in 4?> = integer number 4
Enter low limit for in 4?> = 0
Enter high limit for in 4?> = 7

Enter new 6 char. name of in 5?> = in5
Enter new 18 char. descriptor of in 5?> = integer number 5
Enter low limit for in 5?> = 0
Enter high limit for in 5?> = 7

Enter new 6 char. name of in 6?> = in6
Enter new 18 char. descriptor of in 6?> = integer number 6
Enter low limit for in 6?> = 0
Enter high limit for in 6?> = 7

Enter new 6 char. name of in 7?> = in7
Enter new 18 char. descriptor of in 7?> = integer number 7
Enter low limit for in 7?> = 0
Enter high limit for in 7?> = 7

Enter new 6 char. name of in 8?> = in8
Enter new 18 char. descriptor of in 8?> = integer number 8
Enter low limit for in 8?> = 0
Enter high limit for in 8?> = 7

Enter new 6 char. name of in 9?> = in9
Enter new 18 char. descriptor of in 9?> = integer number 9
Enter low limit for in 9?> = 0
Enter high limit for in 9?> = 7

Enter new 6 char. name of in 10?> = in10
Enter new 18 char. descriptor of in 10?> = integer number 10
Enter low limit for in 10?> = 0
Enter high limit for in 10?> = 7

Enter new 6 char. name of in 11?> = in11
Enter new 18 char. descriptor of in 11?> = integer number 11
Enter low limit for in 11?> = 0
Enter high limit for in 11?> = 7

Enter new 6 char. name of in 12?> = in12
Enter new 18 char. descriptor of in 12?> = integer number 12
Enter low limit for in 12?> = 0
Enter high limit for in 12?> = 7
```

```
Enter new 6 char. name of in 13?> = in13
Enter new 18 char. descriptor of in 13?> = integer number 13
Enter low limit for in 13?> = 0
Enter high limit for in 13?> = 7

Enter new 6 char. name of in 14?> = in14
Enter new 18 char. descriptor of in 14?> = integer number 14
Enter low limit for in 14?> = 0
Enter high limit for in 14?> = 7

Enter new 6 char. name of in 15?> = in15
Enter new 18 char. descriptor of in 15?> = integer number 15
Enter low limit for in 15?> = 0
Enter high limit for in 15?> = 7

Enter new 6 char. name of in 16?> = in16
Enter new 18 char. descriptor of in 16?> = integer number 16
Enter low limit for in 16?> = 0
Enter high limit for in 16?> = 7

Enter new 6 char. name of in 17?> = in17
Enter new 18 char. descriptor of in 17?> = integer number 17
Enter low limit for in 17?> = 0
Enter high limit for in 17?> = 7

Enter new 6 char. name of in 18?> = in18
Enter new 18 char. descriptor of in 18?> = integer number 18
Enter low limit for in 18?> = 0
Enter high limit for in 18?> = 7

Enter new 6 char. name of in 19?> = in19
Enter new 18 char. descriptor of in 19?> = integer number 19
Enter low limit for in 19?> = 0
Enter high limit for in 19?> = 7

Enter new 6 char. name of in 20?> = in20
Enter new 18 char. descriptor of in 20?> = integer number 20
Enter low limit for in 20?> = 0
Enter high limit for in 20?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 21?> = in21
Enter new 18 char. descriptor of in 21?> = integer number 21
Enter low limit for in 21?> = 0
Enter high limit for in 21?> = 100000
High limit is less than low - setting high to 32767
```

```
Enter new 6 char. name of in 22?> = in22
Enter new 18 char. descriptor of in 22?> = integer number 22
Enter low limit for in 22?> = 0
Enter high limit for in 22?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 23?> = in23
Enter new 18 char. descriptor of in 23?> = integer number 23
Enter low limit for in 23?> = 0
Enter high limit for in 23?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 24?> = in24
Enter new 18 char. descriptor of in 24?> = integer number 24
Enter low limit for in 24?> = 0
Enter high limit for in 24?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 25?> = in25
Enter new 18 char. descriptor of in 25?> = integer number 25
Enter low limit for in 25?> = 0
Enter high limit for in 25?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 26?> = in26
Enter new 18 char. descriptor of in 26?> = integer number 26
Enter low limit for in 26?> = 0
Enter high limit for in 26?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 27?> = in27
Enter new 18 char. descriptor of in 27?> = integer number 27
Enter low limit for in 27?> = 0
Enter high limit for in 27?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 28?> = in28
Enter new 18 char. descriptor of in 28?> = integer number 28
Enter low limit for in 28?> = 0
Enter high limit for in 28?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 29?> = in29
Enter new 18 char. descriptor of in 29?> = integer number 29
Enter low limit for in 29?> = 0
Enter high limit for in 29?> = 100000
High limit is less than low - setting high to 32767
```

```
Enter new 6 char. name of in 30?> = in30
Enter new 18 char. descriptor of in 30?> = integer number 30
Enter low limit for in 30?> = 0
Enter high limit for in 30?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 31?> = in31
Enter new 18 char. descriptor of in 31?> = integer number 31
Enter low limit for in 31?> = 0
Enter high limit for in 31?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 32?> = in32
Enter new 18 char. descriptor of in 32?> = integer number 32
Enter low limit for in 32?> = 0
Enter high limit for in 32?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 33?> = in33
Enter new 18 char. descriptor of in 33?> = integer number 33
Enter low limit for in 33?> = 0
Enter high limit for in 33?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 34?> = in34
Enter new 18 char. descriptor of in 34?> = integer number 34
Enter low limit for in 34?> = 0
Enter high limit for in 34?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 35?> = in35
Enter new 18 char. descriptor of in 35?> = integer number 35
Enter low limit for in 35?> = 0
Enter high limit for in 35?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 36?> = in36
Enter new 18 char. descriptor of in 36?> = integer number 36
Enter low limit for in 36?> = 0
Enter high limit for in 36?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 37?> = in37
Enter new 18 char. descriptor of in 37?> = integer number 37
Enter low limit for in 37?> = 0
Enter high limit for in 37?> = 100000
High limit is less than low - setting high to 32767
```

```
Enter new 6 char. name of in 38?> = in38
Enter new 18 char. descriptor of in 38?> = integer number 38
Enter low limit for in 38?> = 0
Enter high limit for in 38?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 39?> = in39
Enter new 18 char. descriptor of in 39?> = integer number 39
Enter low limit for in 39?> = 0
Enter high limit for in 39?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 40?> = imisc1
Enter new 18 char. descriptor of in 40?> = misc1 val1
Enter low limit for in 40?> = 0
Enter high limit for in 40?> = 128

Enter new 6 char. name of in 41?> = imisc2
Enter new 18 char. descriptor of in 41?> = misc2 val2
Enter low limit for in 41?> = 0
Enter high limit for in 41?> = 128

Enter new 6 char. name of in 42?> = adi
Enter new 18 char. descriptor of in 42?> = alm des index
Enter low limit for in 42?> = 0
Enter high limit for in 42?> = 80

Enter new 6 char. name of in 43?> = lmpvar
Enter new 18 char. descriptor of in 43?> = lamp var
Enter low limit for in 43?> = 0
Enter high limit for in 43?> = 4

Enter new 6 char. name of in 44?> = led0
Enter new 18 char. descriptor of in 44?> = led 0
Enter low limit for in 44?> = 0
Enter high limit for in 44?> = 63

Enter new 6 char. name of in 45?> = led1
Enter new 18 char. descriptor of in 45?> = led 1
Enter low limit for in 45?> = 0
Enter high limit for in 45?> = 63

Enter new 6 char. name of in 46?> = led2
Enter new 18 char. descriptor of in 46?> = led 2
Enter low limit for in 46?> = 0
Enter high limit for in 46?> = 63
```

```
Enter new 6 char. name of in 47?> = ispar0
Enter new 18 char. descriptor of in 47?> = spare int 0
Enter low limit for in 47?> = 0
Enter high limit for in 47?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 48?> = ispar1
Enter new 18 char. descriptor of in 48?> = spare int 1
Enter low limit for in 48?> = 0
Enter high limit for in 48?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 49?> = ispar2
Enter new 18 char. descriptor of in 49?> = spare int 2
Enter low limit for in 49?> = 0
Enter high limit for in 49?> = 37000
High limit is less than low - setting high to 32767
MRPF?>  = add 5 st

Enter new 6 char. name of st 0?> = stmr0
Enter new 18 char. descriptor of st 0?> = second timer 0
Enter low limit for st 0?> = 0
Enter high limit for st 0?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of st 1?> = stmr1
Enter new 18 char. descriptor of st 1?> = second timer 1
Enter low limit for st 1?> = 0
Enter high limit for st 1?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of st 2?> = stmr2
Enter new 18 char. descriptor of st 2?> = second timer 2
Enter low limit for st 2?> = 0
Enter high limit for st 2?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of st 3?> = stmr3
Enter new 18 char. descriptor of st 3?> = second timer 3
Enter low limit for st 3?> = 0
Enter high limit for st 3?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of st 4?> = stmr4
Enter new 18 char. descriptor of st 4?> = second timer 4
Enter low limit for st 4?> = 0
Enter high limit for st 4?> = 37000
High limit is less than low - setting high to 32767
MRPF?>  = add 5 mt
```

```
Enter new 6 char. name of mt 0?> = mtmr0
Enter new 18 char. descriptor of mt 0?> = minute timer 0
Enter low limit for mt 0?> = 0
Enter high limit for mt 0?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of mt 1?> = mtmr1
Enter new 18 char. descriptor of mt 1?> = minute timer 1
Enter low limit for mt 1?> = 0
Enter high limit for mt 1?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of mt 2?> = mtmr2
Enter new 18 char. descriptor of mt 2?> = minute timer 2
Enter low limit for mt 2?> = 0
Enter high limit for mt 2?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of mt 3?> = mtmr3
Enter new 18 char. descriptor of mt 3?> = minute timer 3
Enter low limit for mt 3?> = 0
Enter high limit for mt 3?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of mt 4?> = mtmr4
Enter new 18 char. descriptor of mt 4?> = minute timer 4
Enter low limit for mt 4?> = 0
Enter high limit for mt 4?> = 37000
High limit is less than low - setting high to 32767
MRPF?>
 end of file encountered
RPF?>
 end of file encountered
RMS?>
 end of file encountered
```

FARM

```
RMS Version 6.3-2                    05-Dec-94 09:32:21
RMS?> = rpf de farm
RPF?> = rpf de farm
RPF?> = cr farm,*
MRPF?>  = add 20 fl

Enter new 6 char. name of fl 0?> = val0
Enter new 18 char. descriptor of fl 0?> = value number 0
Enter low limit for fl 0?> = 0
Enter high limit for fl 0?> = 10000

Enter new 6 char. name of fl 1?> = val1
Enter new 18 char. descriptor of fl 1?> = value number 1
Enter low limit for fl 1?> = 0
Enter high limit for fl 1?> = 10000

Enter new 6 char. name of fl 2?> = val2
Enter new 18 char. descriptor of fl 2?> = value number 2
Enter low limit for fl 2?> = 0
Enter high limit for fl 2?> = 10000

Enter new 6 char. name of fl 3?> = val3
Enter new 18 char. descriptor of fl 3?> = value number 3
Enter low limit for fl 3?> = 0
Enter high limit for fl 3?> = 10000

Enter new 6 char. name of fl 4?> = val4
Enter new 18 char. descriptor of fl 4?> = value number 4
Enter low limit for fl 4?> = 0
Enter high limit for fl 4?> = 10000

Enter new 6 char. name of fl 5?> = val5
Enter new 18 char. descriptor of fl 5?> = value number 5
Enter low limit for fl 5?> = 0
Enter high limit for fl 5?> = 10000

Enter new 6 char. name of fl 6?> = val6
Enter new 18 char. descriptor of fl 6?> = value number 6
Enter low limit for fl 6?> = 0
Enter high limit for fl 6?> = 10000

Enter new 6 char. name of fl 7?> = val7
Enter new 18 char. descriptor of fl 7?> = value number 7
Enter low limit for fl 7?> = 0
Enter high limit for fl 7?> = 10000

Enter new 6 char. name of fl 8?> = val8
Enter new 18 char. descriptor of fl 8?> = value number 8
Enter low limit for fl 8?> = 0
Enter high limit for fl 8?> = 10000
```

```
Enter new 6 char. name of fl 9?> = val9
Enter new 18 char. descriptor of fl 9?> = value number 9
Enter low limit for fl 9?> = 0
Enter high limit for fl 9?> = 10000

Enter new 6 char. name of fl 10?> = val10
Enter new 18 char. descriptor of fl 10?> = value number 10
Enter low limit for fl 10?> = 0
Enter high limit for fl 10?> = 10000

Enter new 6 char. name of fl 11?> = val11
Enter new 18 char. descriptor of fl 11?> = value number 11
Enter low limit for fl 11?> = 0
Enter high limit for fl 11?> = 10000

Enter new 6 char. name of fl 12?> = val12
Enter new 18 char. descriptor of fl 12?> = value number 12
Enter low limit for fl 12?> = 0
Enter high limit for fl 12?> = 10000

Enter new 6 char. name of fl 13?> = val13
Enter new 18 char. descriptor of fl 13?> = value number 13
Enter low limit for fl 13?> = 0
Enter high limit for fl 13?> = 10000

Enter new 6 char. name of fl 14?> = val14
Enter new 18 char. descriptor of fl 14?> = value number 14
Enter low limit for fl 14?> = 0
Enter high limit for fl 14?> = 10000

Enter new 6 char. name of fl 15?> = val15
Enter new 18 char. descriptor of fl 15?> = value number 15
Enter low limit for fl 15?> = 0
Enter high limit for fl 15?> = 10000

Enter new 6 char. name of fl 16?> = val16
Enter new 18 char. descriptor of fl 16?> = value number 16
Enter low limit for fl 16?> = 0
Enter high limit for fl 16?> = 10000

Enter new 6 char. name of fl 17?> = val17
Enter new 18 char. descriptor of fl 17?> = value number 17
Enter low limit for fl 17?> = 0
Enter high limit for fl 17?> = 10000

Enter new 6 char. name of fl 18?> = val18
Enter new 18 char. descriptor of fl 18?> = value number 18
Enter low limit for fl 18?> = 0
Enter high limit for fl 18?> = 10000
```

```
Enter new 6 char. name of fl 19?> = val19
Enter new 18 char. descriptor of fl 19?> = value number 19
Enter low limit for fl 19?> = 0
Enter high limit for fl 19?> = 10000
MRPF?>  = add 35 in

Enter new 6 char. name of in 0?> = in0
Enter new 18 char. descriptor of in 0?> = integer number 0
Enter low limit for in 0?> = 0
Enter high limit for in 0?> = 7

Enter new 6 char. name of in 1?> = in1
Enter new 18 char. descriptor of in 1?> = integer number 1
Enter low limit for in 1?> = 0
Enter high limit for in 1?> = 7

Enter new 6 char. name of in 2?> = in2
Enter new 18 char. descriptor of in 2?> = integer number 2
Enter low limit for in 2?> = 0
Enter high limit for in 2?> = 7

Enter new 6 char. name of in 3?> = in3
Enter new 18 char. descriptor of in 3?> = integer number 3
Enter low limit for in 3?> = 0
Enter high limit for in 3?> = 7

Enter new 6 char. name of in 4?> = in4
Enter new 18 char. descriptor of in 4?> = integer number 4
Enter low limit for in 4?> = 0
Enter high limit for in 4?> = 7

Enter new 6 char. name of in 5?> = in5
Enter new 18 char. descriptor of in 5?> = integer number 5
Enter low limit for in 5?> = 0
Enter high limit for in 5?> = 7

Enter new 6 char. name of in 6?> = in6
Enter new 18 char. descriptor of in 6?> = integer number 6
Enter low limit for in 6?> = 0
Enter high limit for in 6?> = 7

Enter new 6 char. name of in 7?> = in7
Enter new 18 char. descriptor of in 7?> = integer number 7
Enter low limit for in 7?> = 0
Enter high limit for in 7?> = 7

Enter new 6 char. name of in 8?> = in8
Enter new 18 char. descriptor of in 8?> = integer number 8
Enter low limit for in 8?> = 0
Enter high limit for in 8?> = 7
```

```
Enter new 6 char. name of in 9?> = in9
Enter new 18 char. descriptor of in 9?> = integer number 9
Enter low limit for in 9?> = 0
Enter high limit for in 9?> = 7

Enter new 6 char. name of in 10?> = in10
Enter new 18 char. descriptor of in 10?> = integer number 10
Enter low limit for in 10?> = 0
Enter high limit for in 10?> = 7

Enter new 6 char. name of in 11?> = in11
Enter new 18 char. descriptor of in 11?> = integer number 11
Enter low limit for in 11?> = 0
Enter high limit for in 11?> = 7

Enter new 6 char. name of in 12?> = in12
Enter new 18 char. descriptor of in 12?> = integer number 12
Enter low limit for in 12?> = 0
Enter high limit for in 12?> = 7

Enter new 6 char. name of in 13?> = in13
Enter new 18 char. descriptor of in 13?> = integer number 13
Enter low limit for in 13?> = 0
Enter high limit for in 13?> = 7

Enter new 6 char. name of in 14?> = in14
Enter new 18 char. descriptor of in 14?> = integer number 14
Enter low limit for in 14?> = 0
Enter high limit for in 14?> = 7

Enter new 6 char. name of in 15?> = in15
Enter new 18 char. descriptor of in 15?> = integer number 15
Enter low limit for in 15?> = 0
Enter high limit for in 15?> = 7

Enter new 6 char. name of in 16?> = in16
Enter new 18 char. descriptor of in 16?> = integer number 16
Enter low limit for in 16?> = 0
Enter high limit for in 16?> = 7

Enter new 6 char. name of in 17?> = in17
Enter new 18 char. descriptor of in 17?> = integer number 17
Enter low limit for in 17?> = 0
Enter high limit for in 17?> = 7

Enter new 6 char. name of in 18?> = in18
Enter new 18 char. descriptor of in 18?> = integer number 18
Enter low limit for in 18?> = 0
Enter high limit for in 18?> = 7
```

```
Enter new 6 char. name of in 19?> = in19
Enter new 18 char. descriptor of in 19?> = integer number 19
Enter low limit for in 19?> = 0
Enter high limit for in 19?> = 7

Enter new 6 char. name of in 20?> = in20
Enter new 18 char. descriptor of in 20?> = integer number 20
Enter low limit for in 20?> = 0
Enter high limit for in 20?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 21?> = in21
Enter new 18 char. descriptor of in 21?> = integer number 21
Enter low limit for in 21?> = 0
Enter high limit for in 21?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 22?> = in22
Enter new 18 char. descriptor of in 22?> = integer number 22
Enter low limit for in 22?> = 0
Enter high limit for in 22?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 23?> = in23
Enter new 18 char. descriptor of in 23?> = integer number 23
Enter low limit for in 23?> = 0
Enter high limit for in 23?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 24?> = in24
Enter new 18 char. descriptor of in 24?> = integer number 24
Enter low limit for in 24?> = 0
Enter high limit for in 24?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 25?> = imisc1
Enter new 18 char. descriptor of in 25?> = misc1 val1
Enter low limit for in 25?> = 0
Enter high limit for in 25?> = 128

Enter new 6 char. name of in 26?> = imisc2
Enter new 18 char. descriptor of in 26?> = misc2 val2
Enter low limit for in 26?> = 0
Enter high limit for in 26?> = 128

Enter new 6 char. name of in 27?> = adi
Enter new 18 char. descriptor of in 27?> = alm des index
Enter low limit for in 27?> = 0
Enter high limit for in 27?> = 80
```

```
Enter new 6 char. name of in 28?> = lmpvar
Enter new 18 char. descriptor of in 28?> = lamp var
Enter low limit for in 28?> = 0
Enter high limit for in 28?> = 4

Enter new 6 char. name of in 29?> = led0
Enter new 18 char. descriptor of in 29?> = led 0
Enter low limit for in 29?> = 0
Enter high limit for in 29?> = 63

Enter new 6 char. name of in 30?> = led1
Enter new 18 char. descriptor of in 30?> = led 1
Enter low limit for in 30?> = 0
Enter high limit for in 30?> = 63

Enter new 6 char. name of in 31?> = led2
Enter new 18 char. descriptor of in 31?> = led 2
Enter low limit for in 31?> = 0
Enter high limit for in 31?> = 63

Enter new 6 char. name of in 32?> = ispar0
Enter new 18 char. descriptor of in 32?> = spare int 0
Enter low limit for in 32?> = 0
Enter high limit for in 32?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 33?> = ispar1
Enter new 18 char. descriptor of in 33?> = spare int 1
Enter low limit for in 33?> = 0
Enter high limit for in 33?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 34?> = ispar2
Enter new 18 char. descriptor of in 34?> = spare int 2
Enter low limit for in 34?> = 0
Enter high limit for in 34?> = 37000
High limit is less than low - setting high to 32767
MRPF?>  = add 5 st

Enter new 6 char. name of st 0?> = stmr0
Enter new 18 char. descriptor of st 0?> = second timer 0
Enter low limit for st 0?> = 0
Enter high limit for st 0?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of st 1?> = stmr1
Enter new 18 char. descriptor of st 1?> = second timer 1
Enter low limit for st 1?> = 0
Enter high limit for st 1?> = 37000
High limit is less than low - setting high to 32767
```

```
Enter new 6 char. name of st 2?> = stmr2
Enter new 18 char. descriptor of st 2?> = second timer 2
Enter low limit for st 2?> = 0
Enter high limit for st 2?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of st 3?> = stmr3
Enter new 18 char. descriptor of st 3?> = second timer 3
Enter low limit for st 3?> = 0
Enter high limit for st 3?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of st 4?> = stmr4
Enter new 18 char. descriptor of st 4?> = second timer 4
Enter low limit for st 4?> = 0
Enter high limit for st 4?> = 37000
High limit is less than low - setting high to 32767
MRPF?>  = add 5 mt

Enter new 6 char. name of mt 0?> = mtmr0
Enter new 18 char. descriptor of mt 0?> = minute timer 0
Enter low limit for mt 0?> = 0
Enter high limit for mt 0?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of mt 1?> = mtmr1
Enter new 18 char. descriptor of mt 1?> = minute timer 1
Enter low limit for mt 1?> = 0
Enter high limit for mt 1?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of mt 2?> = mtmr2
Enter new 18 char. descriptor of mt 2?> = minute timer 2
Enter low limit for mt 2?> = 0
Enter high limit for mt 2?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of mt 3?> = mtmr3
Enter new 18 char. descriptor of mt 3?> = minute timer 3
Enter low limit for mt 3?> = 0
Enter high limit for mt 3?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of mt 4?> = mtmr4
Enter new 18 char. descriptor of mt 4?> = minute timer 4
Enter low limit for mt 4?> = 0
Enter high limit for mt 4?> = 37000
High limit is less than low - setting high to 32767
MRPF?>  = add 5 sg
```

```
Enter new 6 char. name of sg 0?> = note0
Enter new 18 char. descriptor of sg 0?> = recipe string 0
Enter maximum size for sg 0?> = 20

Enter new 6 char. name of sg 1?> = note1
Enter new 18 char. descriptor of sg 1?> = recipe string 1
Enter maximum size for sg 1?> = 20

Enter new 6 char. name of sg 2?> = note2
Enter new 18 char. descriptor of sg 2?> = recipe string 2
Enter maximum size for sg 2?> = 20

Enter new 6 char. name of sg 3?> = note3
Enter new 18 char. descriptor of sg 3?> = recipe string 3
Enter maximum size for sg 3?> = 20

Enter new 6 char. name of sg 4?> = note4
Enter new 18 char. descriptor of sg 4?> = recipe string 4
Enter maximum size for sg 4?> = 20
MRPF?>
 end of file encountered
RPF?>
 end of file encountered
RMS?>
 end of file encountered
```

EMTY

```
RMS Version 6.3-2                    05-Dec-94 09:32:42
RMS?> = rpf de emty
RPF?> = rpf de emty
Invalid Command
Open error file SV0:[300,1]U.UPF ,error = -1
RPF?> = cr emty,*
MRPF?>  = add 25 fl

Enter new 6 char. name of fl 0?> = val0
Enter new 18 char. descriptor of fl 0?> = value number 0
Enter low limit for fl 0?> = 0
Enter high limit for fl 0?> = 10000

Enter new 6 char. name of fl 1?> = val1
Enter new 18 char. descriptor of fl 1?> = value number 1
Enter low limit for fl 1?> = 0
Enter high limit for fl 1?> = 10000

Enter new 6 char. name of fl 2?> = val2
Enter new 18 char. descriptor of fl 2?> = value number 2
Enter low limit for fl 2?> = 0
Enter high limit for fl 2?> = 10000

Enter new 6 char. name of fl 3?> = val3
Enter new 18 char. descriptor of fl 3?> = value number 3
Enter low limit for fl 3?> = 0
Enter high limit for fl 3?> = 10000

Enter new 6 char. name of fl 4?> = val4
Enter new 18 char. descriptor of fl 4?> = value number 4
Enter low limit for fl 4?> = 0
Enter high limit for fl 4?> = 10000

Enter new 6 char. name of fl 5?> = val5
Enter new 18 char. descriptor of fl 5?> = value number 5
Enter low limit for fl 5?> = 0
Enter high limit for fl 5?> = 10000

Enter new 6 char. name of fl 6?> = val6
Enter new 18 char. descriptor of fl 6?> = value number 6
Enter low limit for fl 6?> = 0
Enter high limit for fl 6?> = 10000

Enter new 6 char. name of fl 7?> = val7
Enter new 18 char. descriptor of fl 7?> = value number 7
Enter low limit for fl 7?> = 0
Enter high limit for fl 7?> = 10000
```

```
Enter new 6 char. name of fl 8?> = val8
Enter new 18 char. descriptor of fl 8?> = value number 8
Enter low limit for fl 8?> = 0
Enter high limit for fl 8?> = 10000

Enter new 6 char. name of fl 9?> = val9
Enter new 18 char. descriptor of fl 9?> = value number 9
Enter low limit for fl 9?> = 0
Enter high limit for fl 9?> = 10000

Enter new 6 char. name of fl 10?> = val10
Enter new 18 char. descriptor of fl 10?> = value number 10
Enter low limit for fl 10?> = 0
Enter high limit for fl 10?> = 10000

Enter new 6 char. name of fl 11?> = val11
Enter new 18 char. descriptor of fl 11?> = value number 11
Enter low limit for fl 11?> = 0
Enter high limit for fl 11?> = 10000

Enter new 6 char. name of fl 12?> = val12
Enter new 18 char. descriptor of fl 12?> = value number 12
Enter low limit for fl 12?> = 0
Enter high limit for fl 12?> = 10000

Enter new 6 char. name of fl 13?> = val13
Enter new 18 char. descriptor of fl 13?> = value number 13
Enter low limit for fl 13?> = 0
Enter high limit for fl 13?> = 10000

Enter new 6 char. name of fl 14?> = val14
Enter new 18 char. descriptor of fl 14?> = value number 14
Enter low limit for fl 14?> = 0
Enter high limit for fl 14?> = 10000

Enter new 6 char. name of fl 15?> = val15
Enter new 18 char. descriptor of fl 15?> = value number 15
Enter low limit for fl 15?> = 0
Enter high limit for fl 15?> = 10000

Enter new 6 char. name of fl 16?> = val16
Enter new 18 char. descriptor of fl 16?> = value number 16
Enter low limit for fl 16?> = 0
Enter high limit for fl 16?> = 10000

Enter new 6 char. name of fl 17?> = val17
Enter new 18 char. descriptor of fl 17?> = value number 17
Enter low limit for fl 17?> = 0
Enter high limit for fl 17?> = 10000
```

```
Enter new 6 char. name of fl 18?> = val18
Enter new 18 char. descriptor of fl 18?> = value number 18
Enter low limit for fl 18?> = 0
Enter high limit for fl 18?> = 10000

Enter new 6 char. name of fl 19?> = val19
Enter new 18 char. descriptor of fl 19?> = value number 19
Enter low limit for fl 19?> = 0
Enter high limit for fl 19?> = 10000

Enter new 6 char. name of fl 20?> = val20
Enter new 18 char. descriptor of fl 20?> = value number 20
Enter low limit for fl 20?> = 0
Enter high limit for fl 20?> = 10000

Enter new 6 char. name of fl 21?> = val21
Enter new 18 char. descriptor of fl 21?> = value number 21
Enter low limit for fl 21?> = 0
Enter high limit for fl 21?> = 10000

Enter new 6 char. name of fl 22?> = val22
Enter new 18 char. descriptor of fl 22?> = value number 22
Enter low limit for fl 22?> = 0
Enter high limit for fl 22?> = 10000

Enter new 6 char. name of fl 23?> = val23
Enter new 18 char. descriptor of fl 23?> = value number 23
Enter low limit for fl 23?> = 0
Enter high limit for fl 23?> = 10000

Enter new 6 char. name of fl 24?> = val24
Enter new 18 char. descriptor of fl 24?> = value number 24
Enter low limit for fl 24?> = 0
Enter high limit for fl 24?> = 10000
MRPF?>  = add 50 in

Enter new 6 char. name of in 0?> = in0
Enter new 18 char. descriptor of in 0?> = integer number 0
Enter low limit for in 0?> = 0
Enter high limit for in 0?> = 7

Enter new 6 char. name of in 1?> = in1
Enter new 18 char. descriptor of in 1?> = integer number 1
Enter low limit for in 1?> = 0
Enter high limit for in 1?> = 7

Enter new 6 char. name of in 2?> = in2
Enter new 18 char. descriptor of in 2?> = integer number 2
Enter low limit for in 2?> = 0
Enter high limit for in 2?> = 7
```

```
Enter new 6 char. name of in 3?> = in3
Enter new 18 char. descriptor of in 3?> = integer number 3
Enter low limit for in 3?> = 0
Enter high limit for in 3?> = 7

Enter new 6 char. name of in 4?> = in4
Enter new 18 char. descriptor of in 4?> = integer number 4
Enter low limit for in 4?> = 0
Enter high limit for in 4?> = 7

Enter new 6 char. name of in 5?> = in5
Enter new 18 char. descriptor of in 5?> = integer number 5
Enter low limit for in 5?> = 0
Enter high limit for in 5?> = 7

Enter new 6 char. name of in 6?> = in6
Enter new 18 char. descriptor of in 6?> = integer number 6
Enter low limit for in 6?> = 0
Enter high limit for in 6?> = 7

Enter new 6 char. name of in 7?> = in7
Enter new 18 char. descriptor of in 7?> = integer number 7
Enter low limit for in 7?> = 0
Enter high limit for in 7?> = 7

Enter new 6 char. name of in 8?> = in8
Enter new 18 char. descriptor of in 8?> = integer number 8
Enter low limit for in 8?> = 0
Enter high limit for in 8?> = 7

Enter new 6 char. name of in 9?> = in9
Enter new 18 char. descriptor of in 9?> = integer number 9
Enter low limit for in 9?> = 0
Enter high limit for in 9?> = 7

Enter new 6 char. name of in 10?> = in10
Enter new 18 char. descriptor of in 10?> = integer number 10
Enter low limit for in 10?> = 0
Enter high limit for in 10?> = 7

Enter new 6 char. name of in 11?> = in11
Enter new 18 char. descriptor of in 11?> = integer number 11
Enter low limit for in 11?> = 0
Enter high limit for in 11?> = 7

Enter new 6 char. name of in 12?> = in12
Enter new 18 char. descriptor of in 12?> = integer number 12
Enter low limit for in 12?> = 0
Enter high limit for in 12?> = 7
```

```
Enter new 6 char. name of in 13?> = in13
Enter new 18 char. descriptor of in 13?> = integer number 13
Enter low limit for in 13?> = 0
Enter high limit for in 13?> = 7

Enter new 6 char. name of in 14?> = in14
Enter new 18 char. descriptor of in 14?> = integer number 14
Enter low limit for in 14?> = 0
Enter high limit for in 14?> = 7

Enter new 6 char. name of in 15?> = in15
Enter new 18 char. descriptor of in 15?> = integer number 15
Enter low limit for in 15?> = 0
Enter high limit for in 15?> = 7

Enter new 6 char. name of in 16?> = in16
Enter new 18 char. descriptor of in 16?> = integer number 16
Enter low limit for in 16?> = 0
Enter high limit for in 16?> = 7

Enter new 6 char. name of in 17?> = in17
Enter new 18 char. descriptor of in 17?> = integer number 17
Enter low limit for in 17?> = 0
Enter high limit for in 17?> = 7

Enter new 6 char. name of in 18?> = in18
Enter new 18 char. descriptor of in 18?> = integer number 18
Enter low limit for in 18?> = 0
Enter high limit for in 18?> = 7

Enter new 6 char. name of in 19?> = in19
Enter new 18 char. descriptor of in 19?> = integer number 19
Enter low limit for in 19?> = 0
Enter high limit for in 19?> = 7

Enter new 6 char. name of in 20?> = in20
Enter new 18 char. descriptor of in 20?> = integer number 20
Enter low limit for in 20?> = 0
Enter high limit for in 20?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 21?> = in21
Enter new 18 char. descriptor of in 21?> = integer number 21
Enter low limit for in 21?> = 0
Enter high limit for in 21?> = 100000
High limit is less than low - setting high to 32767
```

```
Enter new 6 char. name of in 22?> = in22
Enter new 18 char. descriptor of in 22?> = integer number 22
Enter low limit for in 22?> = 0
Enter high limit for in 22?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 23?> = in23
Enter new 18 char. descriptor of in 23?> = integer number 23
Enter low limit for in 23?> = 0
Enter high limit for in 23?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 24?> = in24
Enter new 18 char. descriptor of in 24?> = integer number 24
Enter low limit for in 24?> = 0
Enter high limit for in 24?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 25?> = in25
Enter new 18 char. descriptor of in 25?> = integer number 25
Enter low limit for in 25?> = 0
Enter high limit for in 25?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 26?> = in26
Enter new 18 char. descriptor of in 26?> = integer number 26
Enter low limit for in 26?> = 0
Enter high limit for in 26?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 27?> = in27
Enter new 18 char. descriptor of in 27?> = integer number 27
Enter low limit for in 27?> = 0
Enter high limit for in 27?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 28?> = in28
Enter new 18 char. descriptor of in 28?> = integer number 28
Enter low limit for in 28?> = 0
Enter high limit for in 28?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 29?> = in29
Enter new 18 char. descriptor of in 29?> = integer number 29
Enter low limit for in 29?> = 0
Enter high limit for in 29?> = 100000
High limit is less than low - setting high to 32767
```

```
Enter new 6 char. name of in 30?> = in30
Enter new 18 char. descriptor of in 30?> = integer number 30
Enter low limit for in 30?> = 0
Enter high limit for in 30?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 31?> = in31
Enter new 18 char. descriptor of in 31?> = integer number 31
Enter low limit for in 31?> = 0
Enter high limit for in 31?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 32?> = in32
Enter new 18 char. descriptor of in 32?> = integer number 32
Enter low limit for in 32?> = 0
Enter high limit for in 32?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 33?> = in33
Enter new 18 char. descriptor of in 33?> = integer number 33
Enter low limit for in 33?> = 0
Enter high limit for in 33?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 34?> = in34
Enter new 18 char. descriptor of in 34?> = integer number 34
Enter low limit for in 34?> = 0
Enter high limit for in 34?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 35?> = in35
Enter new 18 char. descriptor of in 35?> = integer number 35
Enter low limit for in 35?> = 0
Enter high limit for in 35?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 36?> = in36
Enter new 18 char. descriptor of in 36?> = integer number 36
Enter low limit for in 36?> = 0
Enter high limit for in 36?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 37?> = in37
Enter new 18 char. descriptor of in 37?> = integer number 37
Enter low limit for in 37?> = 0
Enter high limit for in 37?> = 100000
High limit is less than low - setting high to 32767
```

```
Enter new 6 char. name of in 38?> = in38
Enter new 18 char. descriptor of in 38?> = integer number 38
Enter low limit for in 38?> = 0
Enter high limit for in 38?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 39?> = in39
Enter new 18 char. descriptor of in 39?> = integer number 39
Enter low limit for in 39?> = 0
Enter high limit for in 39?> = 100000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 40?> = imisc1
Enter new 18 char. descriptor of in 40?> = misc1 val1
Enter low limit for in 40?> = 0
Enter high limit for in 40?> = 128

Enter new 6 char. name of in 41?> = imisc2
Enter new 18 char. descriptor of in 41?> = misc2 val2
Enter low limit for in 41?> = 0
Enter high limit for in 41?> = 128

Enter new 6 char. name of in 42?> = adi
Enter new 18 char. descriptor of in 42?> = alm des index
Enter low limit for in 42?> = 0
Enter high limit for in 42?> = 80

Enter new 6 char. name of in 43?> = lmpvar
Enter new 18 char. descriptor of in 43?> = lamp var
Enter low limit for in 43?> = 0
Enter high limit for in 43?> = 4

Enter new 6 char. name of in 44?> = led0
Enter new 18 char. descriptor of in 44?> = led 0
Enter low limit for in 44?> = 0
Enter high limit for in 44?> = 63

Enter new 6 char. name of in 45?> = led1
Enter new 18 char. descriptor of in 45?> = led 1
Enter low limit for in 45?> = 0
Enter high limit for in 45?> = 63

Enter new 6 char. name of in 46?> = led2
Enter new 18 char. descriptor of in 46?> = led 2
Enter low limit for in 46?> = 0
Enter high limit for in 46?> = 63
```

```
Enter new 6 char. name of in 47?> = ispar0
Enter new 18 char. descriptor of in 47?> = spare int 0
Enter low limit for in 47?> = 0
Enter high limit for in 47?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 48?> = ispar1
Enter new 18 char. descriptor of in 48?> = spare int 1
Enter low limit for in 48?> = 0
Enter high limit for in 48?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of in 49?> = ispar2
Enter new 18 char. descriptor of in 49?> = spare int 2
Enter low limit for in 49?> = 0
Enter high limit for in 49?> = 37000
High limit is less than low - setting high to 32767
MRPF?>  = add 5 st

Enter new 6 char. name of st 0?> = stmr0
Enter new 18 char. descriptor of st 0?> = second timer 0
Enter low limit for st 0?> = 0
Enter high limit for st 0?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of st 1?> = stmr1
Enter new 18 char. descriptor of st 1?> = second timer 1
Enter low limit for st 1?> = 0
Enter high limit for st 1?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of st 2?> = stmr2
Enter new 18 char. descriptor of st 2?> = second timer 2
Enter low limit for st 2?> = 0
Enter high limit for st 2?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of st 3?> = stmr3
Enter new 18 char. descriptor of st 3?> = second timer 3
Enter low limit for st 3?> = 0
Enter high limit for st 3?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of st 4?> = stmr4
Enter new 18 char. descriptor of st 4?> = second timer 4
Enter low limit for st 4?> = 0
Enter high limit for st 4?> = 37000
High limit is less than low - setting high to 32767
MRPF?>  = add 5 mt
```

```
Enter new 6 char. name of mt 0?> = mtmr0
Enter new 18 char. descriptor of mt 0?> = minute timer 0
Enter low limit for mt 0?> = 0
Enter high limit for mt 0?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of mt 1?> = mtmr1
Enter new 18 char. descriptor of mt 1?> = minute timer 1
Enter low limit for mt 1?> = 0
Enter high limit for mt 1?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of mt 2?> = mtmr2
Enter new 18 char. descriptor of mt 2?> = minute timer 2
Enter low limit for mt 2?> = 0
Enter high limit for mt 2?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of mt 3?> = mtmr3
Enter new 18 char. descriptor of mt 3?> = minute timer 3
Enter low limit for mt 3?> = 0
Enter high limit for mt 3?> = 37000
High limit is less than low - setting high to 32767

Enter new 6 char. name of mt 4?> = mtmr4
Enter new 18 char. descriptor of mt 4?> = minute timer 4
Enter low limit for mt 4?> = 0
Enter high limit for mt 4?> = 37000
High limit is less than low - setting high to 32767
MRPF?>
 end of file encountered
RPF?>
 end of file encountered
RMS?>
 end of file encountered
```

APPENDIX D - UNIT PROTOTYPE FILES (UPF)

TOTALO

```
UPF Version 6.3-2       05-Dec-94 09:33:27
UPF?>  = de total0
UPF?>  = cr total0,*
Enter 4 character name of RPF to associate?> = all0
MUPF?>  = add 432 fg
Enter new 6 char. name of fg 0 ?> = fai0
Enter new 6 char. name of fg 1 ?> = fai1
Enter new 6 char. name of fg 2 ?> = fai2
Enter new 6 char. name of fg 3 ?> = fai3
Enter new 6 char. name of fg 4 ?> = fai4
Enter new 6 char. name of fg 5 ?> = fai5
Enter new 6 char. name of fg 6 ?> = fai6
Enter new 6 char. name of fg 7 ?> = fai7
Enter new 6 char. name of fg 8 ?> = fai8
Enter new 6 char. name of fg 9 ?> = fai9
Enter new 6 char. name of fg 10 ?> = fai10
Enter new 6 char. name of fg 11 ?> = fai11
Enter new 6 char. name of fg 12 ?> = fai12
Enter new 6 char. name of fg 13 ?> = fai13
Enter new 6 char. name of fg 14 ?> = fai14
Enter new 6 char. name of fg 15 ?> = fai15
Enter new 6 char. name of fg 16 ?> = fai16
Enter new 6 char. name of fg 17 ?> = fai17
Enter new 6 char. name of fg 18 ?> = fai18
Enter new 6 char. name of fg 19 ?> = fai19
Enter new 6 char. name of fg 20 ?> = fai20
Enter new 6 char. name of fg 21 ?> = fai21
Enter new 6 char. name of fg 22 ?> = fai22
Enter new 6 char. name of fg 23 ?> = fai23
Enter new 6 char. name of fg 24 ?> = fai24
Enter new 6 char. name of fg 25 ?> = fai25
Enter new 6 char. name of fg 26 ?> = fai26
Enter new 6 char. name of fg 27 ?> = fai27
Enter new 6 char. name of fg 28 ?> = fai28
Enter new 6 char. name of fg 29 ?> = fai29
Enter new 6 char. name of fg 30 ?> = fai30
Enter new 6 char. name of fg 31 ?> = fai31
Enter new 6 char. name of fg 32 ?> = fai32
Enter new 6 char. name of fg 33 ?> = fai33
Enter new 6 char. name of fg 34 ?> = fai34
Enter new 6 char. name of fg 35 ?> = fai35
Enter new 6 char. name of fg 36 ?> = fai36
Enter new 6 char. name of fg 37 ?> = fai37
Enter new 6 char. name of fg 38 ?> = fai38
Enter new 6 char. name of fg 39 ?> = fai39
Enter new 6 char. name of fg 40 ?> = fai40
Enter new 6 char. name of fg 41 ?> = fai41
Enter new 6 char. name of fg 42 ?> = fai42
Enter new 6 char. name of fg 43 ?> = fai43
Enter new 6 char. name of fg 44 ?> = fai44
```

```
Enter new 6 char. name of fg 45 ?> = fai45
Enter new 6 char. name of fg 46 ?> = fai46
Enter new 6 char. name of fg 47 ?> = fai47
Enter new 6 char. name of fg 48 ?> = fai48
Enter new 6 char. name of fg 49 ?> = fai49
Enter new 6 char. name of fg 50 ?> = fai50
Enter new 6 char. name of fg 51 ?> = fai51
Enter new 6 char. name of fg 52 ?> = fai52
Enter new 6 char. name of fg 53 ?> = fai53
Enter new 6 char. name of fg 54 ?> = fai54
Enter new 6 char. name of fg 55 ?> = fai55
Enter new 6 char. name of fg 56 ?> = fai56
Enter new 6 char. name of fg 57 ?> = fai57
Enter new 6 char. name of fg 58 ?> = fai58
Enter new 6 char. name of fg 59 ?> = fai59
Enter new 6 char. name of fg 60 ?> = fai60
Enter new 6 char. name of fg 61 ?> = fai61
Enter new 6 char. name of fg 62 ?> = fai62
Enter new 6 char. name of fg 63 ?> = fai63
Enter new 6 char. name of fg 64 ?> = fai64
Enter new 6 char. name of fg 65 ?> = fai65
Enter new 6 char. name of fg 66 ?> = fai66
Enter new 6 char. name of fg 67 ?> = fai67
Enter new 6 char. name of fg 68 ?> = fai68
Enter new 6 char. name of fg 69 ?> = fai69
Enter new 6 char. name of fg 70 ?> = fai70
Enter new 6 char. name of fg 71 ?> = fai71
Enter new 6 char. name of fg 72 ?> = fai72
Enter new 6 char. name of fg 73 ?> = fai73
Enter new 6 char. name of fg 74 ?> = fai74
Enter new 6 char. name of fg 75 ?> = fai75
Enter new 6 char. name of fg 76 ?> = fai76
Enter new 6 char. name of fg 77 ?> = fai77
Enter new 6 char. name of fg 78 ?> = fai78
Enter new 6 char. name of fg 79 ?> = fai79
Enter new 6 char. name of fg 80 ?> = fai80
Enter new 6 char. name of fg 81 ?> = fai81
Enter new 6 char. name of fg 82 ?> = fai82
Enter new 6 char. name of fg 83 ?> = fai83
Enter new 6 char. name of fg 84 ?> = fai84
Enter new 6 char. name of fg 85 ?> = fai85
Enter new 6 char. name of fg 86 ?> = fai86
Enter new 6 char. name of fg 87 ?> = fai87
Enter new 6 char. name of fg 88 ?> = fai88
Enter new 6 char. name of fg 89 ?> = fai89
Enter new 6 char. name of fg 90 ?> = fai90
Enter new 6 char. name of fg 91 ?> = fai91
Enter new 6 char. name of fg 92 ?> = fai92
Enter new 6 char. name of fg 93 ?> = fai93
Enter new 6 char. name of fg 94 ?> = fai94
```

```
Enter new 6 char. name of fg 95 ?> = fai95
Enter new 6 char. name of fg 96 ?> = fai96
Enter new 6 char. name of fg 97 ?> = fai97
Enter new 6 char. name of fg 98 ?> = fai98
Enter new 6 char. name of fg 99 ?> = fai99
Enter new 6 char. name of fg 100 ?> = fai100
Enter new 6 char. name of fg 101 ?> = fai101
Enter new 6 char. name of fg 102 ?> = fai102
Enter new 6 char. name of fg 103 ?> = fai103
Enter new 6 char. name of fg 104 ?> = fai104
Enter new 6 char. name of fg 105 ?> = fai105
Enter new 6 char. name of fg 106 ?> = fai106
Enter new 6 char. name of fg 107 ?> = fai107
Enter new 6 char. name of fg 108 ?> = fai108
Enter new 6 char. name of fg 109 ?> = fai109
Enter new 6 char. name of fg 110 ?> = fai110
Enter new 6 char. name of fg 111 ?> = fai111
Enter new 6 char. name of fg 112 ?> = fai112
Enter new 6 char. name of fg 113 ?> = fai113
Enter new 6 char. name of fg 114 ?> = fai114
Enter new 6 char. name of fg 115 ?> = fai115
Enter new 6 char. name of fg 116 ?> = fai116
Enter new 6 char. name of fg 117 ?> = fai117
Enter new 6 char. name of fg 118 ?> = fai118
Enter new 6 char. name of fg 119 ?> = fai119
Enter new 6 char. name of fg 120 ?> = fai120
Enter new 6 char. name of fg 121 ?> = fai121
Enter new 6 char. name of fg 122 ?> = fai122
Enter new 6 char. name of fg 123 ?> = fai123
Enter new 6 char. name of fg 124 ?> = fai124
Enter new 6 char. name of fg 125 ?> = fai125
Enter new 6 char. name of fg 126 ?> = fai126
Enter new 6 char. name of fg 127 ?> = fai127
Enter new 6 char. name of fg 128 ?> = fdi0
Enter new 6 char. name of fg 129 ?> = fdi1
Enter new 6 char. name of fg 130 ?> = fdi2
Enter new 6 char. name of fg 131 ?> = fdi3
Enter new 6 char. name of fg 132 ?> = fdi4
Enter new 6 char. name of fg 133 ?> = fdi5
Enter new 6 char. name of fg 134 ?> = fdi6
Enter new 6 char. name of fg 135 ?> = fdi7
Enter new 6 char. name of fg 136 ?> = fdi8
Enter new 6 char. name of fg 137 ?> = fdi9
Enter new 6 char. name of fg 138 ?> = fdi10
Enter new 6 char. name of fg 139 ?> = fdi11
Enter new 6 char. name of fg 140 ?> = fdi12
Enter new 6 char. name of fg 141 ?> = fdi13
Enter new 6 char. name of fg 142 ?> = fdi14
Enter new 6 char. name of fg 143 ?> = fdi15
Enter new 6 char. name of fg 144 ?> = fdi16
```

```
Enter new 6 char. name of fg 145 ?> = fdi17
Enter new 6 char. name of fg 146 ?> = fdi18
Enter new 6 char. name of fg 147 ?> = fdi19
Enter new 6 char. name of fg 148 ?> = fdi20
Enter new 6 char. name of fg 149 ?> = fdi21
Enter new 6 char. name of fg 150 ?> = fdi22
Enter new 6 char. name of fg 151 ?> = fdi23
Enter new 6 char. name of fg 152 ?> = fdi24
Enter new 6 char. name of fg 153 ?> = fdi25
Enter new 6 char. name of fg 154 ?> = fdi26
Enter new 6 char. name of fg 155 ?> = fdi27
Enter new 6 char. name of fg 156 ?> = fdi28
Enter new 6 char. name of fg 157 ?> = fdi29
Enter new 6 char. name of fg 158 ?> = fdi30
Enter new 6 char. name of fg 159 ?> = fdi31
Enter new 6 char. name of fg 160 ?> = fdi32
Enter new 6 char. name of fg 161 ?> = fdi33
Enter new 6 char. name of fg 162 ?> = fdi34
Enter new 6 char. name of fg 163 ?> = fdi35
Enter new 6 char. name of fg 164 ?> = fdi36
Enter new 6 char. name of fg 165 ?> = fdi37
Enter new 6 char. name of fg 166 ?> = fdi38
Enter new 6 char. name of fg 167 ?> = fdi39
Enter new 6 char. name of fg 168 ?> = fdi40
Enter new 6 char. name of fg 169 ?> = fdi41
Enter new 6 char. name of fg 170 ?> = fdi42
Enter new 6 char. name of fg 171 ?> = fdi43
Enter new 6 char. name of fg 172 ?> = fdi44
Enter new 6 char. name of fg 173 ?> = fdi45
Enter new 6 char. name of fg 174 ?> = fdi46
Enter new 6 char. name of fg 175 ?> = fdi47
Enter new 6 char. name of fg 176 ?> = fdi48
Enter new 6 char. name of fg 177 ?> = fdi49
Enter new 6 char. name of fg 178 ?> = fdi50
Enter new 6 char. name of fg 179 ?> = fdi51
Enter new 6 char. name of fg 180 ?> = fdi52
Enter new 6 char. name of fg 181 ?> = fdi53
Enter new 6 char. name of fg 182 ?> = fdi54
Enter new 6 char. name of fg 183 ?> = fdi55
Enter new 6 char. name of fg 184 ?> = fdi56
Enter new 6 char. name of fg 185 ?> = fdi57
Enter new 6 char. name of fg 186 ?> = fdi58
Enter new 6 char. name of fg 187 ?> = fdi59
Enter new 6 char. name of fg 188 ?> = fdi60
Enter new 6 char. name of fg 189 ?> = fdi61
Enter new 6 char. name of fg 190 ?> = fdi62
Enter new 6 char. name of fg 191 ?> = fdi63
Enter new 6 char. name of fg 192 ?> = fdi64
Enter new 6 char. name of fg 193 ?> = fdi65
Enter new 6 char. name of fg 194 ?> = fdi66
```

```
Enter new 6 char. name of fg 195 ?> = fdi67
Enter new 6 char. name of fg 196 ?> = fdi68
Enter new 6 char. name of fg 197 ?> = fdi69
Enter new 6 char. name of fg 198 ?> = fdi70
Enter new 6 char. name of fg 199 ?> = fdi71
Enter new 6 char. name of fg 200 ?> = fdi72
Enter new 6 char. name of fg 201 ?> = fdi73
Enter new 6 char. name of fg 202 ?> = fdi74
Enter new 6 char. name of fg 203 ?> = fdi75
Enter new 6 char. name of fg 204 ?> = fdi76
Enter new 6 char. name of fg 205 ?> = fdi77
Enter new 6 char. name of fg 206 ?> = fdi78
Enter new 6 char. name of fg 207 ?> = fdi79
Enter new 6 char. name of fg 208 ?> = fdi80
Enter new 6 char. name of fg 209 ?> = fdi81
Enter new 6 char. name of fg 210 ?> = fdi82
Enter new 6 char. name of fg 211 ?> = fdi83
Enter new 6 char. name of fg 212 ?> = fdi84
Enter new 6 char. name of fg 213 ?> = fdi85
Enter new 6 char. name of fg 214 ?> = fdi86
Enter new 6 char. name of fg 215 ?> = fdi87
Enter new 6 char. name of fg 216 ?> = fdi88
Enter new 6 char. name of fg 217 ?> = fdi89
Enter new 6 char. name of fg 218 ?> = fdi90
Enter new 6 char. name of fg 219 ?> = fdi91
Enter new 6 char. name of fg 220 ?> = fdi92
Enter new 6 char. name of fg 221 ?> = fdi93
Enter new 6 char. name of fg 222 ?> = fdi94
Enter new 6 char. name of fg 223 ?> = fdi95
Enter new 6 char. name of fg 224 ?> = fdi96
Enter new 6 char. name of fg 225 ?> = fdi97
Enter new 6 char. name of fg 226 ?> = fdi98
Enter new 6 char. name of fg 227 ?> = fdi99
Enter new 6 char. name of fg 228 ?> = fdi100
Enter new 6 char. name of fg 229 ?> = fdi101
Enter new 6 char. name of fg 230 ?> = fdi102
Enter new 6 char. name of fg 231 ?> = fdi103
Enter new 6 char. name of fg 232 ?> = fdi104
Enter new 6 char. name of fg 233 ?> = fdi105
Enter new 6 char. name of fg 234 ?> = fdi106
Enter new 6 char. name of fg 235 ?> = fdi107
Enter new 6 char. name of fg 236 ?> = fdi108
Enter new 6 char. name of fg 237 ?> = fdi109
Enter new 6 char. name of fg 238 ?> = fdi110
Enter new 6 char. name of fg 239 ?> = fdi111
Enter new 6 char. name of fg 240 ?> = fdi112
Enter new 6 char. name of fg 241 ?> = fdi113
Enter new 6 char. name of fg 242 ?> = fdi114
Enter new 6 char. name of fg 243 ?> = fdi115
Enter new 6 char. name of fg 244 ?> = fdi116
```

```
Enter new 6 char. name of fg 245 ?> = fdil17
Enter new 6 char. name of fg 246 ?> = fdil18
Enter new 6 char. name of fg 247 ?> = fdil19
Enter new 6 char. name of fg 248 ?> = fdil20
Enter new 6 char. name of fg 249 ?> = fdil21
Enter new 6 char. name of fg 250 ?> = fdil22
Enter new 6 char. name of fg 251 ?> = fdil23
Enter new 6 char. name of fg 252 ?> = fdil24
Enter new 6 char. name of fg 253 ?> = fdil25
Enter new 6 char. name of fg 254 ?> = fdil26
Enter new 6 char. name of fg 255 ?> = fdil27
Enter new 6 char. name of fg 256 ?> = fdv0
Enter new 6 char. name of fg 257 ?> = fdv1
Enter new 6 char. name of fg 258 ?> = fdv2
Enter new 6 char. name of fg 259 ?> = fdv3
Enter new 6 char. name of fg 260 ?> = fdv4
Enter new 6 char. name of fg 261 ?> = fdv5
Enter new 6 char. name of fg 262 ?> = fdv6
Enter new 6 char. name of fg 263 ?> = fdv7
Enter new 6 char. name of fg 264 ?> = fdv8
Enter new 6 char. name of fg 265 ?> = fdv9
Enter new 6 char. name of fg 266 ?> = fdv10
Enter new 6 char. name of fg 267 ?> = fdv11
Enter new 6 char. name of fg 268 ?> = fdv12
Enter new 6 char. name of fg 269 ?> = fdv13
Enter new 6 char. name of fg 270 ?> = fdv14
Enter new 6 char. name of fg 271 ?> = fdv15
Enter new 6 char. name of fg 272 ?> = fdv16
Enter new 6 char. name of fg 273 ?> = fdv17
Enter new 6 char. name of fg 274 ?> = fdv18
Enter new 6 char. name of fg 275 ?> = fdv19
Enter new 6 char. name of fg 276 ?> = fdv20
Enter new 6 char. name of fg 277 ?> = fdv21
Enter new 6 char. name of fg 278 ?> = fdv22
Enter new 6 char. name of fg 279 ?> = fdv23
Enter new 6 char. name of fg 280 ?> = fdv24
Enter new 6 char. name of fg 281 ?> = fdv25
Enter new 6 char. name of fg 282 ?> = fdv26
Enter new 6 char. name of fg 283 ?> = fdv27
Enter new 6 char. name of fg 284 ?> = fdv28
Enter new 6 char. name of fg 285 ?> = fdv29
Enter new 6 char. name of fg 286 ?> = fdv30
Enter new 6 char. name of fg 287 ?> = fdv31
Enter new 6 char. name of fg 288 ?> = fdv32
Enter new 6 char. name of fg 289 ?> = fdv33
Enter new 6 char. name of fg 290 ?> = fdv34
Enter new 6 char. name of fg 291 ?> = fdv35
Enter new 6 char. name of fg 292 ?> = fdv36
Enter new 6 char. name of fg 293 ?> = fdv37
Enter new 6 char. name of fg 294 ?> = fdv38
```

```
Enter new 6 char. name of fg 295 ?> = fdv39
Enter new 6 char. name of fg 296 ?> = fdv40
Enter new 6 char. name of fg 297 ?> = fdv41
Enter new 6 char. name of fg 298 ?> = fdv42
Enter new 6 char. name of fg 299 ?> = fdv43
Enter new 6 char. name of fg 300 ?> = fdv44
Enter new 6 char. name of fg 301 ?> = fdv45
Enter new 6 char. name of fg 302 ?> = fdv46
Enter new 6 char. name of fg 303 ?> = fdv47
Enter new 6 char. name of fg 304 ?> = fdv48
Enter new 6 char. name of fg 305 ?> = fdv49
Enter new 6 char. name of fg 306 ?> = fdv50
Enter new 6 char. name of fg 307 ?> = fdv51
Enter new 6 char. name of fg 308 ?> = fdv52
Enter new 6 char. name of fg 309 ?> = fdv53
Enter new 6 char. name of fg 310 ?> = fdv54
Enter new 6 char. name of fg 311 ?> = fdv55
Enter new 6 char. name of fg 312 ?> = fdv56
Enter new 6 char. name of fg 313 ?> = fdv57
Enter new 6 char. name of fg 314 ?> = fdv58
Enter new 6 char. name of fg 315 ?> = fdv59
Enter new 6 char. name of fg 316 ?> = fdv60
Enter new 6 char. name of fg 317 ?> = fdv61
Enter new 6 char. name of fg 318 ?> = fdv62
Enter new 6 char. name of fg 319 ?> = fdv63
Enter new 6 char. name of fg 320 ?> = fdv64
Enter new 6 char. name of fg 321 ?> = fdv65
Enter new 6 char. name of fg 322 ?> = fdv66
Enter new 6 char. name of fg 323 ?> = fdv67
Enter new 6 char. name of fg 324 ?> = fdv68
Enter new 6 char. name of fg 325 ?> = fdv69
Enter new 6 char. name of fg 326 ?> = fdv70
Enter new 6 char. name of fg 327 ?> = fdv71
Enter new 6 char. name of fg 328 ?> = fdv72
Enter new 6 char. name of fg 329 ?> = fdv73
Enter new 6 char. name of fg 330 ?> = fdv74
Enter new 6 char. name of fg 331 ?> = fdv75
Enter new 6 char. name of fg 332 ?> = fdv76
Enter new 6 char. name of fg 333 ?> = fdv77
Enter new 6 char. name of fg 334 ?> = fdv78
Enter new 6 char. name of fg 335 ?> = fdv79
Enter new 6 char. name of fg 336 ?> = fdv80
Enter new 6 char. name of fg 337 ?> = fdv81
Enter new 6 char. name of fg 338 ?> = fdv82
Enter new 6 char. name of fg 339 ?> = fdv83
Enter new 6 char. name of fg 340 ?> = fdv84
Enter new 6 char. name of fg 341 ?> = fdv85
Enter new 6 char. name of fg 342 ?> = fdv86
Enter new 6 char. name of fg 343 ?> = fdv87
Enter new 6 char. name of fg 344 ?> = fdv88
```

```
Enter new 6 char. name of fg 345 ?> = fdv89
Enter new 6 char. name of fg 346 ?> = fdv90
Enter new 6 char. name of fg 347 ?> = fdv91
Enter new 6 char. name of fg 348 ?> = fdv92
Enter new 6 char. name of fg 349 ?> = fdv93
Enter new 6 char. name of fg 350 ?> = fdv94
Enter new 6 char. name of fg 351 ?> = fdv95
Enter new 6 char. name of fg 352 ?> = fdv96
Enter new 6 char. name of fg 353 ?> = fdv97
Enter new 6 char. name of fg 354 ?> = fdv98
Enter new 6 char. name of fg 355 ?> = fdv99
Enter new 6 char. name of fg 356 ?> = fdv100
Enter new 6 char. name of fg 357 ?> = fdv101
Enter new 6 char. name of fg 358 ?> = fdv102
Enter new 6 char. name of fg 359 ?> = fdv103
Enter new 6 char. name of fg 360 ?> = fdv104
Enter new 6 char. name of fg 361 ?> = fdv105
Enter new 6 char. name of fg 362 ?> = fdv106
Enter new 6 char. name of fg 363 ?> = fdv107
Enter new 6 char. name of fg 364 ?> = fdv108
Enter new 6 char. name of fg 365 ?> = fdv109
Enter new 6 char. name of fg 366 ?> = fdv110
Enter new 6 char. name of fg 367 ?> = fdv111
Enter new 6 char. name of fg 368 ?> = fdv112
Enter new 6 char. name of fg 369 ?> = fdv113
Enter new 6 char. name of fg 370 ?> = fdv114
Enter new 6 char. name of fg 371 ?> = fdv115
Enter new 6 char. name of fg 372 ?> = fdv116
Enter new 6 char. name of fg 373 ?> = fdv117
Enter new 6 char. name of fg 374 ?> = fdv118
Enter new 6 char. name of fg 375 ?> = fdv119
Enter new 6 char. name of fg 376 ?> = fdv120
Enter new 6 char. name of fg 377 ?> = fdv121
Enter new 6 char. name of fg 378 ?> = fdv122
Enter new 6 char. name of fg 379 ?> = fdv123
Enter new 6 char. name of fg 380 ?> = fdv124
Enter new 6 char. name of fg 381 ?> = fdv125
Enter new 6 char. name of fg 382 ?> = fdv126
Enter new 6 char. name of fg 383 ?> = fdv127
Enter new 6 char. name of fg 384 ?> = fx0
Enter new 6 char. name of fg 385 ?> = fx1
Enter new 6 char. name of fg 386 ?> = fx2
Enter new 6 char. name of fg 387 ?> = fx3
Enter new 6 char. name of fg 388 ?> = fx4
Enter new 6 char. name of fg 389 ?> = fx5
Enter new 6 char. name of fg 390 ?> = fx6
Enter new 6 char. name of fg 391 ?> = fx7
Enter new 6 char. name of fg 392 ?> = fx8
Enter new 6 char. name of fg 393 ?> = fx9
Enter new 6 char. name of fg 394 ?> = fx10
```

```
Enter new 6 char. name of fg 395 ?> = fx11
Enter new 6 char. name of fg 396 ?> = fx12
Enter new 6 char. name of fg 397 ?> = fx13
Enter new 6 char. name of fg 398 ?> = fx14
Enter new 6 char. name of fg 399 ?> = fx15
Enter new 6 char. name of fg 400 ?> = fx16
Enter new 6 char. name of fg 401 ?> = fx17
Enter new 6 char. name of fg 402 ?> = fx18
Enter new 6 char. name of fg 403 ?> = fx19
Enter new 6 char. name of fg 404 ?> = fx20
Enter new 6 char. name of fg 405 ?> = fx21
Enter new 6 char. name of fg 406 ?> = fx22
Enter new 6 char. name of fg 407 ?> = fx23
Enter new 6 char. name of fg 408 ?> = fx24
Enter new 6 char. name of fg 409 ?> = fx25
Enter new 6 char. name of fg 410 ?> = fx26
Enter new 6 char. name of fg 411 ?> = fx27
Enter new 6 char. name of fg 412 ?> = fx28
Enter new 6 char. name of fg 413 ?> = fx29
Enter new 6 char. name of fg 414 ?> = fx30
Enter new 6 char. name of fg 415 ?> = fx31
Enter new 6 char. name of fg 416 ?> = fx32
Enter new 6 char. name of fg 417 ?> = fx33
Enter new 6 char. name of fg 418 ?> = fx34
Enter new 6 char. name of fg 419 ?> = fx35
Enter new 6 char. name of fg 420 ?> = fx36
Enter new 6 char. name of fg 421 ?> = fdisp
Enter new 6 char. name of fg 422 ?> = fhold
Enter new 6 char. name of fg 423 ?> = fning
Enter new 6 char. name of fg 424 ?> = fabort
Enter new 6 char. name of fg 425 ?> = fstart
Enter new 6 char. name of fg 426 ?> = fstop
Enter new 6 char. name of fg 427 ?> = ftrip0
Enter new 6 char. name of fg 428 ?> = ftrip1
Enter new 6 char. name of fg 429 ?> = ftrip2
Enter new 6 char. name of fg 430 ?> = ftrip3
Enter new 6 char. name of fg 431 ?> = ftrip4
MUPF?>  = add 89 in
Enter new 6 char. name of in 0 ?> = bitval
Enter new 6 char. name of in 1 ?> = iloop
Enter new 6 char. name of in 2 ?> = jloop
Enter new 6 char. name of in 3 ?> = kloop
Enter new 6 char. name of in 4 ?> = lloop
Enter new 6 char. name of in 5 ?> = mloop
Enter new 6 char. name of in 6 ?> = nloop
Enter new 6 char. name of in 7 ?> = oloop
Enter new 6 char. name of in 8 ?> = ploop
Enter new 6 char. name of in 9 ?> = index
Enter new 6 char. name of in 10 ?> = trip0
Enter new 6 char. name of in 11 ?> = trip1
```

```
Enter new 6 char. name of in 12 ?> = trip2
Enter new 6 char. name of in 13 ?> = trip3
Enter new 6 char. name of in 14 ?> = trip4
Enter new 6 char. name of in 15 ?> = trip5
Enter new 6 char. name of in 16 ?> = trip6
Enter new 6 char. name of in 17 ?> = trip7
Enter new 6 char. name of in 18 ?> = nadi
Enter new 6 char. name of in 19 ?> = blkval
Enter new 6 char. name of in 20 ?> = sort
Enter new 6 char. name of in 21 ?> = bval
Enter new 6 char. name of in 22 ?> = hldstp
Enter new 6 char. name of in 23 ?> = errn
Enter new 6 char. name of in 24 ?> = errl
Enter new 6 char. name of in 25 ?> = int0
Enter new 6 char. name of in 26 ?> = int1
Enter new 6 char. name of in 27 ?> = int2
Enter new 6 char. name of in 28 ?> = int3
Enter new 6 char. name of in 29 ?> = int4
Enter new 6 char. name of in 30 ?> = int5
Enter new 6 char. name of in 31 ?> = int6
Enter new 6 char. name of in 32 ?> = int7
Enter new 6 char. name of in 33 ?> = int8
Enter new 6 char. name of in 34 ?> = int9
Enter new 6 char. name of in 35 ?> = int10
Enter new 6 char. name of in 36 ?> = int11
Enter new 6 char. name of in 37 ?> = int12
Enter new 6 char. name of in 38 ?> = int13
Enter new 6 char. name of in 39 ?> = int14
Enter new 6 char. name of in 40 ?> = int15
Enter new 6 char. name of in 41 ?> = int16
Enter new 6 char. name of in 42 ?> = int17
Enter new 6 char. name of in 43 ?> = int18
Enter new 6 char. name of in 44 ?> = int19
Enter new 6 char. name of in 45 ?> = int20
Enter new 6 char. name of in 46 ?> = int21
Enter new 6 char. name of in 47 ?> = int22
Enter new 6 char. name of in 48 ?> = int23
Enter new 6 char. name of in 49 ?> = int24
Enter new 6 char. name of in 50 ?> = int25
Enter new 6 char. name of in 51 ?> = int26
Enter new 6 char. name of in 52 ?> = int27
Enter new 6 char. name of in 53 ?> = int28
Enter new 6 char. name of in 54 ?> = int29
Enter new 6 char. name of in 55 ?> = int30
Enter new 6 char. name of in 56 ?> = int31
Enter new 6 char. name of in 57 ?> = int32
Enter new 6 char. name of in 58 ?> = int33
Enter new 6 char. name of in 59 ?> = int34
Enter new 6 char. name of in 60 ?> = int35
Enter new 6 char. name of in 61 ?> = int36
```

```
Enter new 6 char. name of in 62 ?> = int37
Enter new 6 char. name of in 63 ?> = int38
Enter new 6 char. name of in 64 ?> = int39
Enter new 6 char. name of in 65 ?> = int40
Enter new 6 char. name of in 66 ?> = int41
Enter new 6 char. name of in 67 ?> = int42
Enter new 6 char. name of in 68 ?> = int43
Enter new 6 char. name of in 69 ?> = int44
Enter new 6 char. name of in 70 ?> = int45
Enter new 6 char. name of in 71 ?> = int46
Enter new 6 char. name of in 72 ?> = int47
Enter new 6 char. name of in 73 ?> = int48
Enter new 6 char. name of in 74 ?> = int49
Enter new 6 char. name of in 75 ?> = int50
Enter new 6 char. name of in 76 ?> = int51
Enter new 6 char. name of in 77 ?> = int52
Enter new 6 char. name of in 78 ?> = int53
Enter new 6 char. name of in 79 ?> = int54
Enter new 6 char. name of in 80 ?> = int55
Enter new 6 char. name of in 81 ?> = int56
Enter new 6 char. name of in 82 ?> = int57
Enter new 6 char. name of in 83 ?> = int58
Enter new 6 char. name of in 84 ?> = int59
Enter new 6 char. name of in 85 ?> = int60
Enter new 6 char. name of in 86 ?> = int61
Enter new 6 char. name of in 87 ?> = int62
Enter new 6 char. name of in 88 ?> = int63
MUPF?>  = add 16 pt
Enter new 6 char. name of pt 0 ?> = pepn
Enter new 6 char. name of pt 1 ?> = pdev
Enter new 6 char. name of pt 2 ?> = pai
Enter new 6 char. name of pt 3 ?> = pdin
Enter new 6 char. name of pt 4 ?> = ptrip
Enter new 6 char. name of pt 5 ?> = pdeadb
Enter new 6 char. name of pt 6 ?> = palarm
Enter new 6 char. name of pt 7 ?> = pack
Enter new 6 char. name of pt 8 ?> = pmessg
Enter new 6 char. name of pt 9 ?> = pdsabl
Enter new 6 char. name of pt 10 ?> = pback
Enter new 6 char. name of pt 11 ?> = preps
Enter new 6 char. name of pt 12 ?> = plast
Enter new 6 char. name of pt 13 ?> = pskid
Enter new 6 char. name of pt 14 ?> = pstat
Enter new 6 char. name of pt 15 ?> = pstrng
MUPF?>  = add 202 sg
Enter new 6 char. name of sg 0 ?> = epns0
Enter max string length for sg 0  ?> = 10
Enter new 6 char. name of sg 1 ?> = epns1
Enter max string length for sg 1  ?> = 10
Enter new 6 char. name of sg 2 ?> = epns2
```

```
Enter max string length for sg 2  ?> = 10
Enter new 6 char. name of sg 3 ?> = epns3
Enter max string length for sg 3  ?> = 10
Enter new 6 char. name of sg 4 ?> = epns4
Enter max string length for sg 4  ?> = 10
Enter new 6 char. name of sg 5 ?> = epns5
Enter max string length for sg 5  ?> = 10
Enter new 6 char. name of sg 6 ?> = epns6
Enter max string length for sg 6  ?> = 10
Enter new 6 char. name of sg 7 ?> = epns7
Enter max string length for sg 7  ?> = 10
Enter new 6 char. name of sg 8 ?> = epns8
Enter max string length for sg 8  ?> = 10
Enter new 6 char. name of sg 9 ?> = epns9
Enter max string length for sg 9  ?> = 10
Enter new 6 char. name of sg 10 ?> = epns10
Enter max string length for sg 10  ?> = 10
Enter new 6 char. name of sg 11 ?> = epns11
Enter max string length for sg 11  ?> = 10
Enter new 6 char. name of sg 12 ?> = epns12
Enter max string length for sg 12  ?> = 10
Enter new 6 char. name of sg 13 ?> = epns13
Enter max string length for sg 13  ?> = 10
Enter new 6 char. name of sg 14 ?> = epns14
Enter max string length for sg 14  ?> = 10
Enter new 6 char. name of sg 15 ?> = epns15
Enter max string length for sg 15  ?> = 10
Enter new 6 char. name of sg 16 ?> = epns16
Enter max string length for sg 16  ?> = 10
Enter new 6 char. name of sg 17 ?> = epns17
Enter max string length for sg 17  ?> = 10
Enter new 6 char. name of sg 18 ?> = epns18
Enter max string length for sg 18  ?> = 10
Enter new 6 char. name of sg 19 ?> = epns19
Enter max string length for sg 19  ?> = 10
Enter new 6 char. name of sg 20 ?> = epns20
Enter max string length for sg 20  ?> = 10
Enter new 6 char. name of sg 21 ?> = epns21
Enter max string length for sg 21  ?> = 10
Enter new 6 char. name of sg 22 ?> = epns22
Enter max string length for sg 22  ?> = 10
Enter new 6 char. name of sg 23 ?> = epns23
Enter max string length for sg 23  ?> = 10
Enter new 6 char. name of sg 24 ?> = epns24
Enter max string length for sg 24  ?> = 10
Enter new 6 char. name of sg 25 ?> = epns25
Enter max string length for sg 25  ?> = 10
Enter new 6 char. name of sg 26 ?> = epns26
Enter max string length for sg 26  ?> = 10
Enter new 6 char. name of sg 27 ?> = epns27
```

```
Enter max string length for sg 27  ?> = 10
Enter new 6 char. name of sg 28 ?> = epns28
Enter max string length for sg 28  ?> = 10
Enter new 6 char. name of sg 29 ?> = epns29
Enter max string length for sg 29  ?> = 10
Enter new 6 char. name of sg 30 ?> = epns30
Enter max string length for sg 30  ?> = 10
Enter new 6 char. name of sg 31 ?> = epns31
Enter max string length for sg 31  ?> = 10
Enter new 6 char. name of sg 32 ?> = epns32
Enter max string length for sg 32  ?> = 10
Enter new 6 char. name of sg 33 ?> = epns33
Enter max string length for sg 33  ?> = 10
Enter new 6 char. name of sg 34 ?> = epns34
Enter max string length for sg 34  ?> = 10
Enter new 6 char. name of sg 35 ?> = epns35
Enter max string length for sg 35  ?> = 10
Enter new 6 char. name of sg 36 ?> = epns36
Enter max string length for sg 36  ?> = 10
Enter new 6 char. name of sg 37 ?> = epns37
Enter max string length for sg 37  ?> = 10
Enter new 6 char. name of sg 38 ?> = epns38
Enter max string length for sg 38  ?> = 10
Enter new 6 char. name of sg 39 ?> = epns39
Enter max string length for sg 39  ?> = 10
Enter new 6 char. name of sg 40 ?> = epns40
Enter max string length for sg 40  ?> = 10
Enter new 6 char. name of sg 41 ?> = epns41
Enter max string length for sg 41  ?> = 10
Enter new 6 char. name of sg 42 ?> = epns42
Enter max string length for sg 42  ?> = 10
Enter new 6 char. name of sg 43 ?> = epns43
Enter max string length for sg 43  ?> = 10
Enter new 6 char. name of sg 44 ?> = epns44
Enter max string length for sg 44  ?> = 10
Enter new 6 char. name of sg 45 ?> = epns45
Enter max string length for sg 45  ?> = 10
Enter new 6 char. name of sg 46 ?> = epns46
Enter max string length for sg 46  ?> = 10
Enter new 6 char. name of sg 47 ?> = epns47
Enter max string length for sg 47  ?> = 10
Enter new 6 char. name of sg 48 ?> = epns48
Enter max string length for sg 48  ?> = 10
Enter new 6 char. name of sg 49 ?> = epns49
Enter max string length for sg 49  ?> = 10
Enter new 6 char. name of sg 50 ?> = epns50
Enter max string length for sg 50  ?> = 10
Enter new 6 char. name of sg 51 ?> = epns51
Enter max string length for sg 51  ?> = 10
Enter new 6 char. name of sg 52 ?> = epns52
```

```
Enter max string length for sg 52  ?> = 10
Enter new 6 char. name of sg 53 ?> = epns53
Enter max string length for sg 53  ?> = 10
Enter new 6 char. name of sg 54 ?> = epns54
Enter max string length for sg 54  ?> = 10
Enter new 6 char. name of sg 55 ?> = epns55
Enter max string length for sg 55  ?> = 10
Enter new 6 char. name of sg 56 ?> = epns56
Enter max string length for sg 56  ?> = 10
Enter new 6 char. name of sg 57 ?> = epns57
Enter max string length for sg 57  ?> = 10
Enter new 6 char. name of sg 58 ?> = epns58
Enter max string length for sg 58  ?> = 10
Enter new 6 char. name of sg 59 ?> = epns59
Enter max string length for sg 59  ?> = 10
Enter new 6 char. name of sg 60 ?> = epns60
Enter max string length for sg 60  ?> = 10
Enter new 6 char. name of sg 61 ?> = epns61
Enter max string length for sg 61  ?> = 10
Enter new 6 char. name of sg 62 ?> = epns62
Enter max string length for sg 62  ?> = 10
Enter new 6 char. name of sg 63 ?> = epns63
Enter max string length for sg 63  ?> = 10
Enter new 6 char. name of sg 64 ?> = epns64
Enter max string length for sg 64  ?> = 10
Enter new 6 char. name of sg 65 ?> = epns65
Enter max string length for sg 65  ?> = 10
Enter new 6 char. name of sg 66 ?> = epns66
Enter max string length for sg 66  ?> = 10
Enter new 6 char. name of sg 67 ?> = epns67
Enter max string length for sg 67  ?> = 10
Enter new 6 char. name of sg 68 ?> = epns68
Enter max string length for sg 68  ?> = 10
Enter new 6 char. name of sg 69 ?> = epns69
Enter max string length for sg 69  ?> = 10
Enter new 6 char. name of sg 70 ?> = epns70
Enter max string length for sg 70  ?> = 10
Enter new 6 char. name of sg 71 ?> = epns71
Enter max string length for sg 71  ?> = 10
Enter new 6 char. name of sg 72 ?> = epns72
Enter max string length for sg 72  ?> = 10
Enter new 6 char. name of sg 73 ?> = epns73
Enter max string length for sg 73  ?> = 10
Enter new 6 char. name of sg 74 ?> = epns74
Enter max string length for sg 74  ?> = 10
Enter new 6 char. name of sg 75 ?> = epns75
Enter max string length for sg 75  ?> = 10
Enter new 6 char. name of sg 76 ?> = epns76
Enter max string length for sg 76  ?> = 10
Enter new 6 char. name of sg 77 ?> = epns77
```

```
Enter max string length for sg 77  ?> = 10
Enter new 6 char. name of sg 78 ?> = epns78
Enter max string length for sg 78  ?> = 10
Enter new 6 char. name of sg 79 ?> = epns79
Enter max string length for sg 79  ?> = 10
Enter new 6 char. name of sg 80 ?> = epns80
Enter max string length for sg 80  ?> = 10
Enter new 6 char. name of sg 81 ?> = epns81
Enter max string length for sg 81  ?> = 10
Enter new 6 char. name of sg 82 ?> = epns82
Enter max string length for sg 82  ?> = 10
Enter new 6 char. name of sg 83 ?> = epns83
Enter max string length for sg 83  ?> = 10
Enter new 6 char. name of sg 84 ?> = epns84
Enter max string length for sg 84  ?> = 10
Enter new 6 char. name of sg 85 ?> = epns85
Enter max string length for sg 85  ?> = 10
Enter new 6 char. name of sg 86 ?> = epns86
Enter max string length for sg 86  ?> = 10
Enter new 6 char. name of sg 87 ?> = epns87
Enter max string length for sg 87  ?> = 10
Enter new 6 char. name of sg 88 ?> = epns88
Enter max string length for sg 88  ?> = 10
Enter new 6 char. name of sg 89 ?> = epns89
Enter max string length for sg 89  ?> = 10
Enter new 6 char. name of sg 90 ?> = epns90
Enter max string length for sg 90  ?> = 10
Enter new 6 char. name of sg 91 ?> = epns91
Enter max string length for sg 91  ?> = 10
Enter new 6 char. name of sg 92 ?> = epns92
Enter max string length for sg 92  ?> = 10
Enter new 6 char. name of sg 93 ?> = epns93
Enter max string length for sg 93  ?> = 10
Enter new 6 char. name of sg 94 ?> = epns94
Enter max string length for sg 94  ?> = 10
Enter new 6 char. name of sg 95 ?> = epns95
Enter max string length for sg 95  ?> = 10
Enter new 6 char. name of sg 96 ?> = epns96
Enter max string length for sg 96  ?> = 10
Enter new 6 char. name of sg 97 ?> = epns97
Enter max string length for sg 97  ?> = 10
Enter new 6 char. name of sg 98 ?> = epns98
Enter max string length for sg 98  ?> = 10
Enter new 6 char. name of sg 99 ?> = epns99
Enter max string length for sg 99  ?> = 10
Enter new 6 char. name of sg 100 ?> = lsg0
Enter max string length for sg 100  ?> = 30
Enter new 6 char. name of sg 101 ?> = lsg1
Enter max string length for sg 101  ?> = 30
Enter new 6 char. name of sg 102 ?> = lsg2
```

```
Enter max string length for sg 102  ?> = 30
Enter new 6 char. name of sg 103 ?> = lsg3
Enter max string length for sg 103  ?> = 30
Enter new 6 char. name of sg 104 ?> = lsg4
Enter max string length for sg 104  ?> = 30
Enter new 6 char. name of sg 105 ?> = lsg5
Enter max string length for sg 105  ?> = 30
Enter new 6 char. name of sg 106 ?> = lsg6
Enter max string length for sg 106  ?> = 30
Enter new 6 char. name of sg 107 ?> = lsg7
Enter max string length for sg 107  ?> = 30
Enter new 6 char. name of sg 108 ?> = lsg8
Enter max string length for sg 108  ?> = 30
Enter new 6 char. name of sg 109 ?> = lsg9
Enter max string length for sg 109  ?> = 30
Enter new 6 char. name of sg 110 ?> = lsg10
Enter max string length for sg 110  ?> = 30
Enter new 6 char. name of sg 111 ?> = lsg11
Enter max string length for sg 111  ?> = 30
Enter new 6 char. name of sg 112 ?> = lsg12
Enter max string length for sg 112  ?> = 30
Enter new 6 char. name of sg 113 ?> = lsg13
Enter max string length for sg 113  ?> = 30
Enter new 6 char. name of sg 114 ?> = lsg14
Enter max string length for sg 114  ?> = 30
Enter new 6 char. name of sg 115 ?> = lsg15
Enter max string length for sg 115  ?> = 30
Enter new 6 char. name of sg 116.?> = lsg16
Enter max string length for sg 116  ?> = 30
Enter new 6 char. name of sg 117 ?> = lsg17
Enter max string length for sg 117  ?> = 30
Enter new 6 char. name of sg 118 ?> = lsg18
Enter max string length for sg 118  ?> = 30
Enter new 6 char. name of sg 119 ?> = lsg19
Enter max string length for sg 119  ?> = 30
Enter new 6 char. name of sg 120 ?> = lsg20
Enter max string length for sg 120  ?> = 30
Enter new 6 char. name of sg 121 ?> = lsg21
Enter max string length for sg 121  ?> = 30
Enter new 6 char. name of sg 122 ?> = lsg22
Enter max string length for sg 122  ?> = 30
Enter new 6 char. name of sg 123 ?> = lsg23
Enter max string length for sg 123  ?> = 30
Enter new 6 char. name of sg 124 ?> = lsg24
Enter max string length for sg 124  ?> = 30
Enter new 6 char. name of sg 125 ?> = lsg25
Enter max string length for sg 125  ?> = 30
Enter new 6 char. name of sg 126 ?> = lsg26
Enter max string length for sg 126  ?> = 30
Enter new 6 char. name of sg 127 ?> = lsg27
```

```
Enter max string length for sg 127  ?> = 30
Enter new 6 char. name of sg 128 ?> = lsg28
Enter max string length for sg 128  ?> = 30
Enter new 6 char. name of sg 129 ?> = lsg29
Enter max string length for sg 129  ?> = 30
Enter new 6 char. name of sg 130 ?> = lsg30
Enter max string length for sg 130  ?> = 30
Enter new 6 char. name of sg 131 ?> = lsg31
Enter max string length for sg 131  ?> = 30
Enter new 6 char. name of sg 132 ?> = lsg32
Enter max string length for sg 132  ?> = 30
Enter new 6 char. name of sg 133 ?> = lsg33
Enter max string length for sg 133  ?> = 30
Enter new 6 char. name of sg 134 ?> = lsg34
Enter max string length for sg 134  ?> = 30
Enter new 6 char. name of sg 135 ?> = lsg35
Enter max string length for sg 135  ?> = 30
Enter new 6 char. name of sg 136 ?> = lsg36
Enter max string length for sg 136  ?> = 30
Enter new 6 char. name of sg 137 ?> = lsg37
Enter max string length for sg 137  ?> = 30
Enter new 6 char. name of sg 138 ?> = lsg38
Enter max string length for sg 138  ?> = 30
Enter new 6 char. name of sg 139 ?> = lsg39
Enter max string length for sg 139  ?> = 30
Enter new 6 char. name of sg 140 ?> = lsg40
Enter max string length for sg 140  ?> = 30
Enter new 6 char. name of sg 141 ?> = lsg41
Enter max string length for sg 141  ?> = 30
Enter new 6 char. name of sg 142 ?> = lsg42
Enter max string length for sg 142  ?> = 30
Enter new 6 char. name of sg 143 ?> = lsg43
Enter max string length for sg 143  ?> = 30
Enter new 6 char. name of sg 144 ?> = lsg44
Enter max string length for sg 144  ?> = 30
Enter new 6 char. name of sg 145 ?> = lsg45
Enter max string length for sg 145  ?> = 30
Enter new 6 char. name of sg 146 ?> = lsg46
Enter max string length for sg 146  ?> = 30
Enter new 6 char. name of sg 147 ?> = lsg47
Enter max string length for sg 147  ?> = 30
Enter new 6 char. name of sg 148 ?> = lsg48
Enter max string length for sg 148  ?> = 30
Enter new 6 char. name of sg 149 ?> = lsg49
Enter max string length for sg 149  ?> = 30
Enter new 6 char. name of sg 150 ?> = lsg50
Enter max string length for sg 150  ?> = 30
Enter new 6 char. name of sg 151 ?> = lsg51
Enter max string length for sg 151  ?> = 30
Enter new 6 char. name of sg 152 ?> = lsg52
```

```
Enter max string length for sg 152  ?> = 30
Enter new 6 char. name of sg 153 ?> = lsg53
Enter max string length for sg 153  ?> = 30
Enter new 6 char. name of sg 154 ?> = lsg54
Enter max string length for sg 154  ?> = 30
Enter new 6 char. name of sg 155 ?> = lsg55
Enter max string length for sg 155  ?> = 30
Enter new 6 char. name of sg 156 ?> = lsg56
Enter max string length for sg 156  ?> = 30
Enter new 6 char. name of sg 157 ?> = lsg57
Enter max string length for sg 157  ?> = 30
Enter new 6 char. name of sg 158 ?> = lsg58
Enter max string length for sg 158  ?> = 30
Enter new 6 char. name of sg 159 ?> = lsg59
Enter max string length for sg 159  ?> = 30
Enter new 6 char. name of sg 160 ?> = lsg60
Enter max string length for sg 160  ?> = 30
Enter new 6 char. name of sg 161 ?> = lsg61
Enter max string length for sg 161  ?> = 30
Enter new 6 char. name of sg 162 ?> = lsg62
Enter max string length for sg 162  ?> = 30
Enter new 6 char. name of sg 163 ?> = lsg63
Enter max string length for sg 163  ?> = 30
Enter new 6 char. name of sg 164 ?> = lsg64
Enter max string length for sg 164  ?> = 30
Enter new 6 char. name of sg 165 ?> = lsg65
Enter max string length for sg 165  ?> = 30
Enter new 6 char. name of sg 166 ?> = lsg66
Enter max string length for sg 166  ?> = 30
Enter new 6 char. name of sg 167 ?> = lsg67
Enter max string length for sg 167  ?> = 30
Enter new 6 char. name of sg 168 ?> = lsg68
Enter max string length for sg 168  ?> = 30
Enter new 6 char. name of sg 169 ?> = lsg69
Enter max string length for sg 169  ?> = 30
Enter new 6 char. name of sg 170 ?> = lsg70
Enter max string length for sg 170  ?> = 30
Enter new 6 char. name of sg 171 ?> = lsg71
Enter max string length for sg 171  ?> = 30
Enter new 6 char. name of sg 172 ?> = lsg72
Enter max string length for sg 172  ?> = 30
Enter new 6 char. name of sg 173 ?> = lsg73
Enter max string length for sg 173  ?> = 30
Enter new 6 char. name of sg 174 ?> = lsg74
Enter max string length for sg 174  ?> = 30
Enter new 6 char. name of sg 175 ?> = lsg75
Enter max string length for sg 175  ?> = 30
Enter new 6 char. name of sg 176 ?> = lsg76
Enter max string length for sg 176  ?> = 30
Enter new 6 char. name of sg 177 ?> = lsg77
```

```
Enter max string length for sg 177  ?> = 30
Enter new 6 char. name of sg 178 ?> = lsg78
Enter max string length for sg 178  ?> = 30
Enter new 6 char. name of sg 179 ?> = lsg79
Enter max string length for sg 179  ?> = 30
Enter new 6 char. name of sg 180 ?> = lsg80
Enter max string length for sg 180  ?> = 30
Enter new 6 char. name of sg 181 ?> = lsg81
Enter max string length for sg 181  ?> = 30
Enter new 6 char. name of sg 182 ?> = lsg82
Enter max string length for sg 182  ?> = 30
Enter new 6 char. name of sg 183 ?> = lsg83
Enter max string length for sg 183  ?> = 30
Enter new 6 char. name of sg 184 ?> = lsg84
Enter max string length for sg 184  ?> = 30
Enter new 6 char. name of sg 185 ?> = lsg85
Enter max string length for sg 185  ?> = 30
Enter new 6 char. name of sg 186 ?> = lsg86
Enter max string length for sg 186  ?> = 30
Enter new 6 char. name of sg 187 ?> = lsg87
Enter max string length for sg 187  ?> = 30
Enter new 6 char. name of sg 188 ?> = lsg88
Enter max string length for sg 188  ?> = 30
Enter new 6 char. name of sg 189 ?> = lsg89
Enter max string length for sg 189  ?> = 30
Enter new 6 char. name of sg 190 ?> = lsg90
Enter max string length for sg 190  ?> = 30
Enter new 6 char. name of sg 191 ?> = lsg91
Enter max string length for sg 191  ?> = 30
Enter new 6 char. name of sg 192 ?> = lsg92
Enter max string length for sg 192  ?> = 30
Enter new 6 char. name of sg 193 ?> = lsg93
Enter max string length for sg 193  ?> = 30
Enter new 6 char. name of sg 194 ?> = lsg94
Enter max string length for sg 194  ?> = 30
Enter new 6 char. name of sg 195 ?> = lsg95
Enter max string length for sg 195  ?> = 30
Enter new 6 char. name of sg 196 ?> = lsg96
Enter max string length for sg 196  ?> = 30
Enter new 6 char. name of sg 197 ?> = lsg97
Enter max string length for sg 197  ?> = 30
Enter new 6 char. name of sg 198 ?> = lsg98
Enter max string length for sg 198  ?> = 30
Enter new 6 char. name of sg 199 ?> = lsg99
Enter max string length for sg 199  ?> = 30
Enter new 6 char. name of sg 200 ?> = time0
Enter max string length for sg 200  ?> = 10
Enter new 6 char. name of sg 201 ?> = date0
Enter max string length for sg 201  ?> = 10
MUPF?>  = add 5 st
```

```
Enter new 6 char. name of st 0 ?> = rcptmr
Enter new 6 char. name of st 1 ?> = vltmr0
Enter new 6 char. name of st 2 ?> = vltmr1
Enter new 6 char. name of st 3 ?> = vltmr2
Enter new 6 char. name of st 4 ?> = vltmr3
MUPF?>  = add 5 mt
Enter new 6 char. name of mt 0 ?> = dtmr0
Enter new 6 char. name of mt 1 ?> = dtmr1
Enter new 6 char. name of mt 2 ?> = dtmr2
Enter new 6 char. name of mt 3 ?> = dtmr3
Enter new 6 char. name of mt 4 ?> = dtmr4
MUPF?>  =
UPF?>
 end of file encountered
```

SHORTO

```
UPF Version 6.3-2       05-Dec-94 09:35:38
UPF?>  = de short0
UPF?>  = cr short0,*
Enter 4 character name of RPF to associate?> = farm
MUPF?>  =  add 208 fg
Enter new 6 char. name of fg 0 ?> = falm0
Enter new 6 char. name of fg 1 ?> = falm1
Enter new 6 char. name of fg 2 ?> = falm2
Enter new 6 char. name of fg 3 ?> = falm3
Enter new 6 char. name of fg 4 ?> = falm4
Enter new 6 char. name of fg 5 ?> = falm5
Enter new 6 char. name of fg 6 ?> = falm6
Enter new 6 char. name of fg 7 ?> = falm7
Enter new 6 char. name of fg 8 ?> = falm8
Enter new 6 char. name of fg 9 ?> = falm9
Enter new 6 char. name of fg 10 ?> = falm10
Enter new 6 char. name of fg 11 ?> = falm11
Enter new 6 char. name of fg 12 ?> = falm12
Enter new 6 char. name of fg 13 ?> = falm13
Enter new 6 char. name of fg 14 ?> = falm14
Enter new 6 char. name of fg 15 ?> = falm15
Enter new 6 char. name of fg 16 ?> = falm16
Enter new 6 char. name of fg 17 ?> = falm17
Enter new 6 char. name of fg 18 ?> = falm18
Enter new 6 char. name of fg 19 ?> = falm19
Enter new 6 char. name of fg 20 ?> = falm20
Enter new 6 char. name of fg 21 ?> = falm21
Enter new 6 char. name of fg 22 ?> = falm22
Enter new 6 char. name of fg 23 ?> = falm23
Enter new 6 char. name of fg 24 ?> = falm24
Enter new 6 char. name of fg 25 ?> = falm25
Enter new 6 char. name of fg 26 ?> = falm26
Enter new 6 char. name of fg 27 ?> = falm27
Enter new 6 char. name of fg 28 ?> = falm28
Enter new 6 char. name of fg 29 ?> = falm29
Enter new 6 char. name of fg 30 ?> = falm30
Enter new 6 char. name of fg 31 ?> = falm31
Enter new 6 char. name of fg 32 ?> = falm32
Enter new 6 char. name of fg 33 ?> = falm33
Enter new 6 char. name of fg 34 ?> = falm34
Enter new 6 char. name of fg 35 ?> = falm35
Enter new 6 char. name of fg 36 ?> = falm36
Enter new 6 char. name of fg 37 ?> = falm37
Enter new 6 char. name of fg 38 ?> = falm38
Enter new 6 char. name of fg 39 ?> = falm39
Enter new 6 char. name of fg 40 ?> = fpas0
Enter new 6 char. name of fg 41 ?> = fpas1
Enter new 6 char. name of fg 42 ?> = fpas2
Enter new 6 char. name of fg 43 ?> = fpas3
Enter new 6 char. name of fg 44 ?> = fpas4
```

```
Enter new 6 char. name of fg 45 ?> = fpas5
Enter new 6 char. name of fg 46 ?> = fpas6
Enter new 6 char. name of fg 47 ?> = fpas7
Enter new 6 char. name of fg 48 ?> = fpas8
Enter new 6 char. name of fg 49 ?> = fpas9
Enter new 6 char. name of fg 50 ?> = fpas10
Enter new 6 char. name of fg 51 ?> = fpas11
Enter new 6 char. name of fg 52 ?> = fpas12
Enter new 6 char. name of fg 53 ?> = fpas13
Enter new 6 char. name of fg 54 ?> = fpas14
Enter new 6 char. name of fg 55 ?> = fpas15
Enter new 6 char. name of fg 56 ?> = fpas16
Enter new 6 char. name of fg 57 ?> = fpas17
Enter new 6 char. name of fg 58 ?> = fpas18
Enter new 6 char. name of fg 59 ?> = fpas19
Enter new 6 char. name of fg 60 ?> = fpas20
Enter new 6 char. name of fg 61 ?> = fpas21
Enter new 6 char. name of fg 62 ?> = fpas22
Enter new 6 char. name of fg 63 ?> = fpas23
Enter new 6 char. name of fg 64 ?> = fpas24
Enter new 6 char. name of fg 65 ?> = fpas25
Enter new 6 char. name of fg 66 ?> = fpas26
Enter new 6 char. name of fg 67 ?> = fpas27
Enter new 6 char. name of fg 68 ?> = fpas28
Enter new 6 char. name of fg 69 ?> = fpas29
Enter new 6 char. name of fg 70 ?> = fpas30
Enter new 6 char. name of fg 71 ?> = fpas31
Enter new 6 char. name of fg 72 ?> = fpas32
Enter new 6 char. name of fg 73 ?> = fpas33
Enter new 6 char. name of fg 74 ?> = fpas34
Enter new 6 char. name of fg 75 ?> = fpas35
Enter new 6 char. name of fg 76 ?> = fpas36
Enter new 6 char. name of fg 77 ?> = fpas37
Enter new 6 char. name of fg 78 ?> = fpas38
Enter new 6 char. name of fg 79 ?> = fpas39
Enter new 6 char. name of fg 80 ?> = fdis0
Enter new 6 char. name of fg 81 ?> = fdis1
Enter new 6 char. name of fg 82 ?> = fdis2
Enter new 6 char. name of fg 83 ?> = fdis3
Enter new 6 char. name of fg 84 ?> = fdis4
Enter new 6 char. name of fg 85 ?> = fdis5
Enter new 6 char. name of fg 86 ?> = fdis6
Enter new 6 char. name of fg 87 ?> = fdis7
Enter new 6 char. name of fg 88 ?> = fdis8
Enter new 6 char. name of fg 89 ?> = fdis9
Enter new 6 char. name of fg 90 ?> = fdis10
Enter new 6 char. name of fg 91 ?> = fdis11
Enter new 6 char. name of fg 92 ?> = fdis12
Enter new 6 char. name of fg 93 ?> = fdis13
Enter new 6 char. name of fg 94 ?> = fdis14
```

```
Enter new 6 char. name of fg 95 ?> = fdis15
Enter new 6 char. name of fg 96 ?> = fdis16
Enter new 6 char. name of fg 97 ?> = fdis17
Enter new 6 char. name of fg 98 ?> = fdis18
Enter new 6 char. name of fg 99 ?> = fdis19
Enter new 6 char. name of fg 100 ?> = fdis20
Enter new 6 char. name of fg 101 ?> = fdis21
Enter new 6 char. name of fg 102 ?> = fdis22
Enter new 6 char. name of fg 103 ?> = fdis23
Enter new 6 char. name of fg 104 ?> = fdis24
Enter new 6 char. name of fg 105 ?> = fdis25
Enter new 6 char. name of fg 106 ?> = fdis26
Enter new 6 char. name of fg 107 ?> = fdis27
Enter new 6 char. name of fg 108 ?> = fdis28
Enter new 6 char. name of fg 109 ?> = fdis29
Enter new 6 char. name of fg 110 ?> = fdis30
Enter new 6 char. name of fg 111 ?> = fdis31
Enter new 6 char. name of fg 112 ?> = fdis32
Enter new 6 char. name of fg 113 ?> = fdis33
Enter new 6 char. name of fg 114 ?> = fdis34
Enter new 6 char. name of fg 115 ?> = fdis35
Enter new 6 char. name of fg 116 ?> = fdis36
Enter new 6 char. name of fg 117 ?> = fdis37
Enter new 6 char. name of fg 118 ?> = fdis38
Enter new 6 char. name of fg 119 ?> = fdis39
Enter new 6 char. name of fg 120 ?> = frv0
Enter new 6 char. name of fg 121 ?> = frv1
Enter new 6 char. name of fg 122 ?> = frv2
Enter new 6 char. name of fg 123 ?> = frv3
Enter new 6 char. name of fg 124 ?> = frv4
Enter new 6 char. name of fg 125 ?> = frv5
Enter new 6 char. name of fg 126 ?> = frv6
Enter new 6 char. name of fg 127 ?> = frv7
Enter new 6 char. name of fg 128 ?> = frv8
Enter new 6 char. name of fg 129 ?> = frv9
Enter new 6 char. name of fg 130 ?> = frv10
Enter new 6 char. name of fg 131 ?> = frv11
Enter new 6 char. name of fg 132 ?> = frv12
Enter new 6 char. name of fg 133 ?> = frv13
Enter new 6 char. name of fg 134 ?> = frv14
Enter new 6 char. name of fg 135 ?> = frv15
Enter new 6 char. name of fg 136 ?> = frv16
Enter new 6 char. name of fg 137 ?> = frv17
Enter new 6 char. name of fg 138 ?> = frv18
Enter new 6 char. name of fg 139 ?> = frv19
Enter new 6 char. name of fg 140 ?> = frv20
Enter new 6 char. name of fg 141 ?> = frv21
Enter new 6 char. name of fg 142 ?> = frv22
Enter new 6 char. name of fg 143 ?> = frv23
Enter new 6 char. name of fg 144 ?> = frv24
```

```
Enter new 6 char. name of fg 145 ?> = frv25
Enter new 6 char. name of fg 146 ?> = frv26
Enter new 6 char. name of fg 147 ?> = frv27
Enter new 6 char. name of fg 148 ?> = frv28
Enter new 6 char. name of fg 149 ?> = frv29
Enter new 6 char. name of fg 150 ?> = frv30
Enter new 6 char. name of fg 151 ?> = frv31
Enter new 6 char. name of fg 152 ?> = frv32
Enter new 6 char. name of fg 153 ?> = frv33
Enter new 6 char. name of fg 154 ?> = frv34
Enter new 6 char. name of fg 155 ?> = frv35
Enter new 6 char. name of fg 156 ?> = frv36
Enter new 6 char. name of fg 157 ?> = frv37
Enter new 6 char. name of fg 158 ?> = frv38
Enter new 6 char. name of fg 159 ?> = frv39
Enter new 6 char. name of fg 160 ?> = fleak0
Enter new 6 char. name of fg 161 ?> = fleak1
Enter new 6 char. name of fg 162 ?> = fleak2
Enter new 6 char. name of fg 163 ?> = fleak3
Enter new 6 char. name of fg 164 ?> = fleak4
Enter new 6 char. name of fg 165 ?> = fleak5
Enter new 6 char. name of fg 166 ?> = fleak6
Enter new 6 char. name of fg 167 ?> = fleak7
Enter new 6 char. name of fg 168 ?> = fsn0
Enter new 6 char. name of fg 169 ?> = fsn1
Enter new 6 char. name of fg 170 ?> = fsn2
Enter new 6 char. name of fg 171 ?> = fsn3
Enter new 6 char. name of fg 172 ?> = fsn4
Enter new 6 char. name of fg 173 ?> = fsn5
Enter new 6 char. name of fg 174 ?> = fsn6
Enter new 6 char. name of fg 175 ?> = fsn7
Enter new 6 char. name of fg 176 ?> = fsl0
Enter new 6 char. name of fg 177 ?> = fsl1
Enter new 6 char. name of fg 178 ?> = fsl2
Enter new 6 char. name of fg 179 ?> = fsl3
Enter new 6 char. name of fg 180 ?> = fsl4
Enter new 6 char. name of fg 181 ?> = fsl5
Enter new 6 char. name of fg 182 ?> = fsl6
Enter new 6 char. name of fg 183 ?> = fsl7
Enter new 6 char. name of fg 184 ?> = ffarm
Enter new 6 char. name of fg 185 ?> = fspit
Enter new 6 char. name of fg 186 ?> = fvlpit
Enter new 6 char. name of fg 187 ?> = fhvac
Enter new 6 char. name of fg 188 ?> = f271
Enter new 6 char. name of fg 189 ?> = fflush
Enter new 6 char. name of fg 190 ?> = fleak
Enter new 6 char. name of fg 191 ?> = fdisp
Enter new 6 char. name of fg 192 ?> = fhold
Enter new 6 char. name of fg 193 ?> = fning
Enter new 6 char. name of fg 194 ?> = fabort
```

```
Enter new 6 char. name of fg 195 ?> = fstart
Enter new 6 char. name of fg 196 ?> = fpass
Enter new 6 char. name of fg 197 ?> = ftrip0
Enter new 6 char. name of fg 198 ?> = ftrip1
Enter new 6 char. name of fg 199 ?> = ftrip2
Enter new 6 char. name of fg 200 ?> = ftrip3
Enter new 6 char. name of fg 201 ?> = ftrip4
Enter new 6 char. name of fg 202 ?> = fspar0
Enter new 6 char. name of fg 203 ?> = fspar1
Enter new 6 char. name of fg 204 ?> = falmon
Enter new 6 char. name of fg 205 ?> = fack
Enter new 6 char. name of fg 206 ?> = fclear
Enter new 6 char. name of fg 207 ?> = ack
MUPF?>  = add 174 in
Enter new 6 char. name of in 0 ?> = unitt
Enter new 6 char. name of in 1 ?> = analog
Enter new 6 char. name of in 2 ?> = bitval
Enter new 6 char. name of in 3 ?> = lmpval
Enter new 6 char. name of in 4 ?> = skdkey
Enter new 6 char. name of in 5 ?> = iloop
Enter new 6 char. name of in 6 ?> = jloop
Enter new 6 char. name of in 7 ?> = index
Enter new 6 char. name of in 8 ?> = last
Enter new 6 char. name of in 9 ?> = curent
Enter new 6 char. name of in 10 ?> = nxt
Enter new 6 char. name of in 11 ?> = int
Enter new 6 char. name of in 12 ?> = remain
Enter new 6 char. name of in 13 ?> = trip
Enter new 6 char. name of in 14 ?> = nadi
Enter new 6 char. name of in 15 ?> = blkval
Enter new 6 char. name of in 16 ?> = sort
Enter new 6 char. name of in 17 ?> = bval
Enter new 6 char. name of in 18 ?> = hldstp
Enter new 6 char. name of in 19 ?> = errn
Enter new 6 char. name of in 20 ?> = errl
Enter new 6 char. name of in 21 ?> = adi
Enter new 6 char. name of in 22 ?> = star
Enter new 6 char. name of in 23 ?> = white
Enter new 6 char. name of in 24 ?> = valve0
Enter new 6 char. name of in 25 ?> = valve1
Enter new 6 char. name of in 26 ?> = valve2
Enter new 6 char. name of in 27 ?> = valve3
Enter new 6 char. name of in 28 ?> = int0
Enter new 6 char. name of in 29 ?> = int1
Enter new 6 char. name of in 30 ?> = int2
Enter new 6 char. name of in 31 ?> = int3
Enter new 6 char. name of in 32 ?> = int4
Enter new 6 char. name of in 33 ?> = int5
Enter new 6 char. name of in 34 ?> = int6
Enter new 6 char. name of in 35 ?> = int7
```

```
Enter new 6 char. name of in 36 ?> = int8
Enter new 6 char. name of in 37 ?> = int9
Enter new 6 char. name of in 38 ?> = int10
Enter new 6 char. name of in 39 ?> = int11
Enter new 6 char. name of in 40 ?> = int12
Enter new 6 char. name of in 41 ?> = int13
Enter new 6 char. name of in 42 ?> = int14
Enter new 6 char. name of in 43 ?> = int15
Enter new 6 char. name of in 44 ?> = int16
Enter new 6 char. name of in 45 ?> = int17
Enter new 6 char. name of in 46 ?> = int18
Enter new 6 char. name of in 47 ?> = int19
Enter new 6 char. name of in 48 ?> = int20
Enter new 6 char. name of in 49 ?> = int21
Enter new 6 char. name of in 50 ?> = int22
Enter new 6 char. name of in 51 ?> = int23
Enter new 6 char. name of in 52 ?> = int24
Enter new 6 char. name of in 53 ?> = int25
Enter new 6 char. name of in 54 ?> = int26
Enter new 6 char. name of in 55 ?> = int27
Enter new 6 char. name of in 56 ?> = int28
Enter new 6 char. name of in 57 ?> = int29
Enter new 6 char. name of in 58 ?> = int30
Enter new 6 char. name of in 59 ?> = int31
Enter new 6 char. name of in 60 ?> = int32
Enter new 6 char. name of in 61 ?> = int33
Enter new 6 char. name of in 62 ?> = int34
Enter new 6 char. name of in 63 ?> = int35
Enter new 6 char. name of in 64 ?> = int36
Enter new 6 char. name of in 65 ?> = int37
Enter new 6 char. name of in 66 ?> = int38
Enter new 6 char. name of in 67 ?> = int39
Enter new 6 char. name of in 68 ?> = type0
Enter new 6 char. name of in 69 ?> = type1
Enter new 6 char. name of in 70 ?> = type2
Enter new 6 char. name of in 71 ?> = type3
Enter new 6 char. name of in 72 ?> = type4
Enter new 6 char. name of in 73 ?> = type5
Enter new 6 char. name of in 74 ?> = type6
Enter new 6 char. name of in 75 ?> = type7
Enter new 6 char. name of in 76 ?> = type8
Enter new 6 char. name of in 77 ?> = type9
Enter new 6 char. name of in 78 ?> = type10
Enter new 6 char. name of in 79 ?> = type11
Enter new 6 char. name of in 80 ?> = type12
Enter new 6 char. name of in 81 ?> = type13
Enter new 6 char. name of in 82 ?> = type14
Enter new 6 char. name of in 83 ?> = type15
Enter new 6 char. name of in 84 ?> = type16
Enter new 6 char. name of in 85 ?> = type17
```

```
Enter new 6 char. name of in 86 ?> = type18
Enter new 6 char. name of in 87 ?> = type19
Enter new 6 char. name of in 88 ?> = type20
Enter new 6 char. name of in 89 ?> = type21
Enter new 6 char. name of in 90 ?> = type22
Enter new 6 char. name of in 91 ?> = type23
Enter new 6 char. name of in 92 ?> = type24
Enter new 6 char. name of in 93 ?> = type25
Enter new 6 char. name of in 94 ?> = type26
Enter new 6 char. name of in 95 ?> = type27
Enter new 6 char. name of in 96 ?> = type28
Enter new 6 char. name of in 97 ?> = type29
Enter new 6 char. name of in 98 ?> = type30
Enter new 6 char. name of in 99 ?> = type31
Enter new 6 char. name of in 100 ?> = type32
Enter new 6 char. name of in 101 ?> = type33
Enter new 6 char. name of in 102 ?> = type34
Enter new 6 char. name of in 103 ?> = type35
Enter new 6 char. name of in 104 ?> = type36
Enter new 6 char. name of in 105 ?> = type37
Enter new 6 char. name of in 106 ?> = type38
Enter new 6 char. name of in 107 ?> = type39
Enter new 6 char. name of in 108 ?> = skid0
Enter new 6 char. name of in 109 ?> = skid1
Enter new 6 char. name of in 110 ?> = skid2
Enter new 6 char. name of in 111 ?> = skid3
Enter new 6 char. name of in 112 ?> = skid4
Enter new 6 char. name of in 113 ?> = skid5
Enter new 6 char. name of in 114 ?> = skid6
Enter new 6 char. name of in 115 ?> = skid7
Enter new 6 char. name of in 116 ?> = skid8
Enter new 6 char. name of in 117 ?> = skid9
Enter new 6 char. name of in 118 ?> = skid10
Enter new 6 char. name of in 119 ?> = skid11
Enter new 6 char. name of in 120 ?> = skid12
Enter new 6 char. name of in 121 ?> = skid13
Enter new 6 char. name of in 122 ?> = skid14
Enter new 6 char. name of in 123 ?> = skid15
Enter new 6 char. name of in 124 ?> = skid16
Enter new 6 char. name of in 125 ?> = skid17
Enter new 6 char. name of in 126 ?> = skid18
Enter new 6 char. name of in 127 ?> = skid19
Enter new 6 char. name of in 128 ?> = skid20
Enter new 6 char. name of in 129 ?> = skid21
Enter new 6 char. name of in 130 ?> = skid22
Enter new 6 char. name of in 131 ?> = skid23
Enter new 6 char. name of in 132 ?> = skid24
Enter new 6 char. name of in 133 ?> = skid25
Enter new 6 char. name of in 134 ?> = skid26
Enter new 6 char. name of in 135 ?> = skid27
```

```
Enter new 6 char. name of in 136 ?> = skid28
Enter new 6 char. name of in 137 ?> = skid29
Enter new 6 char. name of in 138 ?> = skid30
Enter new 6 char. name of in 139 ?> = skid31
Enter new 6 char. name of in 140 ?> = skid32
Enter new 6 char. name of in 141 ?> = skid33
Enter new 6 char. name of in 142 ?> = skid34
Enter new 6 char. name of in 143 ?> = skid35
Enter new 6 char. name of in 144 ?> = skid36
Enter new 6 char. name of in 145 ?> = skid37
Enter new 6 char. name of in 146 ?> = skid38
Enter new 6 char. name of in 147 ?> = skid39
Enter new 6 char. name of in 148 ?> = dev0
Enter new 6 char. name of in 149 ?> = dev1
Enter new 6 char. name of in 150 ?> = dev2
Enter new 6 char. name of in 151 ?> = dev3
Enter new 6 char. name of in 152 ?> = dev4
Enter new 6 char. name of in 153 ?> = dev5
Enter new 6 char. name of in 154 ?> = dev6
Enter new 6 char. name of in 155 ?> = dev7
Enter new 6 char. name of in 156 ?> = dev8
Enter new 6 char. name of in 157 ?> = dev9
Enter new 6 char. name of in 158 ?> = leak0
Enter new 6 char. name of in 159 ?> = leak1
Enter new 6 char. name of in 160 ?> = leak2
Enter new 6 char. name of in 161 ?> = leak3
Enter new 6 char. name of in 162 ?> = leak4
Enter new 6 char. name of in 163 ?> = leak5
Enter new 6 char. name of in 164 ?> = color0
Enter new 6 char. name of in 165 ?> = color1
Enter new 6 char. name of in 166 ?> = color2
Enter new 6 char. name of in 167 ?> = color3
Enter new 6 char. name of in 168 ?> = color4
Enter new 6 char. name of in 169 ?> = grafx0
Enter new 6 char. name of in 170 ?> = grafx1
Enter new 6 char. name of in 171 ?> = grafx2
Enter new 6 char. name of in 172 ?> = grafx3
Enter new 6 char. name of in 173 ?> = grafx4
MUPF?>  = add 30 fl
Enter new 6 char. name of fl 0 ?> = float0
Enter new 6 char. name of fl 1 ?> = float1
Enter new 6 char. name of fl 2 ?> = float2
Enter new 6 char. name of fl 3 ?> = float3
Enter new 6 char. name of fl 4 ?> = float4
Enter new 6 char. name of fl 5 ?> = float5
Enter new 6 char. name of fl 6 ?> = float6
Enter new 6 char. name of fl 7 ?> = float7
Enter new 6 char. name of fl 8 ?> = float8
Enter new 6 char. name of fl 9 ?> = float9
Enter new 6 char. name of fl 10 ?> = ai0
```

```
Enter new 6 char. name of fl 11 ?> = ai1
Enter new 6 char. name of fl 12 ?> = ai2
Enter new 6 char. name of fl 13 ?> = ai3
Enter new 6 char. name of fl 14 ?> = ai4
Enter new 6 char. name of fl 15 ?> = ai5
Enter new 6 char. name of fl 16 ?> = ai6
Enter new 6 char. name of fl 17 ?> = ai7
Enter new 6 char. name of fl 18 ?> = ai8
Enter new 6 char. name of fl 19 ?> = ai9
Enter new 6 char. name of fl 20 ?> = ai10
Enter new 6 char. name of fl 21 ?> = ai11
Enter new 6 char. name of fl 22 ?> = ai12
Enter new 6 char. name of fl 23 ?> = ai13
Enter new 6 char. name of fl 24 ?> = ai14
Enter new 6 char. name of fl 25 ?> = ai15
Enter new 6 char. name of fl 26 ?> = ai16
Enter new 6 char. name of fl 27 ?> = ai17
Enter new 6 char. name of fl 28 ?> = ai18
Enter new 6 char. name of fl 29 ?> = ai19
MUPF?>  = add 10 pt
Enter new 6 char. name of pt 0 ?> = pepn
Enter new 6 char. name of pt 1 ?> = pdin
Enter new 6 char. name of pt 2 ?> = pdev
Enter new 6 char. name of pt 3 ?> = pai
Enter new 6 char. name of pt 4 ?> = palarm
Enter new 6 char. name of pt 5 ?> = pskid
Enter new 6 char. name of pt 6 ?> = pstat
Enter new 6 char. name of pt 7 ?> = ptype
Enter new 6 char. name of pt 8 ?> = ppass
Enter new 6 char. name of pt 9 ?> = prev
MUPF?>  = add 35 sg
Enter new 6 char. name of sg 0 ?> = swhen0
Enter max string length for sg 0  ?> = 8
Enter new 6 char. name of sg 1
Enter new 6 char. name of sg 1 ?> = swhen1
Enter max string length for sg 1  ?> = 8
Enter new 6 char. name of sg 2 ?> = swhen2
Enter max string length for sg 2  ?> = 8
Enter new 6 char. name of sg 3 ?> = swhen3
Enter max string length for sg 3  ?> = 8
Enter new 6 char. name of sg 4 ?> = swhen4
Enter max string length for sg 4  ?> = 8
Enter new 6 char. name of sg 5 ?> = ssng0
Enter max string length for sg 5  ?> = 8
Enter new 6 char. name of sg 6 ?> = ssng1
Enter max string length for sg 6  ?> = 8
Enter new 6 char. name of sg 7 ?> = ssng2
Enter max string length for sg 7  ?> = 8
Enter new 6 char. name of sg 8 ?> = ssng3
Enter max string length for sg 8  ?> = 8
```

```
Enter new 6 char. name of sg 9 ?> = ssng4
Enter max string length for sg 9  ?> = 8
Enter new 6 char. name of sg 10 ?> = sdate0
Enter max string length for sg 10  ?> = 10
Enter new 6 char. name of sg 11 ?> = sdate1
Enter max string length for sg 11  ?> = 10
Enter new 6 char. name of sg 12 ?> = sdate2
Enter max string length for sg 12  ?> = 10
Enter new 6 char. name of sg 13 ?> = sdate3
Enter max string length for sg 13  ?> = 10
Enter new 6 char. name of sg 14 ?> = sdate4
Enter max string length for sg 14  ?> = 10
Enter new 6 char. name of sg 15 ?> = epns0
Enter max string length for sg 15  ?> = 10
Enter new 6 char. name of sg 16 ?> = epns1
Enter max string length for sg 16  ?> = 10
Enter new 6 char. name of sg 17 ?> = epns2
Enter max string length for sg 17  ?> = 10
Enter new 6 char. name of sg 18 ?> = epns3
Enter max string length for sg 18  ?> = 10
Enter new 6 char. name of sg 19 ?> = epns4
Enter max string length for sg 19  ?> = 10
Enter new 6 char. name of sg 20 ?> = salm0
Enter max string length for sg 20  ?> = 20
Enter new 6 char. name of sg 21 ?> = salm1
Enter max string length for sg 21  ?> = 20
Enter new 6 char. name of sg 22 ?> = salm2
Enter max string length for sg 22  ?> = 20
Enter new 6 char. name of sg 23 ?> = salm3
Enter max string length for sg 23  ?> = 20
Enter new 6 char. name of sg 24 ?> = salm4
Enter max string length for sg 24  ?> = 20
Enter new 6 char. name of sg 25 ?> = lsg0
Enter max string length for sg 25  ?> = 30
Enter new 6 char. name of sg 26 ?> = lsg1
Enter max string length for sg 26  ?> = 30
Enter new 6 char. name of sg 27 ?> = lsg2
Enter max string length for sg 27  ?> = 30
Enter new 6 char. name of sg 28 ?> = lsg3
Enter max string length for sg 28  ?> = 30
Enter new 6 char. name of sg 29 ?> = lsg4
Enter max string length for sg 29  ?> = 30
Enter new 6 char. name of sg 30 ?> = strng0
Enter max string length for sg 30  ?> = 30
Enter new 6 char. name of sg 31 ?> = strng1
Enter max string length for sg 31  ?> = 30
Enter new 6 char. name of sg 32 ?> = strng2
Enter max string length for sg 32  ?> = 30
Enter new 6 char. name of sg 33 ?> = strng3
Enter max string length for sg 33  ?> = 30
```

```
Enter new 6 char. name of sg 34 ?> = strng4
Enter max string length for sg 34  ?> = 30
MUPF?>  =
UPF?>
 end of file encountered
```

PRCESS

```
UPF Version 6.3-2       05-Dec-94 09:36:16
UPF?>  = de prcess
UPF?>  = cr prcess,*
Enter 4 character name of RPF to associate?> = 242a
MUPF?>  = add 368 fg
Enter new 6 char. name of fg 0 ?> = falm0
Enter new 6 char. name of fg 1 ?> = falm1
Enter new 6 char. name of fg 2 ?> = falm2
Enter new 6 char. name of fg 3 ?> = falm3
Enter new 6 char. name of fg 4 ?> = falm4
Enter new 6 char. name of fg 5 ?> = falm5
Enter new 6 char. name of fg 6 ?> = falm6
Enter new 6 char. name of fg 7 ?> = falm7
Enter new 6 char. name of fg 8 ?> = falm8
Enter new 6 char. name of fg 9 ?> = falm9
Enter new 6 char. name of fg 10 ?> = falm10
Enter new 6 char. name of fg 11 ?> = falm11
Enter new 6 char. name of fg 12 ?> = falm12
Enter new 6 char. name of fg 13 ?> = falm13
Enter new 6 char. name of fg 14 ?> = falm14
Enter new 6 char. name of fg 15 ?> = falm15
Enter new 6 char. name of fg 16 ?> = falm16
Enter new 6 char. name of fg 17 ?> = falm17
Enter new 6 char. name of fg 18 ?> = falm18
Enter new 6 char. name of fg 19 ?> = falm19
Enter new 6 char. name of fg 20 ?> = falm20
Enter new 6 char. name of fg 21 ?> = falm21
Enter new 6 char. name of fg 22 ?> = falm22
Enter new 6 char. name of fg 23 ?> = falm23
Enter new 6 char. name of fg 24 ?> = falm24
Enter new 6 char. name of fg 25 ?> = falm25
Enter new 6 char. name of fg 26 ?> = falm26
Enter new 6 char. name of fg 27 ?> = falm27
Enter new 6 char. name of fg 28 ?> = falm28
Enter new 6 char. name of fg 29 ?> = falm29
Enter new 6 char. name of fg 30 ?> = falm30
Enter new 6 char. name of fg 31 ?> = falm31
Enter new 6 char. name of fg 32 ?> = falm32
Enter new 6 char. name of fg 33 ?> = falm33
Enter new 6 char. name of fg 34 ?> = falm34
Enter new 6 char. name of fg 35 ?> = falm35
Enter new 6 char. name of fg 36 ?> = falm36
Enter new 6 char. name of fg 37 ?> = falm37
Enter new 6 char. name of fg 38 ?> = falm38
Enter new 6 char. name of fg 39 ?> = falm39
Enter new 6 char. name of fg 40 ?> = falm40
Enter new 6 char. name of fg 41 ?> = falm41
Enter new 6 char. name of fg 42 ?> = falm42
Enter new 6 char. name of fg 43 ?> = falm43
Enter new 6 char. name of fg 44 ?> = falm44
```

```
Enter new 6 char. name of fg 45 ?> = falm45
Enter new 6 char. name of fg 46 ?> = falm46
Enter new 6 char. name of fg 47 ?> = falm47
Enter new 6 char. name of fg 48 ?> = falm48
Enter new 6 char. name of fg 49 ?> = falm49
Enter new 6 char. name of fg 50 ?> = falm50
Enter new 6 char. name of fg 51 ?> = falm51
Enter new 6 char. name of fg 52 ?> = falm52
Enter new 6 char. name of fg 53 ?> = falm53
Enter new 6 char. name of fg 54 ?> = falm54
Enter new 6 char. name of fg 55 ?> = falm55
Enter new 6 char. name of fg 56 ?> = falm56
Enter new 6 char. name of fg 57 ?> = falm57
Enter new 6 char. name of fg 58 ?> = falm58
Enter new 6 char. name of fg 59 ?> = falm59
Enter new 6 char. name of fg 60 ?> = falm60
Enter new 6 char. name of fg 61 ?> = falm61
Enter new 6 char. name of fg 62 ?> = falm62
Enter new 6 char. name of fg 63 ?> = fpas0
Enter new 6 char. name of fg 64 ?> = fpas1
Enter new 6 char. name of fg 65 ?> = fpas2
Enter new 6 char. name of fg 66 ?> = fpas3
Enter new 6 char. name of fg 67 ?> = fpas4
Enter new 6 char. name of fg 68 ?> = fpas5
Enter new 6 char. name of fg 69 ?> = fpas6
Enter new 6 char. name of fg 70 ?> = fpas7
Enter new 6 char. name of fg 71 ?> = fpas8
Enter new 6 char. name of fg 72 ?> = fpas9
Enter new 6 char. name of fg 73 ?> = fpas10
Enter new 6 char. name of fg 74 ?> = fpas11
Enter new 6 char. name of fg 75 ?> = fpas12
Enter new 6 char. name of fg 76 ?> = fpas13
Enter new 6 char. name of fg 77 ?> = fpas14
Enter new 6 char. name of fg 78 ?> = fpas15
Enter new 6 char. name of fg 79 ?> = fpas16
Enter new 6 char. name of fg 80 ?> = fpas17
Enter new 6 char. name of fg 81 ?> = fpas18
Enter new 6 char. name of fg 82 ?> = fpas19
Enter new 6 char. name of fg 83 ?> = fpas20
Enter new 6 char. name of fg 84 ?> = fpas21
Enter new 6 char. name of fg 85 ?> = fpas22
Enter new 6 char. name of fg 86 ?> = fpas23
Enter new 6 char. name of fg 87 ?> = fpas24
Enter new 6 char. name of fg 88 ?> = fpas25
Enter new 6 char. name of fg 89 ?> = fpas26
Enter new 6 char. name of fg 90 ?> = fpas27
Enter new 6 char. name of fg 91 ?> = fpas28
Enter new 6 char. name of fg 92 ?> = fpas29
Enter new 6 char. name of fg 93 ?> = fpas30
Enter new 6 char. name of fg 94 ?> = fpas31
```

```
Enter new 6 char. name of fg 95 ?> = fpas32
Enter new 6 char. name of fg 96 ?> = fpas33
Enter new 6 char. name of fg 97 ?> = fpas34
Enter new 6 char. name of fg 98 ?> = fpas35
Enter new 6 char. name of fg 99 ?> = fpas36
Enter new 6 char. name of fg 100 ?> = fpas37
Enter new 6 char. name of fg 101 ?> = fpas38
Enter new 6 char. name of fg 102 ?> = fpas39
Enter new 6 char. name of fg 103 ?> = fpas40
Enter new 6 char. name of fg 104 ?> = fpas41
Enter new 6 char. name of fg 105 ?> = fpas42
Enter new 6 char. name of fg 106 ?> = fpas43
Enter new 6 char. name of fg 107 ?> = fpas44
Enter new 6 char. name of fg 108 ?> = fpas45
Enter new 6 char. name of fg 109 ?> = fpas46
Enter new 6 char. name of fg 110 ?> = fpas47
Enter new 6 char. name of fg 111 ?> = fpas48
Enter new 6 char. name of fg 112 ?> = fpas49
Enter new 6 char. name of fg 113 ?> = fpas50
Enter new 6 char. name of fg 114 ?> = fpas51
Enter new 6 char. name of fg 115 ?> = fpas52
Enter new 6 char. name of fg 116 ?> = fpas53
Enter new 6 char. name of fg 117 ?> = fpas54
Enter new 6 char. name of fg 118 ?> = fpas55
Enter new 6 char. name of fg 119 ?> = fpas56
Enter new 6 char. name of fg 120 ?> = fpas57
Enter new 6 char. name of fg 121 ?> = fpas58
Enter new 6 char. name of fg 122 ?> = fpas59
Enter new 6 char. name of fg 123 ?> = fpas60
Enter new 6 char. name of fg 124 ?> = fpas61
Enter new 6 char. name of fg 125 ?> = fpas62
Enter new 6 char. name of fg 126 ?> = fpas63
Enter new 6 char. name of fg 127 ?> = frv0
Enter new 6 char. name of fg 128 ?> = frv1
Enter new 6 char. name of fg 129 ?> = frv2
Enter new 6 char. name of fg 130 ?> = frv3
Enter new 6 char. name of fg 131 ?> = frv4
Enter new 6 char. name of fg 132 ?> = frv5
Enter new 6 char. name of fg 133 ?> = frv6
Enter new 6 char. name of fg 134 ?> = frv7
Enter new 6 char. name of fg 135 ?> = frv8
Enter new 6 char. name of fg 136 ?> = frv9
Enter new 6 char. name of fg 137 ?> = frv10
Enter new 6 char. name of fg 138 ?> = frv11
Enter new 6 char. name of fg 139 ?> = frv12
Enter new 6 char. name of fg 140 ?> = frv13
Enter new 6 char. name of fg 141 ?> = frv14
Enter new 6 char. name of fg 142 ?> = frv15
Enter new 6 char. name of fg 143 ?> = frv16
Enter new 6 char. name of fg 144 ?> = frv17
```

```
Enter new 6 char. name of fg 145 ?> = frv18
Enter new 6 char. name of fg 146 ?> = frv19
Enter new 6 char. name of fg 147 ?> = frv20
Enter new 6 char. name of fg 148 ?> = frv21
Enter new 6 char. name of fg 149 ?> = frv22
Enter new 6 char. name of fg 150 ?> = frv23
Enter new 6 char. name of fg 151 ?> = frv24
Enter new 6 char. name of fg 152 ?> = frv25
Enter new 6 char. name of fg 153 ?> = frv26
Enter new 6 char. name of fg 154 ?> = frv27
Enter new 6 char. name of fg 155 ?> = frv28
Enter new 6 char. name of fg 156 ?> = frv29
Enter new 6 char. name of fg 157 ?> = frv30
Enter new 6 char. name of fg 158 ?> = frv31
Enter new 6 char. name of fg 159 ?> = frv32
Enter new 6 char. name of fg 160 ?> = frv33
Enter new 6 char. name of fg 161 ?> = frv34
Enter new 6 char. name of fg 162 ?> = frv35
Enter new 6 char. name of fg 163 ?> = frv36
Enter new 6 char. name of fg 164 ?> = frv37
Enter new 6 char. name of fg 165 ?> = frv38
Enter new 6 char. name of fg 166 ?> = frv39
Enter new 6 char. name of fg 167 ?> = frv40
Enter new 6 char. name of fg 168 ?> = frv41
Enter new 6 char. name of fg 169 ?> = frv42
Enter new 6 char. name of fg 170 ?> = frv43
Enter new 6 char. name of fg 171 ?> = frv44
Enter new 6 char. name of fg 172 ?> = frv45
Enter new 6 char. name of fg 173 ?> = frv46
Enter new 6 char. name of fg 174 ?> = frv47
Enter new 6 char. name of fg 175 ?> = frv48
Enter new 6 char. name of fg 176 ?> = frv49
Enter new 6 char. name of fg 177 ?> = frv50
Enter new 6 char. name of fg 178 ?> = frv51
Enter new 6 char. name of fg 179 ?> = frv52
Enter new 6 char. name of fg 180 ?> = frv53
Enter new 6 char. name of fg 181 ?> = frv54
Enter new 6 char. name of fg 182 ?> = frv55
Enter new 6 char. name of fg 183 ?> = frv56
Enter new 6 char. name of fg 184 ?> = frv57
Enter new 6 char. name of fg 185 ?> = frv58
Enter new 6 char. name of fg 186 ?> = frv59
Enter new 6 char. name of fg 187 ?> = frv60
Enter new 6 char. name of fg 188 ?> = frv61
Enter new 6 char. name of fg 189 ?> = frv62
Enter new 6 char. name of fg 190 ?> = frv63
Enter new 6 char. name of fg 191 ?> = aleak0
Enter new 6 char. name of fg 192 ?> = aleak1
Enter new 6 char. name of fg 193 ?> = aleak2
Enter new 6 char. name of fg 194 ?> = aleak3
```

```
Enter new 6 char. name of fg 195 ?> = bleak0
Enter new 6 char. name of fg 196 ?> = bleak1
Enter new 6 char. name of fg 197 ?> = bleak2
Enter new 6 char. name of fg 198 ?> = bleak3
Enter new 6 char. name of fg 199 ?> = cleak0
Enter new 6 char. name of fg 200 ?> = cleak1
Enter new 6 char. name of fg 201 ?> = cleak2
Enter new 6 char. name of fg 202 ?> = cleak3
Enter new 6 char. name of fg 203 ?> = dleak0
Enter new 6 char. name of fg 204 ?> = dleak1
Enter new 6 char. name of fg 205 ?> = dleak2
Enter new 6 char. name of fg 206 ?> = dleak3
Enter new 6 char. name of fg 207 ?> = at0
Enter new 6 char. name of fg 208 ?> = at1
Enter new 6 char. name of fg 209 ?> = at2
Enter new 6 char. name of fg 210 ?> = at3
Enter new 6 char. name of fg 211 ?> = at4
Enter new 6 char. name of fg 212 ?> = at5
Enter new 6 char. name of fg 213 ?> = at6
Enter new 6 char. name of fg 214 ?> = at7
Enter new 6 char. name of fg 215 ?> = at8
Enter new 6 char. name of fg 216 ?> = at9
Enter new 6 char. name of fg 217 ?> = at10
Enter new 6 char. name of fg 218 ?> = at11
Enter new 6 char. name of fg 219 ?> = at12
Enter new 6 char. name of fg 220 ?> = at13
Enter new 6 char. name of fg 221 ?> = at14
Enter new 6 char. name of fg 222 ?> = at15
Enter new 6 char. name of fg 223 ?> = at16
Enter new 6 char. name of fg 224 ?> = at17
Enter new 6 char. name of fg 225 ?> = at18
Enter new 6 char. name of fg 226 ?> = at19
Enter new 6 char. name of fg 227 ?> = at20
Enter new 6 char. name of fg 228 ?> = at21
Enter new 6 char. name of fg 229 ?> = at22
Enter new 6 char. name of fg 230 ?> = at23
Enter new 6 char. name of fg 231 ?> = at24
Enter new 6 char. name of fg 232 ?> = at25
Enter new 6 char. name of fg 233 ?> = at26
Enter new 6 char. name of fg 234 ?> = at27
Enter new 6 char. name of fg 235 ?> = at28
Enter new 6 char. name of fg 236 ?> = at29
Enter new 6 char. name of fg 237 ?> = at30
Enter new 6 char. name of fg 238 ?> = at31
Enter new 6 char. name of fg 239 ?> = bt0
Enter new 6 char. name of fg 240 ?> = bt1
Enter new 6 char. name of fg 241 ?> = bt2
Enter new 6 char. name of fg 242 ?> = bt3
Enter new 6 char. name of fg 243 ?> = bt4
Enter new 6 char. name of fg 244 ?> = bt5
```

```
Enter new 6 char. name of fg 245 ?> = bt6
Enter new 6 char. name of fg 246 ?> = bt7
Enter new 6 char. name of fg 247 ?> = bt8
Enter new 6 char. name of fg 248 ?> = bt9
Enter new 6 char. name of fg 249 ?> = bt10
Enter new 6 char. name of fg 250 ?> = bt11
Enter new 6 char. name of fg 251 ?> = bt12
Enter new 6 char. name of fg 252 ?> = bt13
Enter new 6 char. name of fg 253 ?> = bt14
Enter new 6 char. name of fg 254 ?> = bt15
Enter new 6 char. name of fg 255 ?> = bt16
Enter new 6 char. name of fg 256 ?> = bt17
Enter new 6 char. name of fg 257 ?> = bt18
Enter new 6 char. name of fg 258 ?> = bt19
Enter new 6 char. name of fg 259 ?> = bt20
Enter new 6 char. name of fg 260 ?> = bt21
Enter new 6 char. name of fg 261 ?> = bt22
Enter new 6 char. name of fg 262 ?> = bt23
Enter new 6 char. name of fg 263 ?> = bt24
Enter new 6 char. name of fg 264 ?> = bt25
Enter new 6 char. name of fg 265 ?> = bt26
Enter new 6 char. name of fg 266 ?> = bt27
Enter new 6 char. name of fg 267 ?> = bt28
Enter new 6 char. name of fg 268 ?> = bt29
Enter new 6 char. name of fg 269 ?> = bt30
Enter new 6 char. name of fg 270 ?> = bt31
Enter new 6 char. name of fg 271 ?> = ct0
Enter new 6 char. name of fg 272 ?> = ct1
Enter new 6 char. name of fg 273 ?> = ct2
Enter new 6 char. name of fg 274 ?> = ct3
Enter new 6 char. name of fg 275 ?> = ct4
Enter new 6 char. name of fg 276 ?> = ct5
Enter new 6 char. name of fg 277 ?> = ct6
Enter new 6 char. name of fg 278 ?> = ct7
Enter new 6 char. name of fg 279 ?> = ct8
Enter new 6 char. name of fg 280 ?> = ct9
Enter new 6 char. name of fg 281 ?> = ct10
Enter new 6 char. name of fg 282 ?> = ct11
Enter new 6 char. name of fg 283 ?> = ct12
Enter new 6 char. name of fg 284 ?> = ct13
Enter new 6 char. name of fg 285 ?> = ct14
Enter new 6 char. name of fg 286 ?> = ct15
Enter new 6 char. name of fg 287 ?> = ct16
Enter new 6 char. name of fg 288 ?> = ct17
Enter new 6 char. name of fg 289 ?> = ct18
Enter new 6 char. name of fg 290 ?> = ct19
Enter new 6 char. name of fg 291 ?> = ct20
Enter new 6 char. name of fg 292 ?> = ct21
Enter new 6 char. name of fg 293 ?> = ct22
Enter new 6 char. name of fg 294 ?> = ct23
```

```
Enter new 6 char. name of fg 295 ?> = ct24
Enter new 6 char. name of fg 296 ?> = ct25
Enter new 6 char. name of fg 297 ?> = ct26
Enter new 6 char. name of fg 298 ?> = ct27
Enter new 6 char. name of fg 299 ?> = ct28
Enter new 6 char. name of fg 300 ?> = ct29
Enter new 6 char. name of fg 301 ?> = ct30
Enter new 6 char. name of fg 302 ?> = ct31
Enter new 6 char. name of fg 303 ?> = sa0
Enter new 6 char. name of fg 304 ?> = sa1
Enter new 6 char. name of fg 305 ?> = sa2
Enter new 6 char. name of fg 306 ?> = sa3
Enter new 6 char. name of fg 307 ?> = sa4
Enter new 6 char. name of fg 308 ?> = sa5
Enter new 6 char. name of fg 309 ?> = sa6
Enter new 6 char. name of fg 310 ?> = sa7
Enter new 6 char. name of fg 311 ?> = sb0
Enter new 6 char. name of fg 312 ?> = sb1
Enter new 6 char. name of fg 313 ?> = sb2
Enter new 6 char. name of fg 314 ?> = sb3
Enter new 6 char. name of fg 315 ?> = sb4
Enter new 6 char. name of fg 316 ?> = sb5
Enter new 6 char. name of fg 317 ?> = sb6
Enter new 6 char. name of fg 318 ?> = sb7
Enter new 6 char. name of fg 319 ?> = sc0
Enter new 6 char. name of fg 320 ?> = sc1
Enter new 6 char. name of fg 321 ?> = sc2
Enter new 6 char. name of fg 322 ?> = sc3
Enter new 6 char. name of fg 323 ?> = sc4
Enter new 6 char. name of fg 324 ?> = sc5
Enter new 6 char. name of fg 325 ?> = sc6
Enter new 6 char. name of fg 326 ?> = sc7
Enter new 6 char. name of fg 327 ?> = sd0
Enter new 6 char. name of fg 328 ?> = sd1
Enter new 6 char. name of fg 329 ?> = sd2
Enter new 6 char. name of fg 330 ?> = sd3
Enter new 6 char. name of fg 331 ?> = sd4
Enter new 6 char. name of fg 332 ?> = sd5
Enter new 6 char. name of fg 333 ?> = sd6
Enter new 6 char. name of fg 334 ?> = sd7
Enter new 6 char. name of fg 335 ?> = gt0
Enter new 6 char. name of fg 336 ?> = gt1
Enter new 6 char. name of fg 337 ?> = gt2
Enter new 6 char. name of fg 338 ?> = gt3
Enter new 6 char. name of fg 339 ?> = gt4
Enter new 6 char. name of fg 340 ?> = gt5
Enter new 6 char. name of fg 341 ?> = gt6
Enter new 6 char. name of fg 342 ?> = gt7
Enter new 6 char. name of fg 343 ?> = gt8
Enter new 6 char. name of fg 344 ?> = gt9
```

```
Enter new 6 char. name of fg 345 ?> = gtl0
Enter new 6 char. name of fg 346 ?> = gtl1
Enter new 6 char. name of fg 347 ?> = gtl2
Enter new 6 char. name of fg 348 ?> = gtl3
Enter new 6 char. name of fg 349 ?> = gtl4
Enter new 6 char. name of fg 350 ?> = gtl5
Enter new 6 char. name of fg 351 ?> = fhold
Enter new 6 char. name of fg 352 ?> = fning
Enter new 6 char. name of fg 353 ?> = fabort
Enter new 6 char. name of fg 354 ?> = fstart
Enter new 6 char. name of fg 355 ?> = fpass
Enter new 6 char. name of fg 356 ?> = ftrip0
Enter new 6 char. name of fg 357 ?> = ftrip1
Enter new 6 char. name of fg 358 ?> = ftrip2
Enter new 6 char. name of fg 359 ?> = ftrip3
Enter new 6 char. name of fg 360 ?> = ftrip4
Enter new 6 char. name of fg 361 ?> = fspar0
Enter new 6 char. name of fg 362 ?> = fspar1
Enter new 6 char. name of fg 363 ?> = fsn0
Enter new 6 char. name of fg 364 ?> = falmon
Enter new 6 char. name of fg 365 ?> = fack
Enter new 6 char. name of fg 366 ?> = fclear
Enter new 6 char. name of fg 367 ?> = ack
MUPF?>  = add 170 in
Enter new 6 char. name of in 0 ?> = unitt
Enter new 6 char. name of in 1 ?> = analog
Enter new 6 char. name of in 2 ?> = white
Enter new 6 char. name of in 3 ?> = bitval
Enter new 6 char. name of in 4 ?> = lmpvar
Enter new 6 char. name of in 5 ?> = skdkey
Enter new 6 char. name of in 6 ?> = iloop
Enter new 6 char. name of in 7 ?> = jloop
Enter new 6 char. name of in 8 ?> = index
Enter new 6 char. name of in 9 ?> = last
Enter new 6 char. name of in 10 ?> = curent
Enter new 6 char. name of in 11 ?> = nxt
Enter new 6 char. name of in 12 ?> = int
Enter new 6 char. name of in 13 ?> = remain
Enter new 6 char. name of in 14 ?> = trip
Enter new 6 char. name of in 15 ?> = nadi
Enter new 6 char. name of in 16 ?> = blkval
Enter new 6 char. name of in 17 ?> = sort
Enter new 6 char. name of in 18 ?> = bval
Enter new 6 char. name of in 19 ?> = hldstp
Enter new 6 char. name of in 20 ?> = errn
Enter new 6 char. name of in 21 ?> = errl
Enter new 6 char. name of in 22 ?> = adi
Enter new 6 char. name of in 23 ?> = star
Enter new 6 char. name of in 24 ?> = dev0
Enter new 6 char. name of in 25 ?> = dev1
```

```
Enter new 6 char. name of in 26 ?> = dev2
Enter new 6 char. name of in 27 ?> = dev3
Enter new 6 char. name of in 28 ?> = dev4
Enter new 6 char. name of in 29 ?> = dev5
Enter new 6 char. name of in 30 ?> = dev6
Enter new 6 char. name of in 31 ?> = dev7
Enter new 6 char. name of in 32 ?> = dev8
Enter new 6 char. name of in 33 ?> = dev9
Enter new 6 char. name of in 34 ?> = skid0
Enter new 6 char. name of in 35 ?> = skid1
Enter new 6 char. name of in 36 ?> = skid2
Enter new 6 char. name of in 37 ?> = skid3
Enter new 6 char. name of in 38 ?> = skid4
Enter new 6 char. name of in 39 ?> = skid5
Enter new 6 char. name of in 40 ?> = skid6
Enter new 6 char. name of in 41 ?> = skid7
Enter new 6 char. name of in 42 ?> = skid8
Enter new 6 char. name of in 43 ?> = skid9
Enter new 6 char. name of in 44 ?> = skid10
Enter new 6 char. name of in 45 ?> = skid11
Enter new 6 char. name of in 46 ?> = skid12
Enter new 6 char. name of in 47 ?> = skid13
Enter new 6 char. name of in 48 ?> = skid14
Enter new 6 char. name of in 49 ?> = skid15
Enter new 6 char. name of in 50 ?> = skid16
Enter new 6 char. name of in 51 ?> = skid17
Enter new 6 char. name of in 52 ?> = skid18
Enter new 6 char. name of in 53 ?> = skid19
Enter new 6 char. name of in 54 ?> = skid20
Enter new 6 char. name of in 55 ?> = skid21
Enter new 6 char. name of in 56 ?> = skid22
Enter new 6 char. name of in 57 ?> = skid23
Enter new 6 char. name of in 58 ?> = skid24
Enter new 6 char. name of in 59 ?> = skid25
Enter new 6 char. name of in 60 ?> = skid26
Enter new 6 char. name of in 61 ?> = skid27
Enter new 6 char. name of in 62 ?> = skid28
Enter new 6 char. name of in 63 ?> = skid29
Enter new 6 char. name of in 64 ?> = skid30
Enter new 6 char. name of in 65 ?> = skid31
Enter new 6 char. name of in 66 ?> = skid32
Enter new 6 char. name of in 67 ?> = skid33
Enter new 6 char. name of in 68 ?> = skid34
Enter new 6 char. name of in 69 ?> = skid35
Enter new 6 char. name of in 70 ?> = skid36
Enter new 6 char. name of in 71 ?> = skid37
Enter new 6 char. name of in 72 ?> = skid38
Enter new 6 char. name of in 73 ?> = skid39
Enter new 6 char. name of in 74 ?> = type0
Enter new 6 char. name of in 75 ?> = type1
```

```
Enter new 6 char. name of in 76 ?> = type2
Enter new 6 char. name of in 77 ?> = type3
Enter new 6 char. name of in 78 ?> = type4
Enter new 6 char. name of in 79 ?> = type5
Enter new 6 char. name of in 80 ?> = type6
Enter new 6 char. name of in 81 ?> = type7
Enter new 6 char. name of in 82 ?> = type8
Enter new 6 char. name of in 83 ?> = type9
Enter new 6 char. name of in 84 ?> = type10
Enter new 6 char. name of in 85 ?> = type11
Enter new 6 char. name of in 86 ?> = type12
Enter new 6 char. name of in 87 ?> = type13
Enter new 6 char. name of in 88 ?> = type14
Enter new 6 char. name of in 89 ?> = type15
Enter new 6 char. name of in 90 ?> = type16
Enter new 6 char. name of in 91 ?> = type17
Enter new 6 char. name of in 92 ?> = type18
Enter new 6 char. name of in 93 ?> = type19
Enter new 6 char. name of in 94 ?> = type20
Enter new 6 char. name of in 95 ?> = type21
Enter new 6 char. name of in 96 ?> = type22
Enter new 6 char. name of in 97 ?> = type23
Enter new 6 char. name of in 98 ?> = type24
Enter new 6 char. name of in 99 ?> = type25
Enter new 6 char. name of in 100 ?> = type26
Enter new 6 char. name of in 101 ?> = type27
Enter new 6 char. name of in 102 ?> = type28
Enter new 6 char. name of in 103 ?> = type29
Enter new 6 char. name of in 104 ?> = type30
Enter new 6 char. name of in 105 ?> = type31
Enter new 6 char. name of in 106 ?> = type32
Enter new 6 char. name of in 107 ?> = type33
Enter new 6 char. name of in 108 ?> = type34
Enter new 6 char. name of in 109 ?> = type35
Enter new 6 char. name of in 110 ?> = type36
Enter new 6 char. name of in 111 ?> = type37
Enter new 6 char. name of in 112 ?> = type38
Enter new 6 char. name of in 113 ?> = type39
Enter new 6 char. name of in 114 ?> = int0
Enter new 6 char. name of in 115 ?> = int1
Enter new 6 char. name of in 116 ?> = int2
Enter new 6 char. name of in 117 ?> = int3
Enter new 6 char. name of in 118 ?> = int4
Enter new 6 char. name of in 119 ?> = int5
Enter new 6 char. name of in 120 ?> = int6
Enter new 6 char. name of in 121 ?> = int7
Enter new 6 char. name of in 122 ?> = int8
Enter new 6 char. name of in 123 ?> = int9
Enter new 6 char. name of in 124 ?> = int10
Enter new 6 char. name of in 125 ?> = int11
```

```
Enter new 6 char. name of in 126 ?> = int12
Enter new 6 char. name of in 127 ?> = int13
Enter new 6 char. name of in 128 ?> = int14
Enter new 6 char. name of in 129 ?> = int15
Enter new 6 char. name of in 130 ?> = int16
Enter new 6 char. name of in 131 ?> = int17
Enter new 6 char. name of in 132 ?> = int18
Enter new 6 char. name of in 133 ?> = int19
Enter new 6 char. name of in 134 ?> = int20
Enter new 6 char. name of in 135 ?> = int21
Enter new 6 char. name of in 136 ?> = int22
Enter new 6 char. name of in 137 ?> = int23
Enter new 6 char. name of in 138 ?> = int24
Enter new 6 char. name of in 139 ?> = int25
Enter new 6 char. name of in 140 ?> = int26
Enter new 6 char. name of in 141 ?> = int27
Enter new 6 char. name of in 142 ?> = int28
Enter new 6 char. name of in 143 ?> = int29
Enter new 6 char. name of in 144 ?> = int30
Enter new 6 char. name of in 145 ?> = int31
Enter new 6 char. name of in 146 ?> = int32
Enter new 6 char. name of in 147 ?> = int33
Enter new 6 char. name of in 148 ?> = int34
Enter new 6 char. name of in 149 ?> = int35
Enter new 6 char. name of in 150 ?> = int36
Enter new 6 char. name of in 151 ?> = int37
Enter new 6 char. name of in 152 ?> = int38
Enter new 6 char. name of in 153 ?> = int39
Enter new 6 char. name of in 154 ?> = leak0
Enter new 6 char. name of in 155 ?> = leak1
Enter new 6 char. name of in 156 ?> = leak2
Enter new 6 char. name of in 157 ?> = leak3
Enter new 6 char. name of in 158 ?> = leak4
Enter new 6 char. name of in 159 ?> = leak5
Enter new 6 char. name of in 160 ?> = color0
Enter new 6 char. name of in 161 ?> = color1
Enter new 6 char. name of in 162 ?> = color2
Enter new 6 char. name of in 163 ?> = color3
Enter new 6 char. name of in 164 ?> = color4
Enter new 6 char. name of in 165 ?> = grafx0
Enter new 6 char. name of in 166 ?> = grafx1
Enter new 6 char. name of in 167 ?> = grafx2
Enter new 6 char. name of in 168 ?> = grafx3
Enter new 6 char. name of in 169 ?> = grafx4
MUPF?>  = add 36 fl
Enter new 6 char. name of fl 0 ?> = ai0
Enter new 6 char. name of fl 1 ?> = ai1
Enter new 6 char. name of fl 2 ?> = ai2
Enter new 6 char. name of fl 3 ?> = ai3
Enter new 6 char. name of fl 4 ?> = ai4
```

```
Enter new 6 char. name of fl 5 ?> = ai5
Enter new 6 char. name of fl 6 ?> = ai6
Enter new 6 char. name of fl 7 ?> = ai7
Enter new 6 char. name of fl 8 ?> = ai8
Enter new 6 char. name of fl 9 ?> = ai9
Enter new 6 char. name of fl 10 ?> = ai10
Enter new 6 char. name of fl 11 ?> = ai11
Enter new 6 char. name of fl 12 ?> = ai12
Enter new 6 char. name of fl 13 ?> = ai13
Enter new 6 char. name of fl 14 ?> = ai14
Enter new 6 char. name of fl 15 ?> = ai15
Enter new 6 char. name of fl 16 ?> = ai16
Enter new 6 char. name of fl 17 ?> = ai17
Enter new 6 char. name of fl 18 ?> = ai18
Enter new 6 char. name of fl 19 ?> = ai19
Enter new 6 char. name of fl 20 ?> = ai20
Enter new 6 char. name of fl 21 ?> = ai21
Enter new 6 char. name of fl 22 ?> = ai22
Enter new 6 char. name of fl 23 ?> = ai23
Enter new 6 char. name of fl 24 ?> = ai24
Enter new 6 char. name of fl 25 ?> = ai25
Enter new 6 char. name of fl 26 ?> = ai26
Enter new 6 char. name of fl 27 ?> = ai27
Enter new 6 char. name of fl 28 ?> = ai28
Enter new 6 char. name of fl 29 ?> = ai29
Enter new 6 char. name of fl 30 ?> = ai30
Enter new 6 char. name of fl 31 ?> = ai31
Enter new 6 char. name of fl 32 ?> = ai32
Enter new 6 char. name of fl 33 ?> = ai33
Enter new 6 char. name of fl 34 ?> = ai34
Enter new 6 char. name of fl 35 ?> = ai35
MUPF?>  = add 11 pt
Enter new 6 char. name of pt 0 ?> = pepn
Enter new 6 char. name of pt 1 ?> = pdin
Enter new 6 char. name of pt 2 ?> = pdev
Enter new 6 char. name of pt 3 ?> = pai
Enter new 6 char. name of pt 4 ?> = palarm
Enter new 6 char. name of pt 5 ?> = pskid
Enter new 6 char. name of pt 6 ?> = pstat
Enter new 6 char. name of pt 7 ?> = ptype
Enter new 6 char. name of pt 8 ?> = ppass
Enter new 6 char. name of pt 9 ?> = prev
Enter new 6 char. name of pt 10 ?> = pspar0
MUPF?>  = add  15 sg
Enter new 6 char. name of sg 0 ?> = epns0
Enter max string length for sg 0  ?> = 10
Enter new 6 char. name of sg 1 ?> = epns1
Enter max string length for sg 1  ?> = 10
Enter new 6 char. name of sg 2 ?> = epns2
Enter max string length for sg 2  ?> = 10
```

```
Enter new 6 char. name of sg 3 ?> = epns3
Enter max string length for sg 3  ?> = 10
Enter new 6 char. name of sg 4 ?> = epns4
Enter max string length for sg 4  ?> = 10
Enter new 6 char. name of sg 5 ?> = lsg0
Enter max string length for sg 5  ?> = 30
Enter new 6 char. name of sg 6 ?> = lsg1
Enter max string length for sg 6  ?> = 30
Enter new 6 char. name of sg 7 ?> = lsg2
Enter max string length for sg 7  ?> = 30
Enter new 6 char. name of sg 8 ?> = lsg3
Enter max string length for sg 8  ?> = 30
Enter new 6 char. name of sg 9 ?> = lsg4
Enter max string length for sg 9  ?> = 30
Enter new 6 char. name of sg 10 ?> = strng0
Enter max string length for sg 10  ?> = 30
Enter new 6 char. name of sg 11 ?> = strng1
Enter max string length for sg 11  ?> = 30
Enter new 6 char. name of sg 12 ?> = strng2
Enter max string length for sg 12  ?> = 30
Enter new 6 char. name of sg 13 ?> = strng3
Enter max string length for sg 13  ?> = 30
Enter new 6 char. name of sg 14 ?> = strng4
Enter max string length for sg 14  ?> = 30
MUPF?>
 end of file encountered
UPF?>
 end of file encountered
```

OVRVEW

```
UPF Version 6.3-2       05-Dec-94 09:37:11
UPF?>  = de ovrvew
UPF?>  = cr ovrvew,*
Enter 4 character name of RPF to associate?> = 242a
MUPF?>  = add 720 fg
Enter new 6 char. name of fg 0 ?> = fai0
Enter new 6 char. name of fg 1 ?> = fai1
Enter new 6 char. name of fg 2 ?> = fai2
Enter new 6 char. name of fg 3 ?> = fai3
Enter new 6 char. name of fg 4 ?> = fai4
Enter new 6 char. name of fg 5 ?> = fai5
Enter new 6 char. name of fg 6 ?> = fai6
Enter new 6 char. name of fg 7 ?> = fai7
Enter new 6 char. name of fg 8 ?> = fai8
Enter new 6 char. name of fg 9 ?> = fai9
Enter new 6 char. name of fg 10 ?> = fai10
Enter new 6 char. name of fg 11 ?> = fai11
Enter new 6 char. name of fg 12 ?> = fai12
Enter new 6 char. name of fg 13 ?> = fai13
Enter new 6 char. name of fg 14 ?> = fai14
Enter new 6 char. name of fg 15 ?> = fai15
Enter new 6 char. name of fg 16 ?> = fai16
Enter new 6 char. name of fg 17 ?> = fai17
Enter new 6 char. name of fg 18 ?> = fai18
Enter new 6 char. name of fg 19 ?> = fai19
Enter new 6 char. name of fg 20 ?> = fai20
Enter new 6 char. name of fg 21 ?> = fai21
Enter new 6 char. name of fg 22 ?> = fai22
Enter new 6 char. name of fg 23 ?> = fai23
Enter new 6 char. name of fg 24 ?> = fai24
Enter new 6 char. name of fg 25 ?> = fai25
Enter new 6 char. name of fg 26 ?> = fai26
Enter new 6 char. name of fg 27 ?> = fai27
Enter new 6 char. name of fg 28 ?> = fai28
Enter new 6 char. name of fg 29 ?> = fai29
Enter new 6 char. name of fg 30 ?> = fai30
Enter new 6 char. name of fg 31 ?> = fai31
Enter new 6 char. name of fg 32 ?> = fai32
Enter new 6 char. name of fg 33 ?> = fai33
Enter new 6 char. name of fg 34 ?> = fai34
Enter new 6 char. name of fg 35 ?> = fai35
Enter new 6 char. name of fg 36 ?> = fai36
Enter new 6 char. name of fg 37 ?> = fai37
Enter new 6 char. name of fg 38 ?> = fai38
Enter new 6 char. name of fg 39 ?> = fai39
Enter new 6 char. name of fg 40 ?> = fai40
Enter new 6 char. name of fg 41 ?> = fai41
Enter new 6 char. name of fg 42 ?> = fai42
Enter new 6 char. name of fg 43 ?> = fai43
Enter new 6 char. name of fg 44 ?> = fai44
```

```
Enter new 6 char. name of fg 45 ?> = fai45
Enter new 6 char. name of fg 46 ?> = fai46
Enter new 6 char. name of fg 47 ?> = fai47
Enter new 6 char. name of fg 48 ?> = fai48
Enter new 6 char. name of fg 49 ?> = fai49
Enter new 6 char. name of fg 50 ?> = fai50
Enter new 6 char. name of fg 51 ?> = fai51
Enter new 6 char. name of fg 52 ?> = fai52
Enter new 6 char. name of fg 53 ?> = fai53
Enter new 6 char. name of fg 54 ?> = fai54
Enter new 6 char. name of fg 55 ?> = fai55
Enter new 6 char. name of fg 56 ?> = fai56
Enter new 6 char. name of fg 57 ?> = fai57
Enter new 6 char. name of fg 58 ?> = fai58
Enter new 6 char. name of fg 59 ?> = fai59
Enter new 6 char. name of fg 60 ?> = fai60
Enter new 6 char. name of fg 61 ?> = fai61
Enter new 6 char. name of fg 62 ?> = fai62
Enter new 6 char. name of fg 63 ?> = fai63
Enter new 6 char. name of fg 64 ?> = fai64
Enter new 6 char. name of fg 65 ?> = fai65
Enter new 6 char. name of fg 66 ?> = fai66
Enter new 6 char. name of fg 67 ?> = fai67
Enter new 6 char. name of fg 68 ?> = fai68
Enter new 6 char. name of fg 69 ?> = fai69
Enter new 6 char. name of fg 70 ?> = fai70
Enter new 6 char. name of fg 71 ?> = fai71
Enter new 6 char. name of fg 72 ?> = fai72
Enter new 6 char. name of fg 73 ?> = fai73
Enter new 6 char. name of fg 74 ?> = fai74
Enter new 6 char. name of fg 75 ?> = fai75
Enter new 6 char. name of fg 76 ?> = fai76
Enter new 6 char. name of fg 77 ?> = fai77
Enter new 6 char. name of fg 78 ?> = fai78
Enter new 6 char. name of fg 79 ?> = fai79
Enter new 6 char. name of fg 80 ?> = fai80
Enter new 6 char. name of fg 81 ?> = fai81
Enter new 6 char. name of fg 82 ?> = fai82
Enter new 6 char. name of fg 83 ?> = fai83
Enter new 6 char. name of fg 84 ?> = fai84
Enter new 6 char. name of fg 85 ?> = fai85
Enter new 6 char. name of fg 86 ?> = fai86
Enter new 6 char. name of fg 87 ?> = fai87
Enter new 6 char. name of fg 88 ?> = fai88
Enter new 6 char. name of fg 89 ?> = fai89
Enter new 6 char. name of fg 90 ?> = fai90
Enter new 6 char. name of fg 91 ?> = fai91
Enter new 6 char. name of fg 92 ?> = fai92
Enter new 6 char. name of fg 93 ?> = fai93
Enter new 6 char. name of fg 94 ?> = fai94
```

```
Enter new 6 char. name of fg 95 ?> = fai95
Enter new 6 char. name of fg 96 ?> = fai96
Enter new 6 char. name of fg 97 ?> = fai97
Enter new 6 char. name of fg 98 ?> = fai98
Enter new 6 char. name of fg 99 ?> = fai99
Enter new 6 char. name of fg 100 ?> = fai100
Enter new 6 char. name of fg 101 ?> = fai101
Enter new 6 char. name of fg 102 ?> = fai102
Enter new 6 char. name of fg 103 ?> = fai103
Enter new 6 char. name of fg 104 ?> = fai104
Enter new 6 char. name of fg 105 ?> = fai105
Enter new 6 char. name of fg 106 ?> = fai106
Enter new 6 char. name of fg 107 ?> = fai107
Enter new 6 char. name of fg 108 ?> = fai108
Enter new 6 char. name of fg 109 ?> = fai109
Enter new 6 char. name of fg 110 ?> = fai110
Enter new 6 char. name of fg 111 ?> = fai111
Enter new 6 char. name of fg 112 ?> = fai112
Enter new 6 char. name of fg 113 ?> = fai113
Enter new 6 char. name of fg 114 ?> = fai114
Enter new 6 char. name of fg 115 ?> = fai115
Enter new 6 char. name of fg 116 ?> = fai116
Enter new 6 char. name of fg 117 ?> = fai117
Enter new 6 char. name of fg 118 ?> = fai118
Enter new 6 char. name of fg 119 ?> = fai119
Enter new 6 char. name of fg 120 ?> = fai120
Enter new 6 char. name of fg 121 ?> = fai121
Enter new 6 char. name of fg 122 ?> = fai122
Enter new 6 char. name of fg 123 ?> = fai123
Enter new 6 char. name of fg 124 ?> = fai124
Enter new 6 char. name of fg 125 ?> = fai125
Enter new 6 char. name of fg 126 ?> = fai126
Enter new 6 char. name of fg 127 ?> = fai127
Enter new 6 char. name of fg 128 ?> = fai128
Enter new 6 char. name of fg 129 ?> = fai129
Enter new 6 char. name of fg 130 ?> = fai130
Enter new 6 char. name of fg 131 ?> = fai131
Enter new 6 char. name of fg 132 ?> = fai132
Enter new 6 char. name of fg 133 ?> = fai133
Enter new 6 char. name of fg 134 ?> = fai134
Enter new 6 char. name of fg 135 ?> = fai135
Enter new 6 char. name of fg 136 ?> = fai136
Enter new 6 char. name of fg 137 ?> = fai137
Enter new 6 char. name of fg 138 ?> = fai138
Enter new 6 char. name of fg 139 ?> = fai139
Enter new 6 char. name of fg 140 ?> = fai140
Enter new 6 char. name of fg 141 ?> = fai141
Enter new 6 char. name of fg 142 ?> = fai142
Enter new 6 char. name of fg 143 ?> = fai143
Enter new 6 char. name of fg 144 ?> = fai144
```

```
Enter new 6 char. name of fg 145 ?> = fail45
Enter new 6 char. name of fg 146 ?> = fail46
Enter new 6 char. name of fg 147 ?> = fail47
Enter new 6 char. name of fg 148 ?> = fail48
Enter new 6 char. name of fg 149 ?> = fail49
Enter new 6 char. name of fg 150 ?> = fail50
Enter new 6 char. name of fg 151 ?> = fail51
Enter new 6 char. name of fg 152 ?> = fail52
Enter new 6 char. name of fg 153 ?> = fail53
Enter new 6 char. name of fg 154 ?> = fail54
Enter new 6 char. name of fg 155 ?> = fail55
Enter new 6 char. name of fg 156 ?> = fail56
Enter new 6 char. name of fg 157 ?> = fail57
Enter new 6 char. name of fg 158 ?> = fail58
Enter new 6 char. name of fg 159 ?> = fail59
Enter new 6 char. name of fg 160 ?> = fail60
Enter new 6 char. name of fg 161 ?> = fail61
Enter new 6 char. name of fg 162 ?> = fail62
Enter new 6 char. name of fg 163 ?> = fail63
Enter new 6 char. name of fg 164 ?> = fail64
Enter new 6 char. name of fg 165 ?> = fail65
Enter new 6 char. name of fg 166 ?> = fail66
Enter new 6 char. name of fg 167 ?> = fail67
Enter new 6 char. name of fg 168 ?> = fail68
Enter new 6 char. name of fg 169 ?> = fail69
Enter new 6 char. name of fg 170 ?> = fail70
Enter new 6 char. name of fg 171 ?> = fail71
Enter new 6 char. name of fg 172 ?> = fail72
Enter new 6 char. name of fg 173 ?> = fail73
Enter new 6 char. name of fg 174 ?> = fail74
Enter new 6 char. name of fg 175 ?> = fail75
Enter new 6 char. name of fg 176 ?> = fal0
Enter new 6 char. name of fg 177 ?> = fal1
Enter new 6 char. name of fg 178 ?> = fal2
Enter new 6 char. name of fg 179 ?> = fal3
Enter new 6 char. name of fg 180 ?> = fal4
Enter new 6 char. name of fg 181 ?> = fal5
Enter new 6 char. name of fg 182 ?> = fal6
Enter new 6 char. name of fg 183 ?> = fal7
Enter new 6 char. name of fg 184 ?> = fal8
Enter new 6 char. name of fg 185 ?> = fal9
Enter new 6 char. name of fg 186 ?> = fal10
Enter new 6 char. name of fg 187 ?> = fal11
Enter new 6 char. name of fg 188 ?> = fal12
Enter new 6 char. name of fg 189 ?> = fal13
Enter new 6 char. name of fg 190 ?> = fal14
Enter new 6 char. name of fg 191 ?> = fal15
Enter new 6 char. name of fg 192 ?> = fal16
Enter new 6 char. name of fg 193 ?> = fal17
Enter new 6 char. name of fg 194 ?> = fal18
```

```
Enter new 6 char. name of fg 195 ?> = fal19
Enter new 6 char. name of fg 196 ?> = fal20
Enter new 6 char. name of fg 197 ?> = fal21
Enter new 6 char. name of fg 198 ?> = fal22
Enter new 6 char. name of fg 199 ?> = fal23
Enter new 6 char. name of fg 200 ?> = fal24
Enter new 6 char. name of fg 201 ?> = fal25
Enter new 6 char. name of fg 202 ?> = fal26
Enter new 6 char. name of fg 203 ?> = fal27
Enter new 6 char. name of fg 204 ?> = fal28
Enter new 6 char. name of fg 205 ?> = fal29
Enter new 6 char. name of fg 206 ?> = fal30
Enter new 6 char. name of fg 207 ?> = fal31
Enter new 6 char. name of fg 208 ?> = fal32
Enter new 6 char. name of fg 209 ?> = fal33
Enter new 6 char. name of fg 210 ?> = fal34
Enter new 6 char. name of fg 211 ?> = fal35
Enter new 6 char. name of fg 212 ?> = fal36
Enter new 6 char. name of fg 213 ?> = fal37
Enter new 6 char. name of fg 214 ?> = fal38
Enter new 6 char. name of fg 215 ?> = fal39
Enter new 6 char. name of fg 216 ?> = fal40
Enter new 6 char. name of fg 217 ?> = fal41
Enter new 6 char. name of fg 218 ?> = fal42
Enter new 6 char. name of fg 219 ?> = fal43
Enter new 6 char. name of fg 220 ?> = fal44
Enter new 6 char. name of fg 221 ?> = fal45
Enter new 6 char. name of fg 222 ?> = fal46
Enter new 6 char. name of fg 223 ?> = fal47
Enter new 6 char. name of fg 224 ?> = fal48
Enter new 6 char. name of fg 225 ?> = fal49
Enter new 6 char. name of fg 226 ?> = fal50
Enter new 6 char. name of fg 227 ?> = fal51
Enter new 6 char. name of fg 228 ?> = fal52
Enter new 6 char. name of fg 229 ?> = fal53
Enter new 6 char. name of fg 230 ?> = fal54
Enter new 6 char. name of fg 231 ?> = fal55
Enter new 6 char. name of fg 232 ?> = fal56
Enter new 6 char. name of fg 233 ?> = fal57
Enter new 6 char. name of fg 234 ?> = fal58
Enter new 6 char. name of fg 235 ?> = fal59
Enter new 6 char. name of fg 236 ?> = fal60
Enter new 6 char. name of fg 237 ?> = fal61
Enter new 6 char. name of fg 238 ?> = fal62
Enter new 6 char. name of fg 239 ?> = fal63
Enter new 6 char. name of fg 240 ?> = fal64
Enter new 6 char. name of fg 241 ?> = fal65
Enter new 6 char. name of fg 242 ?> = fal66
Enter new 6 char. name of fg 243 ?> = fal67
Enter new 6 char. name of fg 244 ?> = fal68
```

```
Enter new 6 char. name of fg 245 ?> = fal69
Enter new 6 char. name of fg 246 ?> = fal70
Enter new 6 char. name of fg 247 ?> = fal71
Enter new 6 char. name of fg 248 ?> = fal72
Enter new 6 char. name of fg 249 ?> = fal73
Enter new 6 char. name of fg 250 ?> = fal74
Enter new 6 char. name of fg 251 ?> = fal75
Enter new 6 char. name of fg 252 ?> = fal76
Enter new 6 char. name of fg 253 ?> = fal77
Enter new 6 char. name of fg 254 ?> = fal78
Enter new 6 char. name of fg 255 ?> = fal79
Enter new 6 char. name of fg 256 ?> = fal80
Enter new 6 char. name of fg 257 ?> = fal81
Enter new 6 char. name of fg 258 ?> = fal82
Enter new 6 char. name of fg 259 ?> = fal83
Enter new 6 char. name of fg 260 ?> = fal84
Enter new 6 char. name of fg 261 ?> = fal85
Enter new 6 char. name of fg 262 ?> = fal86
Enter new 6 char. name of fg 263 ?> = fal87
Enter new 6 char. name of fg 264 ?> = fal88
Enter new 6 char. name of fg 265 ?> = fal89
Enter new 6 char. name of fg 266 ?> = fal90
Enter new 6 char. name of fg 267 ?> = fal91
Enter new 6 char. name of fg 268 ?> = fal92
Enter new 6 char. name of fg 269 ?> = fal93
Enter new 6 char. name of fg 270 ?> = fal94
Enter new 6 char. name of fg 271 ?> = fal95
Enter new 6 char. name of fg 272 ?> = fal96
Enter new 6 char. name of fg 273 ?> = fal97
Enter new 6 char. name of fg 274 ?> = fal98
Enter new 6 char. name of fg 275 ?> = fal99
Enter new 6 char. name of fg 276 ?> = fal100
Enter new 6 char. name of fg 277 ?> = fal101
Enter new 6 char. name of fg 278 ?> = fal102
Enter new 6 char. name of fg 279 ?> = fal103
Enter new 6 char. name of fg 280 ?> = fal104
Enter new 6 char. name of fg 281 ?> = fal105
Enter new 6 char. name of fg 282 ?> = fal106
Enter new 6 char. name of fg 283 ?> = fal107
Enter new 6 char. name of fg 284 ?> = fal108
Enter new 6 char. name of fg 285 ?> = fal109
Enter new 6 char. name of fg 286 ?> = fal110
Enter new 6 char. name of fg 287 ?> = fal111
Enter new 6 char. name of fg 288 ?> = fal112
Enter new 6 char. name of fg 289 ?> = fal113
Enter new 6 char. name of fg 290 ?> = fal114
Enter new 6 char. name of fg 291 ?> = fal115
Enter new 6 char. name of fg 292 ?> = fal116
Enter new 6 char. name of fg 293 ?> = fal117
Enter new 6 char. name of fg 294 ?> = fal118
```

```
Enter new 6 char. name of fg 295 ?> = fal119
Enter new 6 char. name of fg 296 ?> = fal120
Enter new 6 char. name of fg 297 ?> = fal121
Enter new 6 char. name of fg 298 ?> = fal122
Enter new 6 char. name of fg 299 ?> = fal123
Enter new 6 char. name of fg 300 ?> = fal124
Enter new 6 char. name of fg 301 ?> = fal125
Enter new 6 char. name of fg 302 ?> = fal126
Enter new 6 char. name of fg 303 ?> = fal127
Enter new 6 char. name of fg 304 ?> = fal128
Enter new 6 char. name of fg 305 ?> = fal129
Enter new 6 char. name of fg 306 ?> = fal130
Enter new 6 char. name of fg 307 ?> = fal131
Enter new 6 char. name of fg 308 ?> = fal132
Enter new 6 char. name of fg 309 ?> = fal133
Enter new 6 char. name of fg 310 ?> = fal134
Enter new 6 char. name of fg 311 ?> = fal135
Enter new 6 char. name of fg 312 ?> = fal136
Enter new 6 char. name of fg 313 ?> = fal137
Enter new 6 char. name of fg 314 ?> = fal138
Enter new 6 char. name of fg 315 ?> = fal139
Enter new 6 char. name of fg 316 ?> = fal140
Enter new 6 char. name of fg 317 ?> = fal141
Enter new 6 char. name of fg 318 ?> = fal142
Enter new 6 char. name of fg 319 ?> = fal143
Enter new 6 char. name of fg 320 ?> = fal144
Enter new 6 char. name of fg 321 ?> = fal145
Enter new 6 char. name of fg 322 ?> = fal146
Enter new 6 char. name of fg 323 ?> = fal147
Enter new 6 char. name of fg 324 ?> = fal148
Enter new 6 char. name of fg 325 ?> = fal149
Enter new 6 char. name of fg 326 ?> = fal150
Enter new 6 char. name of fg 327 ?> = fal151
Enter new 6 char. name of fg 328 ?> = fal152
Enter new 6 char. name of fg 329 ?> = fal153
Enter new 6 char. name of fg 330 ?> = fal154
Enter new 6 char. name of fg 331 ?> = fal155
Enter new 6 char. name of fg 332 ?> = fal156
Enter new 6 char. name of fg 333 ?> = fal157
Enter new 6 char. name of fg 334 ?> = fal158
Enter new 6 char. name of fg 335 ?> = fal159
Enter new 6 char. name of fg 336 ?> = fal160
Enter new 6 char. name of fg 337 ?> = fal161
Enter new 6 char. name of fg 338 ?> = fal162
Enter new 6 char. name of fg 339 ?> = fal163
Enter new 6 char. name of fg 340 ?> = fal164
Enter new 6 char. name of fg 341 ?> = fal165
Enter new 6 char. name of fg 342 ?> = fal166
Enter new 6 char. name of fg 343 ?> = fal167
Enter new 6 char. name of fg 344 ?> = fal168
```

```
Enter new 6 char. name of fg 345 ?> = fal169
Enter new 6 char. name of fg 346 ?> = fal170
Enter new 6 char. name of fg 347 ?> = fal171
Enter new 6 char. name of fg 348 ?> = fal172
Enter new 6 char. name of fg 349 ?> = fal173
Enter new 6 char. name of fg 350 ?> = fal174
Enter new 6 char. name of fg 351 ?> = fal175
Enter new 6 char. name of fg 352 ?> = frv0
Enter new 6 char. name of fg 353 ?> = frv1
Enter new 6 char. name of fg 354 ?> = frv2
Enter new 6 char. name of fg 355 ?> = frv3
Enter new 6 char. name of fg 356 ?> = frv4
Enter new 6 char. name of fg 357 ?> = frv5
Enter new 6 char. name of fg 358 ?> = frv6
Enter new 6 char. name of fg 359 ?> = frv7
Enter new 6 char. name of fg 360 ?> = frv8
Enter new 6 char. name of fg 361 ?> = frv9
Enter new 6 char. name of fg 362 ?> = frv10
Enter new 6 char. name of fg 363 ?> = frv11
Enter new 6 char. name of fg 364 ?> = frv12
Enter new 6 char. name of fg 365 ?> = frv13
Enter new 6 char. name of fg 366 ?> = frv14
Enter new 6 char. name of fg 367 ?> = frv15
Enter new 6 char. name of fg 368 ?> = frv16
Enter new 6 char. name of fg 369 ?> = frv17
Enter new 6 char. name of fg 370 ?> = frv18
Enter new 6 char. name of fg 371 ?> = frv19
Enter new 6 char. name of fg 372 ?> = frv20
Enter new 6 char. name of fg 373 ?> = frv21
Enter new 6 char. name of fg 374 ?> = frv22
Enter new 6 char. name of fg 375 ?> = frv23
Enter new 6 char. name of fg 376 ?> = frv24
Enter new 6 char. name of fg 377 ?> = frv25
Enter new 6 char. name of fg 378 ?> = frv26
Enter new 6 char. name of fg 379 ?> = frv27
Enter new 6 char. name of fg 380 ?> = frv28
Enter new 6 char. name of fg 381 ?> = frv29
Enter new 6 char. name of fg 382 ?> = frv30
Enter new 6 char. name of fg 383 ?> = frv31
Enter new 6 char. name of fg 384 ?> = frv32
Enter new 6 char. name of fg 385 ?> = frv33
Enter new 6 char. name of fg 386 ?> = frv34
Enter new 6 char. name of fg 387 ?> = frv35
Enter new 6 char. name of fg 388 ?> = frv36
Enter new 6 char. name of fg 389 ?> = frv37
Enter new 6 char. name of fg 390 ?> = frv38
Enter new 6 char. name of fg 391 ?> = frv39
Enter new 6 char. name of fg 392 ?> = frv40
Enter new 6 char. name of fg 393 ?> = frv41
Enter new 6 char. name of fg 394 ?> = frv42
```

```
Enter new 6 char. name of fg 395 ?> = frv43
Enter new 6 char. name of fg 396 ?> = frv44
Enter new 6 char. name of fg 397 ?> = frv45
Enter new 6 char. name of fg 398 ?> = frv46
Enter new 6 char. name of fg 399 ?> = frv47
Enter new 6 char. name of fg 400 ?> = frv48
Enter new 6 char. name of fg 401 ?> = frv49
Enter new 6 char. name of fg 402 ?> = frv50
Enter new 6 char. name of fg 403 ?> = frv51
Enter new 6 char. name of fg 404 ?> = frv52
Enter new 6 char. name of fg 405 ?> = frv53
Enter new 6 char. name of fg 406 ?> = frv54
Enter new 6 char. name of fg 407 ?> = frv55
Enter new 6 char. name of fg 408 ?> = frv56
Enter new 6 char. name of fg 409 ?> = frv57
Enter new 6 char. name of fg 410 ?> = frv58
Enter new 6 char. name of fg 411 ?> = frv59
Enter new 6 char. name of fg 412 ?> = frv60
Enter new 6 char. name of fg 413 ?> = frv61
Enter new 6 char. name of fg 414 ?> = frv62
Enter new 6 char. name of fg 415 ?> = frv63
Enter new 6 char. name of fg 416 ?> = frv64
Enter new 6 char. name of fg 417 ?> = frv65
Enter new 6 char. name of fg 418 ?> = frv66
Enter new 6 char. name of fg 419 ?> = frv67
Enter new 6 char. name of fg 420 ?> = frv68
Enter new 6 char. name of fg 421 ?> = frv69
Enter new 6 char. name of fg 422 ?> = frv70
Enter new 6 char. name of fg 423 ?> = frv71
Enter new 6 char. name of fg 424 ?> = frv72
Enter new 6 char. name of fg 425 ?> = frv73
Enter new 6 char. name of fg 426 ?> = frv74
Enter new 6 char. name of fg 427 ?> = frv75
Enter new 6 char. name of fg 428 ?> = frv76
Enter new 6 char. name of fg 429 ?> = frv77
Enter new 6 char. name of fg 430 ?> = frv78
Enter new 6 char. name of fg 431 ?> = frv79
Enter new 6 char. name of fg 432 ?> = frv80
Enter new 6 char. name of fg 433 ?> = frv81
Enter new 6 char. name of fg 434 ?> = frv82
Enter new 6 char. name of fg 435 ?> = frv83
Enter new 6 char. name of fg 436 ?> = frv84
Enter new 6 char. name of fg 437 ?> = frv85
Enter new 6 char. name of fg 438 ?> = frv86
Enter new 6 char. name of fg 439 ?> = frv87
Enter new 6 char. name of fg 440 ?> = frv88
Enter new 6 char. name of fg 441 ?> = frv89
Enter new 6 char. name of fg 442 ?> = frv90
Enter new 6 char. name of fg 443 ?> = frv91
Enter new 6 char. name of fg 444 ?> = frv92
```

```
Enter new 6 char. name of fg 445 ?> = frv93
Enter new 6 char. name of fg 446 ?> = frv94
Enter new 6 char. name of fg 447 ?> = frv95
Enter new 6 char. name of fg 448 ?> = frv96
Enter new 6 char. name of fg 449 ?> = frv97
Enter new 6 char. name of fg 450 ?> = frv98
Enter new 6 char. name of fg 451 ?> = frv99
Enter new 6 char. name of fg 452 ?> = frv100
Enter new 6 char. name of fg 453 ?> = frv101
Enter new 6 char. name of fg 454 ?> = frv102
Enter new 6 char. name of fg 455 ?> = frv103
Enter new 6 char. name of fg 456 ?> = frv104
Enter new 6 char. name of fg 457 ?> = frv105
Enter new 6 char. name of fg 458 ?> = frv106
Enter new 6 char. name of fg 459 ?> = frv107
Enter new 6 char. name of fg 460 ?> = frv108
Enter new 6 char. name of fg 461 ?> = frv109
Enter new 6 char. name of fg 462 ?> = frv110
Enter new 6 char. name of fg 463 ?> = frv111
Enter new 6 char. name of fg 464 ?> = frv112
Enter new 6 char. name of fg 465 ?> = frv113
Enter new 6 char. name of fg 466 ?> = frv114
Enter new 6 char. name of fg 467 ?> = frv115
Enter new 6 char. name of fg 468 ?> = frv116
Enter new 6 char. name of fg 469 ?> = frv117
Enter new 6 char. name of fg 470 ?> = frv118
Enter new 6 char. name of fg 471 ?> = frv119
Enter new 6 char. name of fg 472 ?> = frv120
Enter new 6 char. name of fg 473 ?> = frv121
Enter new 6 char. name of fg 474 ?> = frv122
Enter new 6 char. name of fg 475 ?> = frv123
Enter new 6 char. name of fg 476 ?> = frv124
Enter new 6 char. name of fg 477 ?> = frv125
Enter new 6 char. name of fg 478 ?> = frv126
Enter new 6 char. name of fg 479 ?> = frv127
Enter new 6 char. name of fg 480 ?> = frv128
Enter new 6 char. name of fg 481 ?> = frv129
Enter new 6 char. name of fg 482 ?> = frv130
Enter new 6 char. name of fg 483 ?> = frv131
Enter new 6 char. name of fg 484 ?> = frv132
Enter new 6 char. name of fg 485 ?> = frv133
Enter new 6 char. name of fg 486 ?> = frv134
Enter new 6 char. name of fg 487 ?> = frv135
Enter new 6 char. name of fg 488 ?> = frv136
Enter new 6 char. name of fg 489 ?> = frv137
Enter new 6 char. name of fg 490 ?> = frv138
Enter new 6 char. name of fg 491 ?> = frv139
Enter new 6 char. name of fg 492 ?> = frv140
Enter new 6 char. name of fg 493 ?> = frv141
Enter new 6 char. name of fg 494 ?> = frv142
```

```
Enter new 6 char. name of fg 495 ?> = frv143
Enter new 6 char. name of fg 496 ?> = frv144
Enter new 6 char. name of fg 497 ?> = frv145
Enter new 6 char. name of fg 498 ?> = frv146
Enter new 6 char. name of fg 499 ?> = frv147
Enter new 6 char. name of fg 500 ?> = frv148
Enter new 6 char. name of fg 501 ?> = frv149
Enter new 6 char. name of fg 502 ?> = frv150
Enter new 6 char. name of fg 503 ?> = frv151
Enter new 6 char. name of fg 504 ?> = frv152
Enter new 6 char. name of fg 505 ?> = frv153
Enter new 6 char. name of fg 506 ?> = frv154
Enter new 6 char. name of fg 507 ?> = frv155
Enter new 6 char. name of fg 508 ?> = frv156
Enter new 6 char. name of fg 509 ?> = frv157
Enter new 6 char. name of fg 510 ?> = frv158
Enter new 6 char. name of fg 511 ?> = frv159
Enter new 6 char. name of fg 512 ?> = frv160
Enter new 6 char. name of fg 513 ?> = frv161
Enter new 6 char. name of fg 514 ?> = frv162
Enter new 6 char. name of fg 515 ?> = frv163
Enter new 6 char. name of fg 516 ?> = frv164
Enter new 6 char. name of fg 517 ?> = frv165
Enter new 6 char. name of fg 518 ?> = frv166
Enter new 6 char. name of fg 519 ?> = frv167
Enter new 6 char. name of fg 520 ?> = frv168
Enter new 6 char. name of fg 521 ?> = frv169
Enter new 6 char. name of fg 522 ?> = frv170
Enter new 6 char. name of fg 523 ?> = frv171
Enter new 6 char. name of fg 524 ?> = frv172
Enter new 6 char. name of fg 525 ?> = frv173
Enter new 6 char. name of fg 526 ?> = frv174
Enter new 6 char. name of fg 527 ?> = frv175
Enter new 6 char. name of fg 528 ?> = fds0
Enter new 6 char. name of fg 529 ?> = fds1
Enter new 6 char. name of fg 530 ?> = fds2
Enter new 6 char. name of fg 531 ?> = fds3
Enter new 6 char. name of fg 532 ?> = fds4
Enter new 6 char. name of fg 533 ?> = fds5
Enter new 6 char. name of fg 534 ?> = fds6
Enter new 6 char. name of fg 535 ?> = fds7
Enter new 6 char. name of fg 536 ?> = fds8
Enter new 6 char. name of fg 537 ?> = fds9
Enter new 6 char. name of fg 538 ?> = fds10
Enter new 6 char. name of fg 539 ?> = fds11
Enter new 6 char. name of fg 540 ?> = fds12
Enter new 6 char. name of fg 541 ?> = fds13
Enter new 6 char. name of fg 542 ?> = fds14
Enter new 6 char. name of fg 543 ?> = fds15
Enter new 6 char. name of fg 544 ?> = fds16
```

```
Enter new 6 char. name of fg 545 ?> = fds17
Enter new 6 char. name of fg 546 ?> = fds18
Enter new 6 char. name of fg 547 ?> = fds19
Enter new 6 char. name of fg 548 ?> = fds20
Enter new 6 char. name of fg 549 ?> = fds21
Enter new 6 char. name of fg 550 ?> = fds22
Enter new 6 char. name of fg 551 ?> = fds23
Enter new 6 char. name of fg 552 ?> = fds24
Enter new 6 char. name of fg 553 ?> = fds25
Enter new 6 char. name of fg 554 ?> = fds26
Enter new 6 char. name of fg 555 ?> = fds27
Enter new 6 char. name of fg 556 ?> = fds28
Enter new 6 char. name of fg 557 ?> = fds29
Enter new 6 char. name of fg 558 ?> = fds30
Enter new 6 char. name of fg 559 ?> = fds31
Enter new 6 char. name of fg 560 ?> = fds32
Enter new 6 char. name of fg 561 ?> = fds33
Enter new 6 char. name of fg 562 ?> = fds34
Enter new 6 char. name of fg 563 ?> = fds35
Enter new 6 char. name of fg 564 ?> = fds36
Enter new 6 char. name of fg 565 ?> = fds37
Enter new 6 char. name of fg 566 ?> = fds38
Enter new 6 char. name of fg 567 ?> = fds39
Enter new 6 char. name of fg 568 ?> = fds40
Enter new 6 char. name of fg 569 ?> = fds41
Enter new 6 char. name of fg 570 ?> = fds42
Enter new 6 char. name of fg 571 ?> = fds43
Enter new 6 char. name of fg 572 ?> = fds44
Enter new 6 char. name of fg 573 ?> = fds45
Enter new 6 char. name of fg 574 ?> = fds46
Enter new 6 char. name of fg 575 ?> = fds47
Enter new 6 char. name of fg 576 ?> = fds48
Enter new 6 char. name of fg 577 ?> = fds49
Enter new 6 char. name of fg 578 ?> = fds50
Enter new 6 char. name of fg 579 ?> = fds51
Enter new 6 char. name of fg 580 ?> = fds52
Enter new 6 char. name of fg 581 ?> = fds53
Enter new 6 char. name of fg 582 ?> = fds54
Enter new 6 char. name of fg 583 ?> = fds55
Enter new 6 char. name of fg 584 ?> = fds56
Enter new 6 char. name of fg 585 ?> = fds57
Enter new 6 char. name of fg 586 ?> = fds58
Enter new 6 char. name of fg 587 ?> = fds59
Enter new 6 char. name of fg 588 ?> = fds60
Enter new 6 char. name of fg 589 ?> = fds61
Enter new 6 char. name of fg 590 ?> = fds62
Enter new 6 char. name of fg 591 ?> = fds63
Enter new 6 char. name of fg 592 ?> = fds64
Enter new 6 char. name of fg 593 ?> = fds65
Enter new 6 char. name of fg 594 ?> = fds66
```

```
Enter new 6 char. name of fg 595 ?> = fds67
Enter new 6 char. name of fg 596 ?> = fds68
Enter new 6 char. name of fg 597 ?> = fds69
Enter new 6 char. name of fg 598 ?> = fds70
Enter new 6 char. name of fg 599 ?> = fds71
Enter new 6 char. name of fg 600 ?> = fds72
Enter new 6 char. name of fg 601 ?> = fds73
Enter new 6 char. name of fg 602 ?> = fds74
Enter new 6 char. name of fg 603 ?> = fds75
Enter new 6 char. name of fg 604 ?> = fds76
Enter new 6 char. name of fg 605 ?> = fds77
Enter new 6 char. name of fg 606 ?> = fds78
Enter new 6 char. name of fg 607 ?> = fds79
Enter new 6 char. name of fg 608 ?> = fds80
Enter new 6 char. name of fg 609 ?> = fds81
Enter new 6 char. name of fg 610 ?> = fds82
Enter new 6 char. name of fg 611 ?> = fds83
Enter new 6 char. name of fg 612 ?> = fds84
Enter new 6 char. name of fg 613 ?> = fds85
Enter new 6 char. name of fg 614 ?> = fds86
Enter new 6 char. name of fg 615 ?> = fds87
Enter new 6 char. name of fg 616 ?> = fds88
Enter new 6 char. name of fg 617 ?> = fds89
Enter new 6 char. name of fg 618 ?> = fds90
Enter new 6 char. name of fg 619 ?> = fds91
Enter new 6 char. name of fg 620 ?> = fds92
Enter new 6 char. name of fg 621 ?> = fds93
Enter new 6 char. name of fg 622 ?> = fds94
Enter new 6 char. name of fg 623 ?> = fds95
Enter new 6 char. name of fg 624 ?> = fds96
Enter new 6 char. name of fg 625 ?> = fds97
Enter new 6 char. name of fg 626 ?> = fds98
Enter new 6 char. name of fg 627 ?> = fds99
Enter new 6 char. name of fg 628 ?> = fds100
Enter new 6 char. name of fg 629 ?> = fds101
Enter new 6 char. name of fg 630 ?> = fds102
Enter new 6 char. name of fg 631 ?> = fds103
Enter new 6 char. name of fg 632 ?> = fds104
Enter new 6 char. name of fg 633 ?> = fds105
Enter new 6 char. name of fg 634 ?> = fds106
Enter new 6 char. name of fg 635 ?> = fds107
Enter new 6 char. name of fg 636 ?> = fds108
Enter new 6 char. name of fg 637 ?> = fds109
Enter new 6 char. name of fg 638 ?> = fds110
Enter new 6 char. name of fg 639 ?> = fds111
Enter new 6 char. name of fg 640 ?> = fds112
Enter new 6 char. name of fg 641 ?> = fds113
Enter new 6 char. name of fg 642 ?> = fds114
Enter new 6 char. name of fg 643 ?> = fds115
Enter new 6 char. name of fg 644 ?> = fds116
```

```
Enter new 6 char. name of fg 645 ?> = fds117
Enter new 6 char. name of fg 646 ?> = fds118
Enter new 6 char. name of fg 647 ?> = fds119
Enter new 6 char. name of fg 648 ?> = fds120
Enter new 6 char. name of fg 649 ?> = fds121
Enter new 6 char. name of fg 650 ?> = fds122
Enter new 6 char. name of fg 651 ?> = fds123
Enter new 6 char. name of fg 652 ?> = fds124
Enter new 6 char. name of fg 653 ?> = fds125
Enter new 6 char. name of fg 654 ?> = fds126
Enter new 6 char. name of fg 655 ?> = fds127
Enter new 6 char. name of fg 656 ?> = fds128
Enter new 6 char. name of fg 657 ?> = fds129
Enter new 6 char. name of fg 658 ?> = fds130
Enter new 6 char. name of fg 659 ?> = fds131
Enter new 6 char. name of fg 660 ?> = fds132
Enter new 6 char. name of fg 661 ?> = fds133
Enter new 6 char. name of fg 662 ?> = fds134
Enter new 6 char. name of fg 663 ?> = fds135
Enter new 6 char. name of fg 664 ?> = fds136
Enter new 6 char. name of fg 665 ?> = fds137
Enter new 6 char. name of fg 666 ?> = fds138
Enter new 6 char. name of fg 667 ?> = fds139
Enter new 6 char. name of fg 668 ?> = fds140
Enter new 6 char. name of fg 669 ?> = fds141
Enter new 6 char. name of fg 670 ?> = fds142
Enter new 6 char. name of fg 671 ?> = fds143
Enter new 6 char. name of fg 672 ?> = fds144
Enter new 6 char. name of fg 673 ?> = fds145
Enter new 6 char. name of fg 674 ?> = fds146
Enter new 6 char. name of fg 675 ?> = fds147
Enter new 6 char. name of fg 676 ?> = fds148
Enter new 6 char. name of fg 677 ?> = fds149
Enter new 6 char. name of fg 678 ?> = fds150
Enter new 6 char. name of fg 679 ?> = fds151
Enter new 6 char. name of fg 680 ?> = fds152
Enter new 6 char. name of fg 681 ?> = fds153
Enter new 6 char. name of fg 682 ?> = fds154
Enter new 6 char. name of fg 683 ?> = fds155
Enter new 6 char. name of fg 684 ?> = fds156
Enter new 6 char. name of fg 685 ?> = fds157
Enter new 6 char. name of fg 686 ?> = fds158
Enter new 6 char. name of fg 687 ?> = fds159
Enter new 6 char. name of fg 688 ?> = fds160
Enter new 6 char. name of fg 689 ?> = fds161
Enter new 6 char. name of fg 690 ?> = fds162
Enter new 6 char. name of fg 691 ?> = fds163
Enter new 6 char. name of fg 692 ?> = fds164
Enter new 6 char. name of fg 693 ?> = fds165
Enter new 6 char. name of fg 694 ?> = fds166
```

```
Enter new 6 char. name of fg 695 ?> = fds167
Enter new 6 char. name of fg 696 ?> = fds168
Enter new 6 char. name of fg 697 ?> = fds169
Enter new 6 char. name of fg 698 ?> = fds170
Enter new 6 char. name of fg 699 ?> = fds171
Enter new 6 char. name of fg 700 ?> = fds172
Enter new 6 char. name of fg 701 ?> = fds173
Enter new 6 char. name of fg 702 ?> = fds174
Enter new 6 char. name of fg 703 ?> = fds175
Enter new 6 char. name of fg 704 ?> = fdisp
Enter new 6 char. name of fg 705 ?> = fhold
Enter new 6 char. name of fg 706 ?> = fning
Enter new 6 char. name of fg 707 ?> = fabort
Enter new 6 char. name of fg 708 ?> = fstart
Enter new 6 char. name of fg 709 ?> = ftrip0
Enter new 6 char. name of fg 710 ?> = ftrip1
Enter new 6 char. name of fg 711 ?> = ftrip2
Enter new 6 char. name of fg 712 ?> = fspar0
Enter new 6 char. name of fg 713 ?> = fspar1
Enter new 6 char. name of fg 714 ?> = fspar2
Enter new 6 char. name of fg 715 ?> = fspar3
Enter new 6 char. name of fg 716 ?> = fspar4
Enter new 6 char. name of fg 717 ?> = fspar5
Enter new 6 char. name of fg 718 ?> = fspar6
Enter new 6 char. name of fg 719 ?> = fspar7
MUPF?>  = add 367 in
Enter new 6 char. name of in 0 ?> = bitval
Enter new 6 char. name of in 1 ?> = iloop
Enter new 6 char. name of in 2 ?> = jloop
Enter new 6 char. name of in 3 ?> = kloop
Enter new 6 char. name of in 4 ?> = lloop
Enter new 6 char. name of in 5 ?> = index
Enter new 6 char. name of in 6 ?> = trip
Enter new 6 char. name of in 7 ?> = nadi
Enter new 6 char. name of in 8 ?> = blkval
Enter new 6 char. name of in 9 ?> = sort
Enter new 6 char. name of in 10 ?> = bval
Enter new 6 char. name of in 11 ?> = hldstp
Enter new 6 char. name of in 12 ?> = remain
Enter new 6 char. name of in 13 ?> = errn
Enter new 6 char. name of in 14 ?> = errl
Enter new 6 char. name of in 15 ?> = int0
Enter new 6 char. name of in 16 ?> = int1
Enter new 6 char. name of in 17 ?> = int2
Enter new 6 char. name of in 18 ?> = int3
Enter new 6 char. name of in 19 ?> = int4
Enter new 6 char. name of in 20 ?> = int5
Enter new 6 char. name of in 21 ?> = int6
Enter new 6 char. name of in 22 ?> = int7
Enter new 6 char. name of in 23 ?> = int8
```

```
Enter new 6 char. name of in 24 ?> = int9
Enter new 6 char. name of in 25 ?> = int10
Enter new 6 char. name of in 26 ?> = int11
Enter new 6 char. name of in 27 ?> = int12
Enter new 6 char. name of in 28 ?> = int13
Enter new 6 char. name of in 29 ?> = int14
Enter new 6 char. name of in 30 ?> = int15
Enter new 6 char. name of in 31 ?> = int16
Enter new 6 char. name of in 32 ?> = int17
Enter new 6 char. name of in 33 ?> = int18
Enter new 6 char. name of in 34 ?> = int19
Enter new 6 char. name of in 35 ?> = int20
Enter new 6 char. name of in 36 ?> = int21
Enter new 6 char. name of in 37 ?> = int22
Enter new 6 char. name of in 38 ?> = int23
Enter new 6 char. name of in 39 ?> = int24
Enter new 6 char. name of in 40 ?> = int25
Enter new 6 char. name of in 41 ?> = int26
Enter new 6 char. name of in 42 ?> = int27
Enter new 6 char. name of in 43 ?> = int28
Enter new 6 char. name of in 44 ?> = int29
Enter new 6 char. name of in 45 ?> = int30
Enter new 6 char. name of in 46 ?> = int31
Enter new 6 char. name of in 47 ?> = int32
Enter new 6 char. name of in 48 ?> = int33
Enter new 6 char. name of in 49 ?> = int34
Enter new 6 char. name of in 50 ?> = int35
Enter new 6 char. name of in 51 ?> = int36
Enter new 6 char. name of in 52 ?> = int37
Enter new 6 char. name of in 53 ?> = int38
Enter new 6 char. name of in 54 ?> = int39
Enter new 6 char. name of in 55 ?> = int40
Enter new 6 char. name of in 56 ?> = int41
Enter new 6 char. name of in 57 ?> = int42
Enter new 6 char. name of in 58 ?> = int43
Enter new 6 char. name of in 59 ?> = int44
Enter new 6 char. name of in 60 ?> = int45
Enter new 6 char. name of in 61 ?> = int46
Enter new 6 char. name of in 62 ?> = int47
Enter new 6 char. name of in 63 ?> = int48
Enter new 6 char. name of in 64 ?> = int49
Enter new 6 char. name of in 65 ?> = int50
Enter new 6 char. name of in 66 ?> = int51
Enter new 6 char. name of in 67 ?> = int52
Enter new 6 char. name of in 68 ?> = int53
Enter new 6 char. name of in 69 ?> = int54
Enter new 6 char. name of in 70 ?> = int55
Enter new 6 char. name of in 71 ?> = int56
Enter new 6 char. name of in 72 ?> = int57
Enter new 6 char. name of in 73 ?> = int58
```

```
Enter new 6 char. name of in 74 ?> = int59
Enter new 6 char. name of in 75 ?> = int60
Enter new 6 char. name of in 76 ?> = int61
Enter new 6 char. name of in 77 ?> = int62
Enter new 6 char. name of in 78 ?> = int63
Enter new 6 char. name of in 79 ?> = int64
Enter new 6 char. name of in 80 ?> = int65
Enter new 6 char. name of in 81 ?> = int66
Enter new 6 char. name of in 82 ?> = int67
Enter new 6 char. name of in 83 ?> = int68
Enter new 6 char. name of in 84 ?> = int69
Enter new 6 char. name of in 85 ?> = int70
Enter new 6 char. name of in 86 ?> = int71
Enter new 6 char. name of in 87 ?> = int72
Enter new 6 char. name of in 88 ?> = int73
Enter new 6 char. name of in 89 ?> = int74
Enter new 6 char. name of in 90 ?> = int75
Enter new 6 char. name of in 91 ?> = int76
Enter new 6 char. name of in 92 ?> = int77
Enter new 6 char. name of in 93 ?> = int78
Enter new 6 char. name of in 94 ?> = int79
Enter new 6 char. name of in 95 ?> = int80
Enter new 6 char. name of in 96 ?> = int81
Enter new 6 char. name of in 97 ?> = int82
Enter new 6 char. name of in 98 ?> = int83
Enter new 6 char. name of in 99 ?> = int84
Enter new 6 char. name of in 100 ?> = int85
Enter new 6 char. name of in 101 ?> = int86
Enter new 6 char. name of in 102 ?> = int87
Enter new 6 char. name of in 103 ?> = int88
Enter new 6 char. name of in 104 ?> = int89
Enter new 6 char. name of in 105 ?> = int90
Enter new 6 char. name of in 106 ?> = int91
Enter new 6 char. name of in 107 ?> = int92
Enter new 6 char. name of in 108 ?> = int93
Enter new 6 char. name of in 109 ?> = int94
Enter new 6 char. name of in 110 ?> = int95
Enter new 6 char. name of in 111 ?> = int96
Enter new 6 char. name of in 112 ?> = int97
Enter new 6 char. name of in 113 ?> = int98
Enter new 6 char. name of in 114 ?> = int99
Enter new 6 char. name of in 115 ?> = sta100
Enter new 6 char. name of in 116 ?> = sta101
Enter new 6 char. name of in 117 ?> = sta102
Enter new 6 char. name of in 118 ?> = sta103
Enter new 6 char. name of in 119 ?> = sta104
Enter new 6 char. name of in 120 ?> = sta105
Enter new 6 char. name of in 121 ?> = sta106
Enter new 6 char. name of in 122 ?> = sta107
Enter new 6 char. name of in 123 ?> = sta108
```

```
Enter new 6 char. name of in 124 ?> = sta109
Enter new 6 char. name of in 125 ?> = sta110
Enter new 6 char. name of in 126 ?> = sta111
Enter new 6 char. name of in 127 ?> = sta112
Enter new 6 char. name of in 128 ?> = sta113
Enter new 6 char. name of in 129 ?> = sta114
Enter new 6 char. name of in 130 ?> = sta115
Enter new 6 char. name of in 131 ?> = sta116
Enter new 6 char. name of in 132 ?> = sta117
Enter new 6 char. name of in 133 ?> = sta118
Enter new 6 char. name of in 134 ?> = sta119
Enter new 6 char. name of in 135 ?> = sta120
Enter new 6 char. name of in 136 ?> = sta121
Enter new 6 char. name of in 137 ?> = sta122
Enter new 6 char. name of in 138 ?> = sta123
Enter new 6 char. name of in 139 ?> = sta124
Enter new 6 char. name of in 140 ?> = sta125
Enter new 6 char. name of in 141 ?> = sta126
Enter new 6 char. name of in 142 ?> = sta127
Enter new 6 char. name of in 143 ?> = sta128
Enter new 6 char. name of in 144 ?> = sta129
Enter new 6 char. name of in 145 ?> = sta130
Enter new 6 char. name of in 146 ?> = sta131
Enter new 6 char. name of in 147 ?> = sta132
Enter new 6 char. name of in 148 ?> = sta133
Enter new 6 char. name of in 149 ?> = sta134
Enter new 6 char. name of in 150 ?> = sta135
Enter new 6 char. name of in 151 ?> = sta136
Enter new 6 char. name of in 152 ?> = sta137
Enter new 6 char. name of in 153 ?> = sta138
Enter new 6 char. name of in 154 ?> = sta139
Enter new 6 char. name of in 155 ?> = sta140
Enter new 6 char. name of in 156 ?> = sta141
Enter new 6 char. name of in 157 ?> = sta142
Enter new 6 char. name of in 158 ?> = sta143
Enter new 6 char. name of in 159 ?> = sta144
Enter new 6 char. name of in 160 ?> = sta145
Enter new 6 char. name of in 161 ?> = sta146
Enter new 6 char. name of in 162 ?> = sta147
Enter new 6 char. name of in 163 ?> = sta148
Enter new 6 char. name of in 164 ?> = sta149
Enter new 6 char. name of in 165 ?> = sta150
Enter new 6 char. name of in 166 ?> = sta151
Enter new 6 char. name of in 167 ?> = sta152
Enter new 6 char. name of in 168 ?> = sta153
Enter new 6 char. name of in 169 ?> = sta154
Enter new 6 char. name of in 170 ?> = sta155
Enter new 6 char. name of in 171 ?> = sta156
Enter new 6 char. name of in 172 ?> = sta157
Enter new 6 char. name of in 173 ?> = sta158
```

```
Enter new 6 char. name of in 174 ?> = sta159
Enter new 6 char. name of in 175 ?> = sta160
Enter new 6 char. name of in 176 ?> = sta161
Enter new 6 char. name of in 177 ?> = sta162
Enter new 6 char. name of in 178 ?> = sta163
Enter new 6 char. name of in 179 ?> = sta164
Enter new 6 char. name of in 180 ?> = sta165
Enter new 6 char. name of in 181 ?> = sta166
Enter new 6 char. name of in 182 ?> = sta167
Enter new 6 char. name of in 183 ?> = sta168
Enter new 6 char. name of in 184 ?> = sta169
Enter new 6 char. name of in 185 ?> = sta170
Enter new 6 char. name of in 186 ?> = sta171
Enter new 6 char. name of in 187 ?> = sta172
Enter new 6 char. name of in 188 ?> = sta173
Enter new 6 char. name of in 189 ?> = sta174
Enter new 6 char. name of in 190 ?> = sta175
Enter new 6 char. name of in 191 ?> = type0
Enter new 6 char. name of in 192 ?> = type1
Enter new 6 char. name of in 193 ?> = type2
Enter new 6 char. name of in 194 ?> = type3
Enter new 6 char. name of in 195 ?> = type4
Enter new 6 char. name of in 196 ?> = type5
Enter new 6 char. name of in 197 ?> = type6
Enter new 6 char. name of in 198 ?> = type7
Enter new 6 char. name of in 199 ?> = type8
Enter new 6 char. name of in 200 ?> = type9
Enter new 6 char. name of in 201 ?> = type10
Enter new 6 char. name of in 202 ?> = type11
Enter new 6 char. name of in 203 ?> = type12
Enter new 6 char. name of in 204 ?> = type13
Enter new 6 char. name of in 205 ?> = type14
Enter new 6 char. name of in 206 ?> = type15
Enter new 6 char. name of in 207 ?> = type16
Enter new 6 char. name of in 208 ?> = type17
Enter new 6 char. name of in 209 ?> = type18
Enter new 6 char. name of in 210 ?> = type19
Enter new 6 char. name of in 211 ?> = type20
Enter new 6 char. name of in 212 ?> = type21
Enter new 6 char. name of in 213 ?> = type22
Enter new 6 char. name of in 214 ?> = type23
Enter new 6 char. name of in 215 ?> = type24
Enter new 6 char. name of in 216 ?> = type25
Enter new 6 char. name of in 217 ?> = type26
Enter new 6 char. name of in 218 ?> = type27
Enter new 6 char. name of in 219 ?> = type28
Enter new 6 char. name of in 220 ?> = type29
Enter new 6 char. name of in 221 ?> = type30
Enter new 6 char. name of in 222 ?> = type31
Enter new 6 char. name of in 223 ?> = type32
```

```
Enter new 6 char. name of in 224 ?> = type33
Enter new 6 char. name of in 225 ?> = type34
Enter new 6 char. name of in 226 ?> = type35
Enter new 6 char. name of in 227 ?> = type36
Enter new 6 char. name of in 228 ?> = type37
Enter new 6 char. name of in 229 ?> = type38
Enter new 6 char. name of in 230 ?> = type39
Enter new 6 char. name of in 231 ?> = type40
Enter new 6 char. name of in 232 ?> = type41
Enter new 6 char. name of in 233 ?> = type42
Enter new 6 char. name of in 234 ?> = type43
Enter new 6 char. name of in 235 ?> = type44
Enter new 6 char. name of in 236 ?> = type45
Enter new 6 char. name of in 237 ?> = type46
Enter new 6 char. name of in 238 ?> = type47
Enter new 6 char. name of in 239 ?> = type48
Enter new 6 char. name of in 240 ?> = type49
Enter new 6 char. name of in 241 ?> = type50
Enter new 6 char. name of in 242 ?> = type51
Enter new 6 char. name of in 243 ?> = type52
Enter new 6 char. name of in 244 ?> = type53
Enter new 6 char. name of in 245 ?> = type54
Enter new 6 char. name of in 246 ?> = type55
Enter new 6 char. name of in 247 ?> = type56
Enter new 6 char. name of in 248 ?> = type57
Enter new 6 char. name of in 249 ?> = type58
Enter new 6 char. name of in 250 ?> = type59
Enter new 6 char. name of in 251 ?> = type60
Enter new 6 char. name of in 252 ?> = type61
Enter new 6 char. name of in 253 ?> = type62
Enter new 6 char. name of in 254 ?> = type63
Enter new 6 char. name of in 255 ?> = type64
Enter new 6 char. name of in 256 ?> = type65
Enter new 6 char. name of in 257 ?> = type66
Enter new 6 char. name of in 258 ?> = type67
Enter new 6 char. name of in 259 ?> = type68
Enter new 6 char. name of in 260 ?> = type69
Enter new 6 char. name of in 261 ?> = type70
Enter new 6 char. name of in 262 ?> = type71
Enter new 6 char. name of in 263 ?> = type72
Enter new 6 char. name of in 264 ?> = type73
Enter new 6 char. name of in 265 ?> = type74
Enter new 6 char. name of in 266 ?> = type75
Enter new 6 char. name of in 267 ?> = type76
Enter new 6 char. name of in 268 ?> = type77
Enter new 6 char. name of in 269 ?> = type78
Enter new 6 char. name of in 270 ?> = type79
Enter new 6 char. name of in 271 ?> = type80
Enter new 6 char. name of in 272 ?> = type81
Enter new 6 char. name of in 273 ?> = type82
```

```
Enter new 6 char. name of in 274 ?> = type83
Enter new 6 char. name of in 275 ?> = type84
Enter new 6 char. name of in 276 ?> = type85
Enter new 6 char. name of in 277 ?> = type86
Enter new 6 char. name of in 278 ?> = type87
Enter new 6 char. name of in 279 ?> = type88
Enter new 6 char. name of in 280 ?> = type89
Enter new 6 char. name of in 281 ?> = type90
Enter new 6 char. name of in 282 ?> = type91
Enter new 6 char. name of in 283 ?> = type92
Enter new 6 char. name of in 284 ?> = type93
Enter new 6 char. name of in 285 ?> = type94
Enter new 6 char. name of in 286 ?> = type95
Enter new 6 char. name of in 287 ?> = type96
Enter new 6 char. name of in 288 ?> = type97
Enter new 6 char. name of in 289 ?> = type98
Enter new 6 char. name of in 290 ?> = type99
Enter new 6 char. name of in 291 ?> = typ100
Enter new 6 char. name of in 292 ?> = typ101
Enter new 6 char. name of in 293 ?> = typ102
Enter new 6 char. name of in 294 ?> = typ103
Enter new 6 char. name of in 295 ?> = typ104
Enter new 6 char. name of in 296 ?> = typ105
Enter new 6 char. name of in 297 ?> = typ106
Enter new 6 char. name of in 298 ?> = typ107
Enter new 6 char. name of in 299 ?> = typ108
Enter new 6 char. name of in 300 ?> = typ109
Enter new 6 char. name of in 301 ?> = typ110
Enter new 6 char. name of in 302 ?> = typ111
Enter new 6 char. name of in 303 ?> = typ112
Enter new 6 char. name of in 304 ?> = typ113
Enter new 6 char. name of in 305 ?> = typ114
Enter new 6 char. name of in 306 ?> = typ115
Enter new 6 char. name of in 307 ?> = typ116
Enter new 6 char. name of in 308 ?> = typ117
Enter new 6 char. name of in 309 ?> = typ118
Enter new 6 char. name of in 310 ?> = typ119
Enter new 6 char. name of in 311 ?> = typ120
Enter new 6 char. name of in 312 ?> = typ121
Enter new 6 char. name of in 313 ?> = typ122
Enter new 6 char. name of in 314 ?> = typ123
Enter new 6 char. name of in 315 ?> = typ124
Enter new 6 char. name of in 316 ?> = typ125
Enter new 6 char. name of in 317 ?> = typ126
Enter new 6 char. name of in 318 ?> = typ127
Enter new 6 char. name of in 319 ?> = typ128
Enter new 6 char. name of in 320 ?> = typ129
Enter new 6 char. name of in 321 ?> = typ130
Enter new 6 char. name of in 322 ?> = typ131
Enter new 6 char. name of in 323 ?> = typ132
```

```
Enter new 6 char. name of in 324 ?> = typ133
Enter new 6 char. name of in 325 ?> = typ134
Enter new 6 char. name of in 326 ?> = typ135
Enter new 6 char. name of in 327 ?> = typ136
Enter new 6 char. name of in 328 ?> = typ137
Enter new 6 char. name of in 329 ?> = typ138
Enter new 6 char. name of in 330 ?> = typ139
Enter new 6 char. name of in 331 ?> = typ140
Enter new 6 char. name of in 332 ?> = typ141
Enter new 6 char. name of in 333 ?> = typ142
Enter new 6 char. name of in 334 ?> = typ143
Enter new 6 char. name of in 335 ?> = typ144
Enter new 6 char. name of in 336 ?> = typ145
Enter new 6 char. name of in 337 ?> = typ146
Enter new 6 char. name of in 338 ?> = typ147
Enter new 6 char. name of in 339 ?> = typ148
Enter new 6 char. name of in 340 ?> = typ149
Enter new 6 char. name of in 341 ?> = typ150
Enter new 6 char. name of in 342 ?> = typ151
Enter new 6 char. name of in 343 ?> = typ152
Enter new 6 char. name of in 344 ?> = typ153
Enter new 6 char. name of in 345 ?> = typ154
Enter new 6 char. name of in 346 ?> = typ155
Enter new 6 char. name of in 347 ?> = typ156
Enter new 6 char. name of in 348 ?> = typ157
Enter new 6 char. name of in 349 ?> = typ158
Enter new 6 char. name of in 350 ?> = typ159
Enter new 6 char. name of in 351 ?> = typ160
Enter new 6 char. name of in 352 ?> = typ161
Enter new 6 char. name of in 353 ?> = typ162
Enter new 6 char. name of in 354 ?> = typ163
Enter new 6 char. name of in 355 ?> = typ164
Enter new 6 char. name of in 356 ?> = typ165
Enter new 6 char. name of in 357 ?> = typ166
Enter new 6 char. name of in 358 ?> = typ167
Enter new 6 char. name of in 359 ?> = typ168
Enter new 6 char. name of in 360 ?> = typ169
Enter new 6 char. name of in 361 ?> = typ170
Enter new 6 char. name of in 362 ?> = typ171
Enter new 6 char. name of in 363 ?> = typ172
Enter new 6 char. name of in 364 ?> = typ173
Enter new 6 char. name of in 365 ?> = typ174
Enter new 6 char. name of in 366 ?> = typ175
MUPF?>  = add 50 fl
Enter new 6 char. name of fl 0 ?> = trip0
Enter new 6 char. name of fl 1 ?> = trip1
Enter new 6 char. name of fl 2 ?> = trip2
Enter new 6 char. name of fl 3 ?> = trip3
Enter new 6 char. name of fl 4 ?> = trip4
Enter new 6 char. name of fl 5 ?> = trip5
```

```
Enter new 6 char. name of fl 6 ?> = trip6
Enter new 6 char. name of fl 7 ?> = trip7
Enter new 6 char. name of fl 8 ?> = trip8
Enter new 6 char. name of fl 9 ?> = trip9
Enter new 6 char. name of fl 10 ?> = ai0
Enter new 6 char. name of fl 11 ?> = ai1
Enter new 6 char. name of fl 12 ?> = ai2
Enter new 6 char. name of fl 13 ?> = ai3
Enter new 6 char. name of fl 14 ?> = ai4
Enter new 6 char. name of fl 15 ?> = ai5
Enter new 6 char. name of fl 16 ?> = ai6
Enter new 6 char. name of fl 17 ?> = ai7
Enter new 6 char. name of fl 18 ?> = ai8
Enter new 6 char. name of fl 19 ?> = ai9
Enter new 6 char. name of fl 20 ?> = ai10
Enter new 6 char. name of fl 21 ?> = ai11
Enter new 6 char. name of fl 22 ?> = ai12
Enter new 6 char. name of fl 23 ?> = ai13
Enter new 6 char. name of fl 24 ?> = ai14
Enter new 6 char. name of fl 25 ?> = ai15
Enter new 6 char. name of fl 26 ?> = ai16
Enter new 6 char. name of fl 27 ?> = ai17
Enter new 6 char. name of fl 28 ?> = ai18
Enter new 6 char. name of fl 29 ?> = ai19
Enter new 6 char. name of fl 30 ?> = ai20
Enter new 6 char. name of fl 31 ?> = ai21
Enter new 6 char. name of fl 32 ?> = ai22
Enter new 6 char. name of fl 33 ?> = ai23
Enter new 6 char. name of fl 34 ?> = ai24
Enter new 6 char. name of fl 35 ?> = ai25
Enter new 6 char. name of fl 36 ?> = ai26
Enter new 6 char. name of fl 37 ?> = ai27
Enter new 6 char. name of fl 38 ?> = ai28
Enter new 6 char. name of fl 39 ?> = ai29
Enter new 6 char. name of fl 40 ?> = ai30
Enter new 6 char. name of fl 41 ?> = ai31
Enter new 6 char. name of fl 42 ?> = ai32
Enter new 6 char. name of fl 43 ?> = ai33
Enter new 6 char. name of fl 44 ?> = ai34
Enter new 6 char. name of fl 45 ?> = ai35
Enter new 6 char. name of fl 46 ?> = ai36
Enter new 6 char. name of fl 47 ?> = ai37
Enter new 6 char. name of fl 48 ?> = ai38
Enter new 6 char. name of fl 49 ?> = ai39
MUPF?>  = add 12 pt
Enter new 6 char. name of pt 0 ?> = pepn
Enter new 6 char. name of pt 1 ?> = pdev
Enter new 6 char. name of pt 2 ?> = pdin
Enter new 6 char. name of pt 3 ?> = pai
Enter new 6 char. name of pt 4 ?> = ptrip
```

```
Enter new 6 char. name of pt 5 ?> = pack
Enter new 6 char. name of pt 6 ?> = pdsabl
Enter new 6 char. name of pt 7 ?> = preps
Enter new 6 char. name of pt 8 ?> = pskid
Enter new 6 char. name of pt 9 ?> = pstat
Enter new 6 char. name of pt 10 ?> = ptype
Enter new 6 char. name of pt 11 ?> = prev
MUPF?>  = add 25 sg
Enter new 6 char. name of sg 0 ?> = date0
Enter max string length for sg 0  ?> = 10
Enter new 6 char. name of sg 1 ?> = date1
Enter max string length for sg 1  ?> = 10
Enter new 6 char. name of sg 2 ?> = date2
Enter max string length for sg 2  ?> = 10
Enter new 6 char. name of sg 3 ?> = date3
Enter max string length for sg 3  ?> = 10
Enter new 6 char. name of sg 4 ?> = date4
Enter max string length for sg 4  ?> = 10
Enter new 6 char. name of sg 5 ?> = time0
Enter max string length for sg 5  ?> = 8
Enter new 6 char. name of sg 6 ?> = time1
Enter max string length for sg 6  ?> = 8
Enter new 6 char. name of sg 7 ?> = time2
Enter max string length for sg 7  ?> = 8
Enter new 6 char. name of sg 8 ?> = time3
Enter max string length for sg 8  ?> = 8
Enter new 6 char. name of sg 9 ?> = time4
Enter max string length for sg 9  ?> = 8
Enter new 6 char. name of sg 10 ?> = epn0
Enter max string length for sg 10  ?> = 10
Enter new 6 char. name of sg 11 ?> = epn1
Enter max string length for sg 11  ?> = 10
Enter new 6 char. name of sg 12 ?> = epn2
Enter max string length for sg 12  ?> = 10
Enter new 6 char. name of sg 13 ?> = epn3
Enter max string length for sg 13  ?> = 10
Enter new 6 char. name of sg 14 ?> = epn4
Enter max string length for sg 14  ?> = 10
Enter new 6 char. name of sg 15 ?> = mssg0
Enter max string length for sg 15  ?> = 30
Enter new 6 char. name of sg 16 ?> = mssg1
Enter max string length for sg 16  ?> = 30
Enter new 6 char. name of sg 17 ?> = mssg2
Enter max string length for sg 17  ?> = 30
Enter new 6 char. name of sg 18 ?> = mssg3
Enter max string length for sg 18  ?> = 30
Enter new 6 char. name of sg 19 ?> = mssg4
Enter max string length for sg 19  ?> = 30
Enter new 6 char. name of sg 20 ?> = lsg0
Enter max string length for sg 20  ?> = 30
```

```
Enter new 6 char. name of sg 21 ?> = lsg1
Enter max string length for sg 21  ?> = 30
Enter new 6 char. name of sg 22 ?> = lsg2
Enter max string length for sg 22  ?> = 30
Enter new 6 char. name of sg 23 ?> = lsg3
Enter max string length for sg 23  ?> = 30
Enter new 6 char. name of sg 24 ?> = lsg4
Enter max string length for sg 24  ?> = 30
MUPF?>
 end of file encountered
UPF?>
 end of file encountered
```

TOTAL1

```
UPF Version 6.3-2       05-Dec-94 09:39:28
UPF?>  = de total1
UPF?>  = cr total1,*
Enter 4 character name of RPF to associate?> = all1
MUPF?>  = add 784 fg
Enter new 6 char. name of fg 0 ?> = fds0
Enter new 6 char. name of fg 1 ?> = fds1
Enter new 6 char. name of fg 2 ?> = fds2
Enter new 6 char. name of fg 3 ?> = fds3
Enter new 6 char. name of fg 4 ?> = fds4
Enter new 6 char. name of fg 5 ?> = fds5
Enter new 6 char. name of fg 6 ?> = fds6
Enter new 6 char. name of fg 7 ?> = fds7
Enter new 6 char. name of fg 8 ?> = fds8
Enter new 6 char. name of fg 9 ?> = fds9
Enter new 6 char. name of fg 10 ?> = fds10
Enter new 6 char. name of fg 11 ?> = fds11
Enter new 6 char. name of fg 12 ?> = fds12
Enter new 6 char. name of fg 13 ?> = fds13
Enter new 6 char. name of fg 14 ?> = fds14
Enter new 6 char. name of fg 15 ?> = fds15
Enter new 6 char. name of fg 16 ?> = fds16
Enter new 6 char. name of fg 17 ?> = fds17
Enter new 6 char. name of fg 18 ?> = fds18
Enter new 6 char. name of fg 19 ?> = fds19
Enter new 6 char. name of fg 20 ?> = fds20
Enter new 6 char. name of fg 21 ?> = fds21
Enter new 6 char. name of fg 22 ?> = fds22
Enter new 6 char. name of fg 23 ?> = fds23
Enter new 6 char. name of fg 24 ?> = fds24
Enter new 6 char. name of fg 25 ?> = fds25
Enter new 6 char. name of fg 26 ?> = fds26
Enter new 6 char. name of fg 27 ?> = fds27
Enter new 6 char. name of fg 28 ?> = fds28
Enter new 6 char. name of fg 29 ?> = fds29
Enter new 6 char. name of fg 30 ?> = fds30
Enter new 6 char. name of fg 31 ?> = fds31
Enter new 6 char. name of fg 32 ?> = fds32
Enter new 6 char. name of fg 33 ?> = fds33
Enter new 6 char. name of fg 34 ?> = fds34
Enter new 6 char. name of fg 35 ?> = fds35
Enter new 6 char. name of fg 36 ?> = fds36
Enter new 6 char. name of fg 37 ?> = fds37
Enter new 6 char. name of fg 38 ?> = fds38
Enter new 6 char. name of fg 39 ?> = fds39
Enter new 6 char. name of fg 40 ?> = fds40
Enter new 6 char. name of fg 41 ?> = fds41
Enter new 6 char. name of fg 42 ?> = fds42
Enter new 6 char. name of fg 43 ?> = fds43
Enter new 6 char. name of fg 44 ?> = fds44
```

```
Enter new 6 char. name of fg 45 ?> = fds45
Enter new 6 char. name of fg 46 ?> = fds46
Enter new 6 char. name of fg 47 ?> = fds47
Enter new 6 char. name of fg 48 ?> = fds48
Enter new 6 char. name of fg 49 ?> = fds49
Enter new 6 char. name of fg 50 ?> = fds50
Enter new 6 char. name of fg 51 ?> = fds51
Enter new 6 char. name of fg 52 ?> = fds52
Enter new 6 char. name of fg 53 ?> = fds53
Enter new 6 char. name of fg 54 ?> = fds54
Enter new 6 char. name of fg 55 ?> = fds55
Enter new 6 char. name of fg 56 ?> = fds56
Enter new 6 char. name of fg 57 ?> = fds57
Enter new 6 char. name of fg 58 ?> = fds58
Enter new 6 char. name of fg 59 ?> = fds59
Enter new 6 char. name of fg 60 ?> = fds60
Enter new 6 char. name of fg 61 ?> = fds61
Enter new 6 char. name of fg 62 ?> = fds62
Enter new 6 char. name of fg 63 ?> = fds63
Enter new 6 char. name of fg 64 ?> = fds64
Enter new 6 char. name of fg 65 ?> = fds65
Enter new 6 char. name of fg 66 ?> = fds66
Enter new 6 char. name of fg 67 ?> = fds67
Enter new 6 char. name of fg 68 ?> = fds68
Enter new 6 char. name of fg 69 ?> = fds69
Enter new 6 char. name of fg 70 ?> = fds70
Enter new 6 char. name of fg 71 ?> = fds71
Enter new 6 char. name of fg 72 ?> = fds72
Enter new 6 char. name of fg 73 ?> = fds73
Enter new 6 char. name of fg 74 ?> = fds74
Enter new 6 char. name of fg 75 ?> = fds75
Enter new 6 char. name of fg 76 ?> = fds76
Enter new 6 char. name of fg 77 ?> = fds77
Enter new 6 char. name of fg 78 ?> = fds78
Enter new 6 char. name of fg 79 ?> = fds79
Enter new 6 char. name of fg 80 ?> = fds80
Enter new 6 char. name of fg 81 ?> = fds81
Enter new 6 char. name of fg 82 ?> = fds82
Enter new 6 char. name of fg 83 ?> = fds83
Enter new 6 char. name of fg 84 ?> = fds84
Enter new 6 char. name of fg 85 ?> = fds85
Enter new 6 char. name of fg 86 ?> = fds86
Enter new 6 char. name of fg 87 ?> = fds87
Enter new 6 char. name of fg 88 ?> = fds88
Enter new 6 char. name of fg 89 ?> = fds89
Enter new 6 char. name of fg 90 ?> = fds90
Enter new 6 char. name of fg 91 ?> = fds91
Enter new 6 char. name of fg 92 ?> = fds92
Enter new 6 char. name of fg 93 ?> = fds93
Enter new 6 char. name of fg 94 ?> = fds94
```

```
Enter new 6 char. name of fg 95 ?> = fds95
Enter new 6 char. name of fg 96 ?> = fds96
Enter new 6 char. name of fg 97 ?> = fds97
Enter new 6 char. name of fg 98 ?> = fds98
Enter new 6 char. name of fg 99 ?> = fds99
Enter new 6 char. name of fg 100 ?> = fds100
Enter new 6 char. name of fg 101 ?> = fds101
Enter new 6 char. name of fg 102 ?> = fds102
Enter new 6 char. name of fg 103 ?> = fds103
Enter new 6 char. name of fg 104 ?> = fds104
Enter new 6 char. name of fg 105 ?> = fds105
Enter new 6 char. name of fg 106 ?> = fds106
Enter new 6 char. name of fg 107 ?> = fds107
Enter new 6 char. name of fg 108 ?> = fds108
Enter new 6 char. name of fg 109 ?> = fds109
Enter new 6 char. name of fg 110 ?> = fds110
Enter new 6 char. name of fg 111 ?> = fds111
Enter new 6 char. name of fg 112 ?> = fds112
Enter new 6 char. name of fg 113 ?> = fds113
Enter new 6 char. name of fg 114 ?> = fds114
Enter new 6 char. name of fg 115 ?> = fds115
Enter new 6 char. name of fg 116 ?> = fds116
Enter new 6 char. name of fg 117 ?> = fds117
Enter new 6 char. name of fg 118 ?> = fds118
Enter new 6 char. name of fg 119 ?> = fds119
Enter new 6 char. name of fg 120 ?> = fds120
Enter new 6 char. name of fg 121 ?> = fds121
Enter new 6 char. name of fg 122 ?> = fds122
Enter new 6 char. name of fg 123 ?> = fds123
Enter new 6 char. name of fg 124 ?> = fds124
Enter new 6 char. name of fg 125 ?> = fds125
Enter new 6 char. name of fg 126 ?> = fds126
Enter new 6 char. name of fg 127 ?> = fds127
Enter new 6 char. name of fg 128 ?> = fds128
Enter new 6 char. name of fg 129 ?> = fds129
Enter new 6 char. name of fg 130 ?> = fds130
Enter new 6 char. name of fg 131 ?> = fds131
Enter new 6 char. name of fg 132 ?> = fds132
Enter new 6 char. name of fg 133 ?> = fds133
Enter new 6 char. name of fg 134 ?> = fds134
Enter new 6 char. name of fg 135 ?> = fds135
Enter new 6 char. name of fg 136 ?> = fds136
Enter new 6 char. name of fg 137 ?> = fds137
Enter new 6 char. name of fg 138 ?> = fds138
Enter new 6 char. name of fg 139 ?> = fds139
Enter new 6 char. name of fg 140 ?> = fds140
Enter new 6 char. name of fg 141 ?> = fds141
Enter new 6 char. name of fg 142 ?> = fds142
Enter new 6 char. name of fg 143 ?> = fds143
Enter new 6 char. name of fg 144 ?> = fds144
```

```
Enter new 6 char. name of fg 145 ?> = fds145
Enter new 6 char. name of fg 146 ?> = fds146
Enter new 6 char. name of fg 147 ?> = fds147
Enter new 6 char. name of fg 148 ?> = fds148
Enter new 6 char. name of fg 149 ?> = fds149
Enter new 6 char. name of fg 150 ?> = fds150
Enter new 6 char. name of fg 151 ?> = fds151
Enter new 6 char. name of fg 152 ?> = fds152
Enter new 6 char. name of fg 153 ?> = fds153
Enter new 6 char. name of fg 154 ?> = fds154
Enter new 6 char. name of fg 155 ?> = fds155
Enter new 6 char. name of fg 156 ?> = fds156
Enter new 6 char. name of fg 157 ?> = fds157
Enter new 6 char. name of fg 158 ?> = fds158
Enter new 6 char. name of fg 159 ?> = fds159
Enter new 6 char. name of fg 160 ?> = fds160
Enter new 6 char. name of fg 161 ?> = fds161
Enter new 6 char. name of fg 162 ?> = fds162
Enter new 6 char. name of fg 163 ?> = fds163
Enter new 6 char. name of fg 164 ?> = fds164
Enter new 6 char. name of fg 165 ?> = fds165
Enter new 6 char. name of fg 166 ?> = fds166
Enter new 6 char. name of fg 167 ?> = fds167
Enter new 6 char. name of fg 168 ?> = fds168
Enter new 6 char. name of fg 169 ?> = fds169
Enter new 6 char. name of fg 170 ?> = fds170
Enter new 6 char. name of fg 171 ?> = fds171
Enter new 6 char. name of fg 172 ?> = fds172
Enter new 6 char. name of fg 173 ?> = fds173
Enter new 6 char. name of fg 174 ?> = fds174
Enter new 6 char. name of fg 175 ?> = fds175
Enter new 6 char. name of fg 176 ?> = fds176
Enter new 6 char. name of fg 177 ?> = fds177
Enter new 6 char. name of fg 178 ?> = fds178
Enter new 6 char. name of fg 179 ?> = fds179
Enter new 6 char. name of fg 180 ?> = fds180
Enter new 6 char. name of fg 181 ?> = fds181
Enter new 6 char. name of fg 182 ?> = fds182
Enter new 6 char. name of fg 183 ?> = fds183
Enter new 6 char. name of fg 184 ?> = fds184
Enter new 6 char. name of fg 185 ?> = fds185
Enter new 6 char. name of fg 186 ?> = fds186
Enter new 6 char. name of fg 187 ?> = fds187
Enter new 6 char. name of fg 188 ?> = fds188
Enter new 6 char. name of fg 189 ?> = fds189
Enter new 6 char. name of fg 190 ?> = fds190
Enter new 6 char. name of fg 191 ?> = fds191
Enter new 6 char. name of fg 192 ?> = fx0
Enter new 6 char. name of fg 193 ?> = fx1
Enter new 6 char. name of fg 194 ?> = fx2
```

```
Enter new 6 char. name of fg 195 ?> = fx3
Enter new 6 char. name of fg 196 ?> = fx4
Enter new 6 char. name of fg 197 ?> = fx5
Enter new 6 char. name of fg 198 ?> = fx6
Enter new 6 char. name of fg 199 ?> = fx7
Enter new 6 char. name of fg 200 ?> = fx8
Enter new 6 char. name of fg 201 ?> = fx9
Enter new 6 char. name of fg 202 ?> = fx10
Enter new 6 char. name of fg 203 ?> = fx11
Enter new 6 char. name of fg 204 ?> = fx12
Enter new 6 char. name of fg 205 ?> = fx13
Enter new 6 char. name of fg 206 ?> = fx14
Enter new 6 char. name of fg 207 ?> = fx15
Enter new 6 char. name of fg 208 ?> = fx16
Enter new 6 char. name of fg 209 ?> = fx17
Enter new 6 char. name of fg 210 ?> = fx18
Enter new 6 char. name of fg 211 ?> = fx19
Enter new 6 char. name of fg 212 ?> = fx20
Enter new 6 char. name of fg 213 ?> = fx21
Enter new 6 char. name of fg 214 ?> = fx22
Enter new 6 char. name of fg 215 ?> = fx23
Enter new 6 char. name of fg 216 ?> = fx24
Enter new 6 char. name of fg 217 ?> = fx25
Enter new 6 char. name of fg 218 ?> = fx26
Enter new 6 char. name of fg 219 ?> = fx27
Enter new 6 char. name of fg 220 ?> = fx28
Enter new 6 char. name of fg 221 ?> = fx29
Enter new 6 char. name of fg 222 ?> = fx30
Enter new 6 char. name of fg 223 ?> = fx31
Enter new 6 char. name of fg 224 ?> = fx32
Enter new 6 char. name of fg 225 ?> = fx33
Enter new 6 char. name of fg 226 ?> = fx34
Enter new 6 char. name of fg 227 ?> = fx35
Enter new 6 char. name of fg 228 ?> = fx36
Enter new 6 char. name of fg 229 ?> = fx37
Enter new 6 char. name of fg 230 ?> = fx38
Enter new 6 char. name of fg 231 ?> = fx39
Enter new 6 char. name of fg 232 ?> = fx40
Enter new 6 char. name of fg 233 ?> = fx41
Enter new 6 char. name of fg 234 ?> = fx42
Enter new 6 char. name of fg 235 ?> = fx43
Enter new 6 char. name of fg 236 ?> = fx44
Enter new 6 char. name of fg 237 ?> = fx45
Enter new 6 char. name of fg 238 ?> = fx46
Enter new 6 char. name of fg 239 ?> = fx47
Enter new 6 char. name of fg 240 ?> = fx48
Enter new 6 char. name of fg 241 ?> = fx49
Enter new 6 char. name of fg 242 ?> = fx50
Enter new 6 char. name of fg 243 ?> = fx51
Enter new 6 char. name of fg 244 ?> = fx52
```

```
Enter new 6 char. name of fg 245 ?> = fx53
Enter new 6 char. name of fg 246 ?> = fx54
Enter new 6 char. name of fg 247 ?> = fx55
Enter new 6 char. name of fg 248 ?> = fx56
Enter new 6 char. name of fg 249 ?> = fx57
Enter new 6 char. name of fg 250 ?> = fx58
Enter new 6 char. name of fg 251 ?> = fx59
Enter new 6 char. name of fg 252 ?> = fx60
Enter new 6 char. name of fg 253 ?> = fx61
Enter new 6 char. name of fg 254 ?> = fx62
Enter new 6 char. name of fg 255 ?> = fx63
Enter new 6 char. name of fg 256 ?> = fx64
Enter new 6 char. name of fg 257 ?> = fx65
Enter new 6 char. name of fg 258 ?> = fx66
Enter new 6 char. name of fg 259 ?> = fx67
Enter new 6 char. name of fg 260 ?> = fx68
Enter new 6 char. name of fg 261 ?> = fx69
Enter new 6 char. name of fg 262 ?> = fx70
Enter new 6 char. name of fg 263 ?> = fx71
Enter new 6 char. name of fg 264 ?> = fx72
Enter new 6 char. name of fg 265 ?> = fx73
Enter new 6 char. name of fg 266 ?> = fx74
Enter new 6 char. name of fg 267 ?> = fx75
Enter new 6 char. name of fg 268 ?> = fx76
Enter new 6 char. name of fg 269 ?> = fx77
Enter new 6 char. name of fg 270 ?> = fx78
Enter new 6 char. name of fg 271 ?> = fx79
Enter new 6 char. name of fg 272 ?> = fx80
Enter new 6 char. name of fg 273 ?> = fx81
Enter new 6 char. name of fg 274 ?> = fx82
Enter new 6 char. name of fg 275 ?> = fx83
Enter new 6 char. name of fg 276 ?> = fx84
Enter new 6 char. name of fg 277 ?> = fx85
Enter new 6 char. name of fg 278 ?> = fx86
Enter new 6 char. name of fg 279 ?> = fx87
Enter new 6 char. name of fg 280 ?> = fx88
Enter new 6 char. name of fg 281 ?> = fx89
Enter new 6 char. name of fg 282 ?> = fx90
Enter new 6 char. name of fg 283 ?> = fx91
Enter new 6 char. name of fg 284 ?> = fx92
Enter new 6 char. name of fg 285 ?> = fx93
Enter new 6 char. name of fg 286 ?> = fx94
Enter new 6 char. name of fg 287 ?> = fx95
Enter new 6 char. name of fg 288 ?> = fx96
Enter new 6 char. name of fg 289 ?> = fx97
Enter new 6 char. name of fg 290 ?> = fx98
Enter new 6 char. name of fg 291 ?> = fx99
Enter new 6 char. name of fg 292 ?> = fx100
Enter new 6 char. name of fg 293 ?> = fx101
Enter new 6 char. name of fg 294 ?> = fx102
```

```
Enter new 6 char. name of fg 295 ?> = fx103
Enter new 6 char. name of fg 296 ?> = fx104
Enter new 6 char. name of fg 297 ?> = fx105
Enter new 6 char. name of fg 298 ?> = fx106
Enter new 6 char. name of fg 299 ?> = fx107
Enter new 6 char. name of fg 300 ?> = fx108
Enter new 6 char. name of fg 301 ?> = fx109
Enter new 6 char. name of fg 302 ?> = fx110
Enter new 6 char. name of fg 303 ?> = fx111
Enter new 6 char. name of fg 304 ?> = fx112
Enter new 6 char. name of fg 305 ?> = fx113
Enter new 6 char. name of fg 306 ?> = fx114
Enter new 6 char. name of fg 307 ?> = fx115
Enter new 6 char. name of fg 308 ?> = fx116
Enter new 6 char. name of fg 309 ?> = fx117
Enter new 6 char. name of fg 310 ?> = fx118
Enter new 6 char. name of fg 311 ?> = fx119
Enter new 6 char. name of fg 312 ?> = fx120
Enter new 6 char. name of fg 313 ?> = fx121
Enter new 6 char. name of fg 314 ?> = fx122
Enter new 6 char. name of fg 315 ?> = fx123
Enter new 6 char. name of fg 316 ?> = fx124
Enter new 6 char. name of fg 317 ?> = fx125
Enter new 6 char. name of fg 318 ?> = fx126
Enter new 6 char. name of fg 319 ?> = fx127
Enter new 6 char. name of fg 320 ?> = fx128
Enter new 6 char. name of fg 321 ?> = fx129
Enter new 6 char. name of fg 322 ?> = fx130
Enter new 6 char. name of fg 323 ?> = fx131
Enter new 6 char. name of fg 324 ?> = fx132
Enter new 6 char. name of fg 325 ?> = fx133
Enter new 6 char. name of fg 326 ?> = fx134
Enter new 6 char. name of fg 327 ?> = fx135
Enter new 6 char. name of fg 328 ?> = fx136
Enter new 6 char. name of fg 329 ?> = fx137
Enter new 6 char. name of fg 330 ?> = fx138
Enter new 6 char. name of fg 331 ?> = fx139
Enter new 6 char. name of fg 332 ?> = fx140
Enter new 6 char. name of fg 333 ?> = fx141
Enter new 6 char. name of fg 334 ?> = fx142
Enter new 6 char. name of fg 335 ?> = fx143
Enter new 6 char. name of fg 336 ?> = fx144
Enter new 6 char. name of fg 337 ?> = fx145
Enter new 6 char. name of fg 338 ?> = fx146
Enter new 6 char. name of fg 339 ?> = fx147
Enter new 6 char. name of fg 340 ?> = fx148
Enter new 6 char. name of fg 341 ?> = fx149
Enter new 6 char. name of fg 342 ?> = fx150
Enter new 6 char. name of fg 343 ?> = fx151
Enter new 6 char. name of fg 344 ?> = fx152
```

```
Enter new 6 char. name of fg 345 ?> = fx153
Enter new 6 char. name of fg 346 ?> = fx154
Enter new 6 char. name of fg 347 ?> = fx155
Enter new 6 char. name of fg 348 ?> = fx156
Enter new 6 char. name of fg 349 ?> = fx157
Enter new 6 char. name of fg 350 ?> = fx158
Enter new 6 char. name of fg 351 ?> = fx159
Enter new 6 char. name of fg 352 ?> = fx160
Enter new 6 char. name of fg 353 ?> = fx161
Enter new 6 char. name of fg 354 ?> = fx162
Enter new 6 char. name of fg 355 ?> = fx163
Enter new 6 char. name of fg 356 ?> = fx164
Enter new 6 char. name of fg 357 ?> = fx165
Enter new 6 char. name of fg 358 ?> = fx166
Enter new 6 char. name of fg 359 ?> = fx167
Enter new 6 char. name of fg 360 ?> = fx168
Enter new 6 char. name of fg 361 ?> = fx169
Enter new 6 char. name of fg 362 ?> = fx170
Enter new 6 char. name of fg 363 ?> = fx171
Enter new 6 char. name of fg 364 ?> = fx172
Enter new 6 char. name of fg 365 ?> = fx173
Enter new 6 char. name of fg 366 ?> = fx174
Enter new 6 char. name of fg 367 ?> = fx175
Enter new 6 char. name of fg 368 ?> = fx176
Enter new 6 char. name of fg 369 ?> = fx177
Enter new 6 char. name of fg 370 ?> = fx178
Enter new 6 char. name of fg 371 ?> = fx179
Enter new 6 char. name of fg 372 ?> = fx180
Enter new 6 char. name of fg 373 ?> = fx181
Enter new 6 char. name of fg 374 ?> = fx182
Enter new 6 char. name of fg 375 ?> = fx183
Enter new 6 char. name of fg 376 ?> = fx184
Enter new 6 char. name of fg 377 ?> = fx185
Enter new 6 char. name of fg 378 ?> = fx186
Enter new 6 char. name of fg 379 ?> = fx187
Enter new 6 char. name of fg 380 ?> = fx188
Enter new 6 char. name of fg 381 ?> = fx189
Enter new 6 char. name of fg 382 ?> = fx190
Enter new 6 char. name of fg 383 ?> = fx191
Enter new 6 char. name of fg 384 ?> = fa10
Enter new 6 char. name of fg 385 ?> = fa11
Enter new 6 char. name of fg 386 ?> = fa12
Enter new 6 char. name of fg 387 ?> = fa13
Enter new 6 char. name of fg 388 ?> = fa14
Enter new 6 char. name of fg 389 ?> = fa15
Enter new 6 char. name of fg 390 ?> = fa16
Enter new 6 char. name of fg 391 ?> = fa17
Enter new 6 char. name of fg 392 ?> = fa18
Enter new 6 char. name of fg 393 ?> = fa19
Enter new 6 char. name of fg 394 ?> = fa110
```

```
Enter new 6 char. name of fg 395 ?> = fal11
Enter new 6 char. name of fg 396 ?> = fal12
Enter new 6 char. name of fg 397 ?> = fal13
Enter new 6 char. name of fg 398 ?> = fal14
Enter new 6 char. name of fg 399 ?> = fal15
Enter new 6 char. name of fg 400 ?> = fal16
Enter new 6 char. name of fg 401 ?> = fal17
Enter new 6 char. name of fg 402 ?> = fal18
Enter new 6 char. name of fg 403 ?> = fal19
Enter new 6 char. name of fg 404 ?> = fal20
Enter new 6 char. name of fg 405 ?> = fal21
Enter new 6 char. name of fg 406 ?> = fal22
Enter new 6 char. name of fg 407 ?> = fal23
Enter new 6 char. name of fg 408 ?> = fal24
Enter new 6 char. name of fg 409 ?> = fal25
Enter new 6 char. name of fg 410 ?> = fal26
Enter new 6 char. name of fg 411 ?> = fal27
Enter new 6 char. name of fg 412 ?> = fal28
Enter new 6 char. name of fg 413 ?> = fal29
Enter new 6 char. name of fg 414 ?> = fal30
Enter new 6 char. name of fg 415 ?> = fal31
Enter new 6 char. name of fg 416 ?> = fal32
Enter new 6 char. name of fg 417 ?> = fal33
Enter new 6 char. name of fg 418 ?> = fal34
Enter new 6 char. name of fg 419 ?> = fal35
Enter new 6 char. name of fg 420 ?> = fal36
Enter new 6 char. name of fg 421 ?> = fal37
Enter new 6 char. name of fg 422 ?> = fal38
Enter new 6 char. name of fg 423 ?> = fal39
Enter new 6 char. name of fg 424 ?> = fal40
Enter new 6 char. name of fg 425 ?> = fal41
Enter new 6 char. name of fg 426 ?> = fal42
Enter new 6 char. name of fg 427 ?> = fal43
Enter new 6 char. name of fg 428 ?> = fal44
Enter new 6 char. name of fg 429 ?> = fal45
Enter new 6 char. name of fg 430 ?> = fal46
Enter new 6 char. name of fg 431 ?> = fal47
Enter new 6 char. name of fg 432 ?> = fal48
Enter new 6 char. name of fg 433 ?> = fal49
Enter new 6 char. name of fg 434 ?> = fal50
Enter new 6 char. name of fg 435 ?> = fal51
Enter new 6 char. name of fg 436 ?> = fal52
Enter new 6 char. name of fg 437 ?> = fal53
Enter new 6 char. name of fg 438 ?> = fal54
Enter new 6 char. name of fg 439 ?> = fal55
Enter new 6 char. name of fg 440 ?> = fal56
Enter new 6 char. name of fg 441 ?> = fal57
Enter new 6 char. name of fg 442 ?> = fal58
Enter new 6 char. name of fg 443 ?> = fal59
Enter new 6 char. name of fg 444 ?> = fal60
```

```
Enter new 6 char. name of fg 445 ?> = fal61
Enter new 6 char. name of fg 446 ?> = fal62
Enter new 6 char. name of fg 447 ?> = fal63
Enter new 6 char. name of fg 448 ?> = fal64
Enter new 6 char. name of fg 449 ?> = fal65
Enter new 6 char. name of fg 450 ?> = fal66
Enter new 6 char. name of fg 451 ?> = fal67
Enter new 6 char. name of fg 452 ?> = fal68
Enter new 6 char. name of fg 453 ?> = fal69
Enter new 6 char. name of fg 454 ?> = fal70
Enter new 6 char. name of fg 455 ?> = fal71
Enter new 6 char. name of fg 456 ?> = fal72
Enter new 6 char. name of fg 457 ?> = fal73
Enter new 6 char. name of fg 458 ?> = fal74
Enter new 6 char. name of fg 459 ?> = fal75
Enter new 6 char. name of fg 460 ?> = fal76
Enter new 6 char. name of fg 461 ?> = fal77
Enter new 6 char. name of fg 462 ?> = fal78
Enter new 6 char. name of fg 463 ?> = fal79
Enter new 6 char. name of fg 464 ?> = fal80
Enter new 6 char. name of fg 465 ?> = fal81
Enter new 6 char. name of fg 466 ?> = fal82
Enter new 6 char. name of fg 467 ?> = fal83
Enter new 6 char. name of fg 468 ?> = fal84
Enter new 6 char. name of fg 469 ?> = fal85
Enter new 6 char. name of fg 470 ?> = fal86
Enter new 6 char. name of fg 471 ?> = fal87
Enter new 6 char. name of fg 472 ?> = fal88
Enter new 6 char. name of fg 473 ?> = fal89
Enter new 6 char. name of fg 474 ?> = fal90
Enter new 6 char. name of fg 475 ?> = fal91
Enter new 6 char. name of fg 476 ?> = fal92
Enter new 6 char. name of fg 477 ?> = fal93
Enter new 6 char. name of fg 478 ?> = fal94
Enter new 6 char. name of fg 479 ?> = fal95
Enter new 6 char. name of fg 480 ?> = fal96
Enter new 6 char. name of fg 481 ?> = fal97
Enter new 6 char. name of fg 482 ?> = fal98
Enter new 6 char. name of fg 483 ?> = fal99
Enter new 6 char. name of fg 484 ?> = fal100
Enter new 6 char. name of fg 485 ?> = fal101
Enter new 6 char. name of fg 486 ?> = fal102
Enter new 6 char. name of fg 487 ?> = fal103
Enter new 6 char. name of fg 488 ?> = fal104
Enter new 6 char. name of fg 489 ?> = fal105
Enter new 6 char. name of fg 490 ?> = fal106
Enter new 6 char. name of fg 491 ?> = fal107
Enter new 6 char. name of fg 492 ?> = fal108
Enter new 6 char. name of fg 493 ?> = fal109
Enter new 6 char. name of fg 494 ?> = fal110
```

```
Enter new 6 char. name of fg 495 ?> = fal111
Enter new 6 char. name of fg 496 ?> = fal112
Enter new 6 char. name of fg 497 ?> = fal113
Enter new 6 char. name of fg 498 ?> = fal114
Enter new 6 char. name of fg 499 ?> = fal115
Enter new 6 char. name of fg 500 ?> = fal116
Enter new 6 char. name of fg 501 ?> = fal117
Enter new 6 char. name of fg 502 ?> = fal118
Enter new 6 char. name of fg 503 ?> = fal119
Enter new 6 char. name of fg 504 ?> = fal120
Enter new 6 char. name of fg 505 ?> = fal121
Enter new 6 char. name of fg 506 ?> = fal122
Enter new 6 char. name of fg 507 ?> = fal123
Enter new 6 char. name of fg 508 ?> = fal124
Enter new 6 char. name of fg 509 ?> = fal125
Enter new 6 char. name of fg 510 ?> = fal126
Enter new 6 char. name of fg 511 ?> = fal127
Enter new 6 char. name of fg 512 ?> = fal128
Enter new 6 char. name of fg 513 ?> = fal129
Enter new 6 char. name of fg 514 ?> = fal130
Enter new 6 char. name of fg 515 ?> = fal131
Enter new 6 char. name of fg 516 ?> = fal132
Enter new 6 char. name of fg 517 ?> = fal133
Enter new 6 char. name of fg 518 ?> = fal134
Enter new 6 char. name of fg 519 ?> = fal135
Enter new 6 char. name of fg 520 ?> = fal136
Enter new 6 char. name of fg 521 ?> = fal137
Enter new 6 char. name of fg 522 ?> = fal138
Enter new 6 char. name of fg 523 ?> = fal139
Enter new 6 char. name of fg 524 ?> = fal140
Enter new 6 char. name of fg 525 ?> = fal141
Enter new 6 char. name of fg 526 ?> = fal142
Enter new 6 char. name of fg 527 ?> = fal143
Enter new 6 char. name of fg 528 ?> = fal144
Enter new 6 char. name of fg 529 ?> = fal145
Enter new 6 char. name of fg 530 ?> = fal146
Enter new 6 char. name of fg 531 ?> = fal147
Enter new 6 char. name of fg 532 ?> = fal148
Enter new 6 char. name of fg 533 ?> = fal149
Enter new 6 char. name of fg 534 ?> = fal150
Enter new 6 char. name of fg 535 ?> = fal151
Enter new 6 char. name of fg 536 ?> = fal152
Enter new 6 char. name of fg 537 ?> = fal153
Enter new 6 char. name of fg 538 ?> = fal154
Enter new 6 char. name of fg 539 ?> = fal155
Enter new 6 char. name of fg 540 ?> = fal156
Enter new 6 char. name of fg 541 ?> = fal157
Enter new 6 char. name of fg 542 ?> = fal158
Enter new 6 char. name of fg 543 ?> = fal159
Enter new 6 char. name of fg 544 ?> = fal160
```

```
Enter new 6 char. name of fg 545 ?> = fal161
Enter new 6 char. name of fg 546 ?> = fal162
Enter new 6 char. name of fg 547 ?> = fal163
Enter new 6 char. name of fg 548 ?> = fal164
Enter new 6 char. name of fg 549 ?> = fal165
Enter new 6 char. name of fg 550 ?> = fal166
Enter new 6 char. name of fg 551 ?> = fal167
Enter new 6 char. name of fg 552 ?> = fal168
Enter new 6 char. name of fg 553 ?> = fal169
Enter new 6 char. name of fg 554 ?> = fal170
Enter new 6 char. name of fg 555 ?> = fal171
Enter new 6 char. name of fg 556 ?> = fal172
Enter new 6 char. name of fg 557 ?> = fal173
Enter new 6 char. name of fg 558 ?> = fal174
Enter new 6 char. name of fg 559 ?> = fal175
Enter new 6 char. name of fg 560 ?> = fal176
Enter new 6 char. name of fg 561 ?> = fal177
Enter new 6 char. name of fg 562 ?> = fal178
Enter new 6 char. name of fg 563 ?> = fal179
Enter new 6 char. name of fg 564 ?> = fal180
Enter new 6 char. name of fg 565 ?> = fal181
Enter new 6 char. name of fg 566 ?> = fal182
Enter new 6 char. name of fg 567 ?> = fal183
Enter new 6 char. name of fg 568 ?> = fal184
Enter new 6 char. name of fg 569 ?> = fal185
Enter new 6 char. name of fg 570 ?> = fal186
Enter new 6 char. name of fg 571 ?> = fal187
Enter new 6 char. name of fg 572 ?> = fal188
Enter new 6 char. name of fg 573 ?> = fal189
Enter new 6 char. name of fg 574 ?> = fal190
Enter new 6 char. name of fg 575 ?> = fal191
Enter new 6 char. name of fg 576 ?> = frv0
Enter new 6 char. name of fg 577 ?> = frv1
Enter new 6 char. name of fg 578 ?> = frv2
Enter new 6 char. name of fg 579 ?> = frv3
Enter new 6 char. name of fg 580 ?> = frv4
Enter new 6 char. name of fg 581 ?> = frv5
Enter new 6 char. name of fg 582 ?> = frv6
Enter new 6 char. name of fg 583 ?> = frv7
Enter new 6 char. name of fg 584 ?> = frv8
Enter new 6 char. name of fg 585 ?> = frv9
Enter new 6 char. name of fg 586 ?> = frv10
Enter new 6 char. name of fg 587 ?> = frv11
Enter new 6 char. name of fg 588 ?> = frv12
Enter new 6 char. name of fg 589 ?> = frv13
Enter new 6 char. name of fg 590 ?> = frv14
Enter new 6 char. name of fg 591 ?> = frv15
Enter new 6 char. name of fg 592 ?> = frv16
Enter new 6 char. name of fg 593 ?> = frv17
Enter new 6 char. name of fg 594 ?> = frv18
```

```
Enter new 6 char. name of fg 595 ?> = frv19
Enter new 6 char. name of fg 596 ?> = frv20
Enter new 6 char. name of fg 597 ?> = frv21
Enter new 6 char. name of fg 598 ?> = frv22
Enter new 6 char. name of fg 599 ?> = frv23
Enter new 6 char. name of fg 600 ?> = frv24
Enter new 6 char. name of fg 601 ?> = frv25
Enter new 6 char. name of fg 602 ?> = frv26
Enter new 6 char. name of fg 603 ?> = frv27
Enter new 6 char. name of fg 604 ?> = frv28
Enter new 6 char. name of fg 605 ?> = frv29
Enter new 6 char. name of fg 606 ?> = frv30
Enter new 6 char. name of fg 607 ?> = frv31
Enter new 6 char. name of fg 608 ?> = frv32
Enter new 6 char. name of fg 609 ?> = frv33
Enter new 6 char. name of fg 610 ?> = frv34
Enter new 6 char. name of fg 611 ?> = frv35
Enter new 6 char. name of fg 612 ?> = frv36
Enter new 6 char. name of fg 613 ?> = frv37
Enter new 6 char. name of fg 614 ?> = frv38
Enter new 6 char. name of fg 615 ?> = frv39
Enter new 6 char. name of fg 616 ?> = frv40
Enter new 6 char. name of fg 617 ?> = frv41
Enter new 6 char. name of fg 618 ?> = frv42
Enter new 6 char. name of fg 619 ?> = frv43
Enter new 6 char. name of fg 620 ?> = frv44
Enter new 6 char. name of fg 621 ?> = frv45
Enter new 6 char. name of fg 622 ?> = frv46
Enter new 6 char. name of fg 623 ?> = frv47
Enter new 6 char. name of fg 624 ?> = frv48
Enter new 6 char. name of fg 625 ?> = frv49
Enter new 6 char. name of fg 626 ?> = frv50
Enter new 6 char. name of fg 627 ?> = frv51
Enter new 6 char. name of fg 628 ?> = frv52
Enter new 6 char. name of fg 629 ?> = frv53
Enter new 6 char. name of fg 630 ?> = frv54
Enter new 6 char. name of fg 631 ?> = frv55
Enter new 6 char. name of fg 632 ?> = frv56
Enter new 6 char. name of fg 633 ?> = frv57
Enter new 6 char. name of fg 634 ?> = frv58
Enter new 6 char. name of fg 635 ?> = frv59
Enter new 6 char. name of fg 636 ?> = frv60
Enter new 6 char. name of fg 637 ?> = frv61
Enter new 6 char. name of fg 638 ?> = frv62
Enter new 6 char. name of fg 639 ?> = frv63
Enter new 6 char. name of fg 640 ?> = frv64
Enter new 6 char. name of fg 641 ?> = frv65
Enter new 6 char. name of fg 642 ?> = frv66
Enter new 6 char. name of fg 643 ?> = frv67
Enter new 6 char. name of fg 644 ?> = frv68
```

```
Enter new 6 char. name of fg 645 ?> = frv69
Enter new 6 char. name of fg 646 ?> = frv70
Enter new 6 char. name of fg 647 ?> = frv71
Enter new 6 char. name of fg 648 ?> = frv72
Enter new 6 char. name of fg 649 ?> = frv73
Enter new 6 char. name of fg 650 ?> = frv74
Enter new 6 char. name of fg 651 ?> = frv75
Enter new 6 char. name of fg 652 ?> = frv76
Enter new 6 char. name of fg 653 ?> = frv77
Enter new 6 char. name of fg 654 ?> = frv78
Enter new 6 char. name of fg 655 ?> = frv79
Enter new 6 char. name of fg 656 ?> = frv80
Enter new 6 char. name of fg 657 ?> = frv81
Enter new 6 char. name of fg 658 ?> = frv82
Enter new 6 char. name of fg 659 ?> = frv83
Enter new 6 char. name of fg 660 ?> = frv84
Enter new 6 char. name of fg 661 ?> = frv85
Enter new 6 char. name of fg 662 ?> = frv86
Enter new 6 char. name of fg 663 ?> = frv87
Enter new 6 char. name of fg 664 ?> = frv88
Enter new 6 char. name of fg 665 ?> = frv89
Enter new 6 char. name of fg 666 ?> = frv90
Enter new 6 char. name of fg 667 ?> = frv91
Enter new 6 char. name of fg 668 ?> = frv92
Enter new 6 char. name of fg 669 ?> = frv93
Enter new 6 char. name of fg 670 ?> = frv94
Enter new 6 char. name of fg 671 ?> = frv95
Enter new 6 char. name of fg 672 ?> = frv96
Enter new 6 char. name of fg 673 ?> = frv97
Enter new 6 char. name of fg 674 ?> = frv98
Enter new 6 char. name of fg 675 ?> = frv99
Enter new 6 char. name of fg 676 ?> = frv100
Enter new 6 char. name of fg 677 ?> = frv101
Enter new 6 char. name of fg 678 ?> = frv102
Enter new 6 char. name of fg 679 ?> = frv103
Enter new 6 char. name of fg 680 ?> = frv104
Enter new 6 char. name of fg 681 ?> = frv105
Enter new 6 char. name of fg 682 ?> = frv106
Enter new 6 char. name of fg 683 ?> = frv107
Enter new 6 char. name of fg 684 ?> = frv108
Enter new 6 char. name of fg 685 ?> = frv109
Enter new 6 char. name of fg 686 ?> = frv110
Enter new 6 char. name of fg 687 ?> = frv111
Enter new 6 char. name of fg 688 ?> = frv112
Enter new 6 char. name of fg 689 ?> = frv113
Enter new 6 char. name of fg 690 ?> = frv114
Enter new 6 char. name of fg 691 ?> = frv115
Enter new 6 char. name of fg 692 ?> = frv116
Enter new 6 char. name of fg 693 ?> = frv117
Enter new 6 char. name of fg 694 ?> = frv118
```

```
Enter new 6 char. name of fg 695 ?> = frv119
Enter new 6 char. name of fg 696 ?> = frv120
Enter new 6 char. name of fg 697 ?> = frv121
Enter new 6 char. name of fg 698 ?> = frv122
Enter new 6 char. name of fg 699 ?> = frv123
Enter new 6 char. name of fg 700 ?> = frv124
Enter new 6 char. name of fg 701 ?> = frv125
Enter new 6 char. name of fg 702 ?> = frv126
Enter new 6 char. name of fg 703 ?> = frv127
Enter new 6 char. name of fg 704 ?> = frv128
Enter new 6 char. name of fg 705 ?> = frv129
Enter new 6 char. name of fg 706 ?> = frv130
Enter new 6 char. name of fg 707 ?> = frv131
Enter new 6 char. name of fg 708 ?> = frv132
Enter new 6 char. name of fg 709 ?> = frv133
Enter new 6 char. name of fg 710 ?> = frv134
Enter new 6 char. name of fg 711 ?> = frv135
Enter new 6 char. name of fg 712 ?> = frv136
Enter new 6 char. name of fg 713 ?> = frv137
Enter new 6 char. name of fg 714 ?> = frv138
Enter new 6 char. name of fg 715 ?> = frv139
Enter new 6 char. name of fg 716 ?> = frv140
Enter new 6 char. name of fg 717 ?> = frv141
Enter new 6 char. name of fg 718 ?> = frv142
Enter new 6 char. name of fg 719 ?> = frv143
Enter new 6 char. name of fg 720 ?> = frv144
Enter new 6 char. name of fg 721 ?> = frv145
Enter new 6 char. name of fg 722 ?> = frv146
Enter new 6 char. name of fg 723 ?> = frv147
Enter new 6 char. name of fg 724 ?> = frv148
Enter new 6 char. name of fg 725 ?> = frv149
Enter new 6 char. name of fg 726 ?> = frv150
Enter new 6 char. name of fg 727 ?> = frv151
Enter new 6 char. name of fg 728 ?> = frv152
Enter new 6 char. name of fg 729 ?> = frv153
Enter new 6 char. name of fg 730 ?> = frv154
Enter new 6 char. name of fg 731 ?> = frv155
Enter new 6 char. name of fg 732 ?> = frv156
Enter new 6 char. name of fg 733 ?> = frv157
Enter new 6 char. name of fg 734 ?> = frv158
Enter new 6 char. name of fg 735 ?> = frv159
Enter new 6 char. name of fg 736 ?> = frv160
Enter new 6 char. name of fg 737 ?> = frv161
Enter new 6 char. name of fg 738 ?> = frv162
Enter new 6 char. name of fg 739 ?> = frv163
Enter new 6 char. name of fg 740 ?> = frv164
Enter new 6 char. name of fg 741 ?> = frv165
Enter new 6 char. name of fg 742 ?> = frv166
Enter new 6 char. name of fg 743 ?> = frv167
Enter new 6 char. name of fg 744 ?> = frv168
```

```
Enter new 6 char. name of fg 745 ?> = frv169
Enter new 6 char. name of fg 746 ?> = frv170
Enter new 6 char. name of fg 747 ?> = frv171
Enter new 6 char. name of fg 748 ?> = frv172
Enter new 6 char. name of fg 749 ?> = frv173
Enter new 6 char. name of fg 750 ?> = frv174
Enter new 6 char. name of fg 751 ?> = frv175
Enter new 6 char. name of fg 752 ?> = frv176
Enter new 6 char. name of fg 753 ?> = frv177
Enter new 6 char. name of fg 754 ?> = frv178
Enter new 6 char. name of fg 755 ?> = frv179
Enter new 6 char. name of fg 756 ?> = frv180
Enter new 6 char. name of fg 757 ?> = frv181
Enter new 6 char. name of fg 758 ?> = frv182
Enter new 6 char. name of fg 759 ?> = frv183
Enter new 6 char. name of fg 760 ?> = frv184
Enter new 6 char. name of fg 761 ?> = frv185
Enter new 6 char. name of fg 762 ?> = frv186
Enter new 6 char. name of fg 763 ?> = frv187
Enter new 6 char. name of fg 764 ?> = frv188
Enter new 6 char. name of fg 765 ?> = frv189
Enter new 6 char. name of fg 766 ?> = frv190
Enter new 6 char. name of fg 767 ?> = frv191
Enter new 6 char. name of fg 768 ?> = fdisp
Enter new 6 char. name of fg 769 ?> = fhold
Enter new 6 char. name of fg 770 ?> = fning
Enter new 6 char. name of fg 771 ?> = fabort
Enter new 6 char. name of fg 772 ?> = fstart
Enter new 6 char. name of fg 773 ?> = ftrip0
Enter new 6 char. name of fg 774 ?> = ftrip1
Enter new 6 char. name of fg 775 ?> = ftrip2
Enter new 6 char. name of fg 776 ?> = fspar0
Enter new 6 char. name of fg 777 ?> = fspar1
Enter new 6 char. name of fg 778 ?> = fspar2
Enter new 6 char. name of fg 779 ?> = fspar3
Enter new 6 char. name of fg 780 ?> = fspar4
Enter new 6 char. name of fg 781 ?> = fspar5
Enter new 6 char. name of fg 782 ?> = fspar6
Enter new 6 char. name of fg 783 ?> = fspar7
MUPF?>  = add 306 in
Enter new 6 char. name of in 0 ?> = line0
Enter new 6 char. name of in 1 ?> = line1
Enter new 6 char. name of in 2 ?> = line2
Enter new 6 char. name of in 3 ?> = line3
Enter new 6 char. name of in 4 ?> = line4
Enter new 6 char. name of in 5 ?> = line5
Enter new 6 char. name of in 6 ?> = line6
Enter new 6 char. name of in 7 ?> = line7
Enter new 6 char. name of in 8 ?> = line8
Enter new 6 char. name of in 9 ?> = line9
```

```
Enter new 6 char. name of in 10 ?> = line10
Enter new 6 char. name of in 11 ?> = line11
Enter new 6 char. name of in 12 ?> = line12
Enter new 6 char. name of in 13 ?> = line13
Enter new 6 char. name of in 14 ?> = line14
Enter new 6 char. name of in 15 ?> = line15
Enter new 6 char. name of in 16 ?> = line16
Enter new 6 char. name of in 17 ?> = line17
Enter new 6 char. name of in 18 ?> = line18
Enter new 6 char. name of in 19 ?> = line19
Enter new 6 char. name of in 20 ?> = line20
Enter new 6 char. name of in 21 ?> = line21
Enter new 6 char. name of in 22 ?> = line22
Enter new 6 char. name of in 23 ?> = line23
Enter new 6 char. name of in 24 ?> = line24
Enter new 6 char. name of in 25 ?> = line25
Enter new 6 char. name of in 26 ?> = line26
Enter new 6 char. name of in 27 ?> = line27
Enter new 6 char. name of in 28 ?> = line28
Enter new 6 char. name of in 29 ?> = line29
Enter new 6 char. name of in 30 ?> = line30
Enter new 6 char. name of in 31 ?> = line31
Enter new 6 char. name of in 32 ?> = line32
Enter new 6 char. name of in 33 ?> = line33
Enter new 6 char. name of in 34 ?> = line34
Enter new 6 char. name of in 35 ?> = line35
Enter new 6 char. name of in 36 ?> = line36
Enter new 6 char. name of in 37 ?> = line37
Enter new 6 char. name of in 38 ?> = line38
Enter new 6 char. name of in 39 ?> = line39
Enter new 6 char. name of in 40 ?> = line40
Enter new 6 char. name of in 41 ?> = line41
Enter new 6 char. name of in 42 ?> = line42
Enter new 6 char. name of in 43 ?> = line43
Enter new 6 char. name of in 44 ?> = line44
Enter new 6 char. name of in 45 ?> = line45
Enter new 6 char. name of in 46 ?> = line46
Enter new 6 char. name of in 47 ?> = line47
Enter new 6 char. name of in 48 ?> = bitval
Enter new 6 char. name of in 49 ?> = iloop
Enter new 6 char. name of in 50 ?> = jloop
Enter new 6 char. name of in 51 ?> = kloop
Enter new 6 char. name of in 52 ?> = first
Enter new 6 char. name of in 53 ?> = last
Enter new 6 char. name of in 54 ?> = index
Enter new 6 char. name of in 55 ?> = trip
Enter new 6 char. name of in 56 ?> = nadi
Enter new 6 char. name of in 57 ?> = blkval
Enter new 6 char. name of in 58 ?> = sort
Enter new 6 char. name of in 59 ?> = bval
```

```
Enter new 6 char. name of in 60 ?> = hldstp
Enter new 6 char. name of in 61 ?> = remain
Enter new 6 char. name of in 62 ?> = white
Enter new 6 char. name of in 63 ?> = analog
Enter new 6 char. name of in 64 ?> = errn
Enter new 6 char. name of in 65 ?> = errl
Enter new 6 char. name of in 66 ?> = grafx0
Enter new 6 char. name of in 67 ?> = grafx1
Enter new 6 char. name of in 68 ?> = grafx2
Enter new 6 char. name of in 69 ?> = grafx3
Enter new 6 char. name of in 70 ?> = grafx4
Enter new 6 char. name of in 71 ?> = grafx5
Enter new 6 char. name of in 72 ?> = grafx6
Enter new 6 char. name of in 73 ?> = grafx7
Enter new 6 char. name of in 74 ?> = grafx8
Enter new 6 char. name of in 75 ?> = grafx9
Enter new 6 char. name of in 76 ?> = graf10
Enter new 6 char. name of in 77 ?> = graf11
Enter new 6 char. name of in 78 ?> = graf12
Enter new 6 char. name of in 79 ?> = graf13
Enter new 6 char. name of in 80 ?> = graf14
Enter new 6 char. name of in 81 ?> = graf15
Enter new 6 char. name of in 82 ?> = graf16
Enter new 6 char. name of in 83 ?> = graf17
Enter new 6 char. name of in 84 ?> = graf18
Enter new 6 char. name of in 85 ?> = graf19
Enter new 6 char. name of in 86 ?> = graf20
Enter new 6 char. name of in 87 ?> = graf21
Enter new 6 char. name of in 88 ?> = graf22
Enter new 6 char. name of in 89 ?> = graf23
Enter new 6 char. name of in 90 ?> = graf24
Enter new 6 char. name of in 91 ?> = graf25
Enter new 6 char. name of in 92 ?> = graf26
Enter new 6 char. name of in 93 ?> = graf27
Enter new 6 char. name of in 94 ?> = graf28
Enter new 6 char. name of in 95 ?> = graf29
Enter new 6 char. name of in 96 ?> = graf30
Enter new 6 char. name of in 97 ?> = graf31
Enter new 6 char. name of in 98 ?> = graf32
Enter new 6 char. name of in 99 ?> = graf33
Enter new 6 char. name of in 100 ?> = graf34
Enter new 6 char. name of in 101 ?> = graf35
Enter new 6 char. name of in 102 ?> = graf36
Enter new 6 char. name of in 103 ?> = graf37
Enter new 6 char. name of in 104 ?> = graf38
Enter new 6 char. name of in 105 ?> = graf39
Enter new 6 char. name of in 106 ?> = graf40
Enter new 6 char. name of in 107 ?> = graf41
Enter new 6 char. name of in 108 ?> = graf42
Enter new 6 char. name of in 109 ?> = graf43
```

```
Enter new 6 char. name of in 110 ?> = graf44
Enter new 6 char. name of in 111 ?> = graf45
Enter new 6 char. name of in 112 ?> = graf46
Enter new 6 char. name of in 113 ?> = graf47
Enter new 6 char. name of in 114 ?> = int0
Enter new 6 char. name of in 115 ?> = int1
Enter new 6 char. name of in 116 ?> = int2
Enter new 6 char. name of in 117 ?> = int3
Enter new 6 char. name of in 118 ?> = int4
Enter new 6 char. name of in 119 ?> = int5
Enter new 6 char. name of in 120 ?> = int6
Enter new 6 char. name of in 121 ?> = int7
Enter new 6 char. name of in 122 ?> = int8
Enter new 6 char. name of in 123 ?> = int9
Enter new 6 char. name of in 124 ?> = int10
Enter new 6 char. name of in 125 ?> = int11
Enter new 6 char. name of in 126 ?> = int12
Enter new 6 char. name of in 127 ?> = int13
Enter new 6 char. name of in 128 ?> = int14
Enter new 6 char. name of in 129 ?> = int15
Enter new 6 char. name of in 130 ?> = int16
Enter new 6 char. name of in 131 ?> = int17
Enter new 6 char. name of in 132 ?> = int18
Enter new 6 char. name of in 133 ?> = int19
Enter new 6 char. name of in 134 ?> = int20
Enter new 6 char. name of in 135 ?> = int21
Enter new 6 char. name of in 136 ?> = int22
Enter new 6 char. name of in 137 ?> = int23
Enter new 6 char. name of in 138 ?> = int24
Enter new 6 char. name of in 139 ?> = int25
Enter new 6 char. name of in 140 ?> = int26
Enter new 6 char. name of in 141 ?> = int27
Enter new 6 char. name of in 142 ?> = int28
Enter new 6 char. name of in 143 ?> = int29
Enter new 6 char. name of in 144 ?> = int30
Enter new 6 char. name of in 145 ?> = int31
Enter new 6 char. name of in 146 ?> = int32
Enter new 6 char. name of in 147 ?> = int33
Enter new 6 char. name of in 148 ?> = int34
Enter new 6 char. name of in 149 ?> = int35
Enter new 6 char. name of in 150 ?> = int36
Enter new 6 char. name of in 151 ?> = int37
Enter new 6 char. name of in 152 ?> = int38
Enter new 6 char. name of in 153 ?> = int39
Enter new 6 char. name of in 154 ?> = int40
Enter new 6 char. name of in 155 ?> = int41
Enter new 6 char. name of in 156 ?> = int42
Enter new 6 char. name of in 157 ?> = int43
Enter new 6 char. name of in 158 ?> = int44
Enter new 6 char. name of in 159 ?> = int45
```

```
Enter new 6 char. name of in 160 ?> = int46
Enter new 6 char. name of in 161 ?> = int47
Enter new 6 char. name of in 162 ?> = int48
Enter new 6 char. name of in 163 ?> = int49
Enter new 6 char. name of in 164 ?> = int50
Enter new 6 char. name of in 165 ?> = int51
Enter new 6 char. name of in 166 ?> = int52
Enter new 6 char. name of in 167 ?> = int53
Enter new 6 char. name of in 168 ?> = int54
Enter new 6 char. name of in 169 ?> = int55
Enter new 6 char. name of in 170 ?> = int56
Enter new 6 char. name of in 171 ?> = int57
Enter new 6 char. name of in 172 ?> = int58
Enter new 6 char. name of in 173 ?> = int59
Enter new 6 char. name of in 174 ?> = int60
Enter new 6 char. name of in 175 ?> = int61
Enter new 6 char. name of in 176 ?> = int62
Enter new 6 char. name of in 177 ?> = int63
Enter new 6 char. name of in 178 ?> = int64
Enter new 6 char. name of in 179 ?> = int65
Enter new 6 char. name of in 180 ?> = int66
Enter new 6 char. name of in 181 ?> = int67
Enter new 6 char. name of in 182 ?> = int68
Enter new 6 char. name of in 183 ?> = int69
Enter new 6 char. name of in 184 ?> = int70
Enter new 6 char. name of in 185 ?> = int71
Enter new 6 char. name of in 186 ?> = int72
Enter new 6 char. name of in 187 ?> = int73
Enter new 6 char. name of in 188 ?> = int74
Enter new 6 char. name of in 189 ?> = int75
Enter new 6 char. name of in 190 ?> = int76
Enter new 6 char. name of in 191 ?> = int77
Enter new 6 char. name of in 192 ?> = int78
Enter new 6 char. name of in 193 ?> = int79
Enter new 6 char. name of in 194 ?> = int80
Enter new 6 char. name of in 195 ?> = int81
Enter new 6 char. name of in 196 ?> = int82
Enter new 6 char. name of in 197 ?> = int83
Enter new 6 char. name of in 198 ?> = int84
Enter new 6 char. name of in 199 ?> = int85
Enter new 6 char. name of in 200 ?> = int86
Enter new 6 char. name of in 201 ?> = int87
Enter new 6 char. name of in 202 ?> = int88
Enter new 6 char. name of in 203 ?> = int89
Enter new 6 char. name of in 204 ?> = int90
Enter new 6 char. name of in 205 ?> = int91
Enter new 6 char. name of in 206 ?> = int92
Enter new 6 char. name of in 207 ?> = int93
Enter new 6 char. name of in 208 ?> = int94
Enter new 6 char. name of in 209 ?> = int95
```

```
Enter new 6 char. name of in 210 ?> = int96
Enter new 6 char. name of in 211 ?> = int97
Enter new 6 char. name of in 212 ?> = int98
Enter new 6 char. name of in 213 ?> = int99
Enter new 6 char. name of in 214 ?> = int100
Enter new 6 char. name of in 215 ?> = int101
Enter new 6 char. name of in 216 ?> = int102
Enter new 6 char. name of in 217 ?> = int103
Enter new 6 char. name of in 218 ?> = int104
Enter new 6 char. name of in 219 ?> = int105
Enter new 6 char. name of in 220 ?> = int106
Enter new 6 char. name of in 221 ?> = int107
Enter new 6 char. name of in 222 ?> = int108
Enter new 6 char. name of in 223 ?> = int109
Enter new 6 char. name of in 224 ?> = int110
Enter new 6 char. name of in 225 ?> = int111
Enter new 6 char. name of in 226 ?> = int112
Enter new 6 char. name of in 227 ?> = int113
Enter new 6 char. name of in 228 ?> = int114
Enter new 6 char. name of in 229 ?> = int115
Enter new 6 char. name of in 230 ?> = int116
Enter new 6 char. name of in 231 ?> = int117
Enter new 6 char. name of in 232 ?> = int118
Enter new 6 char. name of in 233 ?> = int119
Enter new 6 char. name of in 234 ?> = int120
Enter new 6 char. name of in 235 ?> = int121
Enter new 6 char. name of in 236 ?> = int122
Enter new 6 char. name of in 237 ?> = int123
Enter new 6 char. name of in 238 ?> = int124
Enter new 6 char. name of in 239 ?> = int125
Enter new 6 char. name of in 240 ?> = int126
Enter new 6 char. name of in 241 ?> = int127
Enter new 6 char. name of in 242 ?> = int128
Enter new 6 char. name of in 243 ?> = int129
Enter new 6 char. name of in 244 ?> = int130
Enter new 6 char. name of in 245 ?> = int131
Enter new 6 char. name of in 246 ?> = int132
Enter new 6 char. name of in 247 ?> = int133
Enter new 6 char. name of in 248 ?> = int134
Enter new 6 char. name of in 249 ?> = int135
Enter new 6 char. name of in 250 ?> = int136
Enter new 6 char. name of in 251 ?> = int137
Enter new 6 char. name of in 252 ?> = int138
Enter new 6 char. name of in 253 ?> = int139
Enter new 6 char. name of in 254 ?> = int140
Enter new 6 char. name of in 255 ?> = int141
Enter new 6 char. name of in 256 ?> = int142
Enter new 6 char. name of in 257 ?> = int143
Enter new 6 char. name of in 258 ?> = int144
Enter new 6 char. name of in 259 ?> = int145
```

```
Enter new 6 char. name of in 260 ?> = int146
Enter new 6 char. name of in 261 ?> = int147
Enter new 6 char. name of in 262 ?> = int148
Enter new 6 char. name of in 263 ?> = int149
Enter new 6 char. name of in 264 ?> = int150
Enter new 6 char. name of in 265 ?> = int151
Enter new 6 char. name of in 266 ?> = int152
Enter new 6 char. name of in 267 ?> = int153
Enter new 6 char. name of in 268 ?> = int154
Enter new 6 char. name of in 269 ?> = int155
Enter new 6 char. name of in 270 ?> = int156
Enter new 6 char. name of in 271 ?> = int157
Enter new 6 char. name of in 272 ?> = int158
Enter new 6 char. name of in 273 ?> = int159
Enter new 6 char. name of in 274 ?> = int160
Enter new 6 char. name of in 275 ?> = int161
Enter new 6 char. name of in 276 ?> = int162
Enter new 6 char. name of in 277 ?> = int163
Enter new 6 char. name of in 278 ?> = int164
Enter new 6 char. name of in 279 ?> = int165
Enter new 6 char. name of in 280 ?> = int166
Enter new 6 char. name of in 281 ?> = int167
Enter new 6 char. name of in 282 ?> = int168
Enter new 6 char. name of in 283 ?> = int169
Enter new 6 char. name of in 284 ?> = int170
Enter new 6 char. name of in 285 ?> = int171
Enter new 6 char. name of in 286 ?> = int172
Enter new 6 char. name of in 287 ?> = int173
Enter new 6 char. name of in 288 ?> = int174
Enter new 6 char. name of in 289 ?> = int175
Enter new 6 char. name of in 290 ?> = int176
Enter new 6 char. name of in 291 ?> = int177
Enter new 6 char. name of in 292 ?> = int178
Enter new 6 char. name of in 293 ?> = int179
Enter new 6 char. name of in 294 ?> = int180
Enter new 6 char. name of in 295 ?> = int181
Enter new 6 char. name of in 296 ?> = int182
Enter new 6 char. name of in 297 ?> = int183
Enter new 6 char. name of in 298 ?> = int184
Enter new 6 char. name of in 299 ?> = int185
Enter new 6 char. name of in 300 ?> = int186
Enter new 6 char. name of in 301 ?> = int187
Enter new 6 char. name of in 302 ?> = int188
Enter new 6 char. name of in 303 ?> = int189
Enter new 6 char. name of in 304 ?> = int190
Enter new 6 char. name of in 305 ?> = int191
MUPF?>  = add 100 fl
Enter new 6 char. name of fl 0 ?> = trip0
Enter new 6 char. name of fl 1 ?> = trip1
Enter new 6 char. name of fl 2 ?> = trip2
```

```
Enter new 6 char. name of fl 3 ?> = trip3
Enter new 6 char. name of fl 4 ?> = trip4
Enter new 6 char. name of fl 5 ?> = trip5
Enter new 6 char. name of fl 6 ?> = trip6
Enter new 6 char. name of fl 7 ?> = trip7
Enter new 6 char. name of fl 8 ?> = trip8
Enter new 6 char. name of fl 9 ?> = trip9
Enter new 6 char. name of fl 10 ?> = ai0
Enter new 6 char. name of fl 11 ?> = ai1
Enter new 6 char. name of fl 12 ?> = ai2
Enter new 6 char. name of fl 13 ?> = ai3
Enter new 6 char. name of fl 14 ?> = ai4
Enter new 6 char. name of fl 15 ?> = ai5
Enter new 6 char. name of fl 16 ?> = ai6
Enter new 6 char. name of fl 17 ?> = ai7
Enter new 6 char. name of fl 18 ?> = ai8
Enter new 6 char. name of fl 19 ?> = ai9
Enter new 6 char. name of fl 20 ?> = ai10
Enter new 6 char. name of fl 21 ?> = ai11
Enter new 6 char. name of fl 22 ?> = ai12
Enter new 6 char. name of fl 23 ?> = ai13
Enter new 6 char. name of fl 24 ?> = ai14
Enter new 6 char. name of fl 25 ?> = ai15
Enter new 6 char. name of fl 26 ?> = ai16
Enter new 6 char. name of fl 27 ?> = ai17
Enter new 6 char. name of fl 28 ?> = ai18
Enter new 6 char. name of fl 29 ?> = ai19
Enter new 6 char. name of fl 30 ?> = ai20
Enter new 6 char. name of fl 31 ?> = ai21
Enter new 6 char. name of fl 32 ?> = ai22
Enter new 6 char. name of fl 33 ?> = ai23
Enter new 6 char. name of fl 34 ?> = ai24
Enter new 6 char. name of fl 35 ?> = ai25
Enter new 6 char. name of fl 36 ?> = ai26
Enter new 6 char. name of fl 37 ?> = ai27
Enter new 6 char. name of fl 38 ?> = ai28
Enter new 6 char. name of fl 39 ?> = ai29
Enter new 6 char. name of fl 40 ?> = ai30
Enter new 6 char. name of fl 41 ?> = ai31
Enter new 6 char. name of fl 42 ?> = ai32
Enter new 6 char. name of fl 43 ?> = ai33
Enter new 6 char. name of fl 44 ?> = ai34
Enter new 6 char. name of fl 45 ?> = ai35
Enter new 6 char. name of fl 46 ?> = ai36
Enter new 6 char. name of fl 47 ?> = ai37
Enter new 6 char. name of fl 48.?> = ai38
Enter new 6 char. name of fl 49 ?> = ai39
Enter new 6 char. name of fl 50 ?> = ai40
Enter new 6 char. name of fl 51 ?> = ai41
Enter new 6 char. name of fl 52 ?> = ai42
```

```
Enter new 6 char. name of fl 53 ?> = ai43
Enter new 6 char. name of fl 54 ?> = ai44
Enter new 6 char. name of fl 55 ?> = ai45
Enter new 6 char. name of fl 56 ?> = ai46
Enter new 6 char. name of fl 57 ?> = ai47
Enter new 6 char. name of fl 58 ?> = ai48
Enter new 6 char. name of fl 59 ?> = ai49
Enter new 6 char. name of fl 60 ?> = ai50
Enter new 6 char. name of fl 61 ?> = ai51
Enter new 6 char. name of fl 62 ?> = ai52
Enter new 6 char. name of fl 63 ?> = ai53
Enter new 6 char. name of fl 64 ?> = ai54
Enter new 6 char. name of fl 65 ?> = ai55
Enter new 6 char. name of fl 66 ?> = ai56
Enter new 6 char. name of fl 67 ?> = ai57
Enter new 6 char. name of fl 68 ?> = ai58
Enter new 6 char. name of fl 69 ?> = ai59
Enter new 6 char. name of fl 70 ?> = ai60
Enter new 6 char. name of fl 71 ?> = ai61
Enter new 6 char. name of fl 72 ?> = ai62
Enter new 6 char. name of fl 73 ?> = ai63
Enter new 6 char. name of fl 74 ?> = ai64
Enter new 6 char. name of fl 75 ?> = ai65
Enter new 6 char. name of fl 76 ?> = ai66
Enter new 6 char. name of fl 77 ?> = ai67
Enter new 6 char. name of fl 78 ?> = ai68
Enter new 6 char. name of fl 79 ?> = ai69
Enter new 6 char. name of fl 80 ?> = ai70
Enter new 6 char. name of fl 81 ?> = ai71
Enter new 6 char. name of fl 82 ?> = ai72
Enter new 6 char. name of fl 83 ?> = ai73
Enter new 6 char. name of fl 84 ?> = ai74
Enter new 6 char. name of fl 85 ?> = ai75
Enter new 6 char. name of fl 86 ?> = ai76
Enter new 6 char. name of fl 87 ?> = ai77
Enter new 6 char. name of fl 88 ?> = ai78
Enter new 6 char. name of fl 89 ?> = ai79
Enter new 6 char. name of fl 90 ?> = float0
Enter new 6 char. name of fl 91 ?> = float1
Enter new 6 char. name of fl 92 ?> = float2
Enter new 6 char. name of fl 93 ?> = float3
Enter new 6 char. name of fl 94 ?> = float4
Enter new 6 char. name of fl 95 ?> = float5
Enter new 6 char. name of fl 96 ?> = float6
Enter new 6 char. name of fl 97 ?> = float7
Enter new 6 char. name of fl 98 ?> = float8
Enter new 6 char. name of fl 99 ?> = float9
MUPF?>  = add 20 pt
Enter new 6 char. name of pt 0 ?> = pdate
Enter new 6 char. name of pt 1 ?> = ptime
```

```
Enter new 6 char. name of pt 2 ?> = pepn
Enter new 6 char. name of pt 3 ?> = pmessg
Enter new 6 char. name of pt 4 ?> = pgrafx
Enter new 6 char. name of pt 5 ?> = palarm
Enter new 6 char. name of pt 6 ?> = palm
Enter new 6 char. name of pt 7 ?> = pstat
Enter new 6 char. name of pt 8 ?> = pnext
Enter new 6 char. name of pt 9 ?> = plast
Enter new 6 char. name of pt 10 ?> = pline
Enter new 6 char. name of pt 11 ?> = ptrip
Enter new 6 char. name of pt 12 ?> = pack
Enter new 6 char. name of pt 13 ?> = pdsabl
Enter new 6 char. name of pt 14 ?> = pback
Enter new 6 char. name of pt 15 ?> = preps
Enter new 6 char. name of pt 16 ?> = ptype
Enter new 6 char. name of pt 17 ?> = prev
Enter new 6 char. name of pt 18 ?> = pspar0
Enter new 6 char. name of pt 19 ?> = pspar1
MUPF?>  = add 111 sg
Enter new 6 char. name of sg 0 ?> = date0
Enter max string length for sg 0  ?> = 10
Enter new 6 char. name of sg 1 ?> = date1
Enter max string length for sg 1  ?> = 10
Enter new 6 char. name of sg 2 ?> = date2
Enter max string length for sg 2  ?> = 10
Enter new 6 char. name of sg 3 ?> = date3
Enter max string length for sg 3  ?> = 10
Enter new 6 char. name of sg 4 ?> = date4
Enter max string length for sg 4  ?> = 10
Enter new 6 char. name of sg 5 ?> = time0
Enter max string length for sg 5  ?> = 8
Enter new 6 char. name of sg 6 ?> = time1
Enter max string length for sg 6  ?> = 8
Enter new 6 char. name of sg 7 ?> = time2
Enter max string length for sg 7  ?> = 8
Enter new 6 char. name of sg 8 ?> = time3
Enter max string length for sg 8  ?> = 8
Enter new 6 char. name of sg 9 ?> = time4
Enter max string length for sg 9  ?> = 8
Enter new 6 char. name of sg 10 ?> = epn0
Enter max string length for sg 10  ?> = 10
Enter new 6 char. name of sg 11 ?> = epn1
Enter max string length for sg 11  ?> = 10
Enter new 6 char. name of sg 12 ?> = epn2
Enter max string length for sg 12  ?> = 10
Enter new 6 char. name of sg 13 ?> = epn3
Enter max string length for sg 13  ?> = 10
Enter new 6 char. name of sg 14 ?> = epn4
Enter max string length for sg 14  ?> = 10
Enter new 6 char. name of sg 15 ?> = mssg0
```

```
Enter max string length for sg 15  ?> = 30
Enter new 6 char. name of sg 16 ?> = mssg1
Enter max string length for sg 16  ?> = 30
Enter new 6 char. name of sg 17 ?> = mssg2
Enter max string length for sg 17  ?> = 30
Enter new 6 char. name of sg 18 ?> = mssg3
Enter max string length for sg 18  ?> = 30
Enter new 6 char. name of sg 19 ?> = mssg4
Enter max string length for sg 19  ?> = 30
Enter new 6 char. name of sg 20 ?> = mssg5
Enter max string length for sg 20  ?> = 30
Enter new 6 char. name of sg 21 ?> = mssg6
Enter max string length for sg 21  ?> = 30
Enter new 6 char. name of sg 22 ?> = mssg7
Enter max string length for sg 22  ?> = 30
Enter new 6 char. name of sg 23 ?> = mssg8
Enter max string length for sg 23  ?> = 30
Enter new 6 char. name of sg 24 ?> = mssg9
Enter max string length for sg 24  ?> = 30
Enter new 6 char. name of sg 25 ?> = mssg10
Enter max string length for sg 25  ?> = 30
Enter new 6 char. name of sg 26 ?> = mssg11
Enter max string length for sg 26  ?> = 30
Enter new 6 char. name of sg 27 ?> = mssg12
Enter max string length for sg 27  ?> = 30
Enter new 6 char. name of sg 28 ?> = mssg13
Enter max string length for sg 28  ?> = 30
Enter new 6 char. name of sg 29 ?> = mssg14
Enter max string length for sg 29  ?> = 30
Enter new 6 char. name of sg 30 ?> = mssg15
Enter max string length for sg 30  ?> = 30
Enter new 6 char. name of sg 31 ?> = mssg16
Enter max string length for sg 31  ?> = 30
Enter new 6 char. name of sg 32 ?> = mssg17
Enter max string length for sg 32  ?> = 30
Enter new 6 char. name of sg 33 ?> = mssg18
Enter max string length for sg 33  ?> = 30
Enter new 6 char. name of sg 34 ?> = mssg19
Enter max string length for sg 34  ?> = 30
Enter new 6 char. name of sg 35 ?> = mssg20
Enter max string length for sg 35  ?> = 30
Enter new 6 char. name of sg 36 ?> = mssg21
Enter max string length for sg 36  ?> = 30
Enter new 6 char. name of sg 37 ?> = mssg22
Enter max string length for sg 37  ?> = 30
Enter new 6 char. name of sg 38 ?> = mssg23
Enter max string length for sg 38  ?> = 30
Enter new 6 char. name of sg 39 ?> = mssg24
Enter max string length for sg 39  ?> = 30
Enter new 6 char. name of sg 40 ?> = mssg25
```

```
Enter max string length for sg 40  ?> = 30
Enter new 6 char. name of sg 41 ?> = mssg26
Enter max string length for sg 41  ?> = 30
Enter new 6 char. name of sg 42 ?> = mssg27
Enter max string length for sg 42  ?> = 30
Enter new 6 char. name of sg 43 ?> = mssg28
Enter max string length for sg 43  ?> = 30
Enter new 6 char. name of sg 44 ?> = mssg29
Enter max string length for sg 44  ?> = 30
Enter new 6 char. name of sg 45 ?> = mssg30
Enter max string length for sg 45  ?> = 30
Enter new 6 char. name of sg 46 ?> = mssg31
Enter max string length for sg 46  ?> = 30
Enter new 6 char. name of sg 47 ?> = mssg32
Enter max string length for sg 47  ?> = 30
Enter new 6 char. name of sg 48 ?> = mssg33
Enter max string length for sg 48  ?> = 30
Enter new 6 char. name of sg 49 ?> = mssg34
Enter max string length for sg 49  ?> = 30
Enter new 6 char. name of sg 50 ?> = mssg35
Enter max string length for sg 50  ?> = 30
Enter new 6 char. name of sg 51 ?> = mssg36
Enter max string length for sg 51  ?> = 30
Enter new 6 char. name of sg 52 ?> = mssg37
Enter max string length for sg 52  ?> = 30
Enter new 6 char. name of sg 53 ?> = mssg38
Enter max string length for sg 53  ?> = 30
Enter new 6 char. name of sg 54 ?> = mssg39
Enter max string length for sg 54  ?> = 30
Enter new 6 char. name of sg 55 ?> = mssg40
Enter max string length for sg 55  ?> = 30
Enter new 6 char. name of sg 56 ?> = mssg41
Enter max string length for sg 56  ?> = 30
Enter new 6 char. name of sg 57 ?> = mssg42
Enter max string length for sg 57  ?> = 30
Enter new 6 char. name of sg 58 ?> = mssg43
Enter max string length for sg 58  ?> = 30
Enter new 6 char. name of sg 59 ?> = mssg44
Enter max string length for sg 59  ?> = 30
Enter new 6 char. name of sg 60 ?> = mssg45
Enter max string length for sg 60  ?> = 30
Enter new 6 char. name of sg 61 ?> = mssg46
Enter max string length for sg 61  ?> = 30
Enter new 6 char. name of sg 62 ?> = mssg47
Enter max string length for sg 62  ?> = 30
Enter new 6 char. name of sg 63 ?> = lsg0
Enter max string length for sg 63  ?> = 30
Enter new 6 char. name of sg 64 ?> = lsg1
Enter max string length for sg 64  ?> = 30
Enter new 6 char. name of sg 65 ?> = lsg2
```

```
Enter max string length for sg 65  ?> = 30
Enter new 6 char. name of sg 66 ?> = lsg3
Enter max string length for sg 66  ?> = 30
Enter new 6 char. name of sg 67 ?> = lsg4
Enter max string length for sg 67  ?> = 30
Enter new 6 char. name of sg 68 ?> = lsg5
Enter max string length for sg 68  ?> = 30
Enter new 6 char. name of sg 69 ?> = lsg6
Enter max string length for sg 69  ?> = 30
Enter new 6 char. name of sg 70 ?> = lsg7
Enter max string length for sg 70  ?> = 30
Enter new 6 char. name of sg 71 ?> = lsg8
Enter max string length for sg 71  ?> = 30
Enter new 6 char. name of sg 72 ?> = lsg9
Enter max string length for sg 72  ?> = 30
Enter new 6 char. name of sg 73 ?> = lsg10
Enter max string length for sg 73  ?> = 30
Enter new 6 char. name of sg 74 ?> = lsg11
Enter max string length for sg 74  ?> = 30
Enter new 6 char. name of sg 75 ?> = lsg12
Enter max string length for sg 75  ?> = 30
Enter new 6 char. name of sg 76 ?> = lsg13
Enter max string length for sg 76  ?> = 30
Enter new 6 char. name of sg 77 ?> = lsg14
Enter max string length for sg 77  ?> = 30
Enter new 6 char. name of sg 78 ?> = lsg15
Enter max string length for sg 78  ?> = 30
Enter new 6 char. name of sg 79 ?> = lsg16
Enter max string length for sg 79  ?> = 30
Enter new 6 char. name of sg 80 ?> = lsg17
Enter max string length for sg 80  ?> = 30
Enter new 6 char. name of sg 81 ?> = lsg18
Enter max string length for sg 81  ?> = 30
Enter new 6 char. name of sg 82 ?> = lsg19
Enter max string length for sg 82  ?> = 30
Enter new 6 char. name of sg 83 ?> = lsg20
Enter max string length for sg 83  ?> = 30
Enter new 6 char. name of sg 84 ?> = lsg21
Enter max string length for sg 84  ?> = 30
Enter new 6 char. name of sg 85 ?> = lsg22
Enter max string length for sg 85  ?> = 30
Enter new 6 char. name of sg 86 ?> = lsg23
Enter max string length for sg 86  ?> = 30
Enter new 6 char. name of sg 87 ?> = lsg24
Enter max string length for sg 87  ?> = 30
Enter new 6 char. name of sg 88 ?> = lsg25
Enter max string length for sg 88  ?> = 30
Enter new 6 char. name of sg 89 ?> = lsg26
Enter max string length for sg 89  ?> = 30
Enter new 6 char. name of sg 90 ?> = lsg27
```

```
Enter max string length for sg 90  ?> = 30
Enter new 6 char. name of sg 91 ?> = lsg28
Enter max string length for sg 91  ?> = 30
Enter new 6 char. name of sg 92 ?> = lsg29
Enter max string length for sg 92  ?> = 30
Enter new 6 char. name of sg 93.?> = lsg30
Enter max string length for sg 93  ?> = 30
Enter new 6 char. name of sg 94 ?> = lsg31
Enter max string length for sg 94  ?> = 30
Enter new 6 char. name of sg 95 ?> = lsg32
Enter max string length for sg 95  ?> = 30
Enter new 6 char. name of sg 96 ?> = lsg33
Enter max string length for sg 96  ?> = 30
Enter new 6 char. name of sg 97 ?> = lsg34
Enter max string length for sg 97  ?> = 30
Enter new 6 char. name of sg 98 ?> = lsg35
Enter max string length for sg 98  ?> = 30
Enter new 6 char. name of sg 99 ?> = lsg36
Enter max string length for sg 99  ?> = 30
Enter new 6 char. name of sg 100 ?> = lsg37
Enter max string length for sg 100  ?> = 30
Enter new 6 char. name of sg 101 ?> = lsg38
Enter max string length for sg 101  ?> = 30
Enter new 6 char. name of sg 102 ?> = lsg39
Enter max string length for sg 102  ?> = 30
Enter new 6 char. name of sg 103 ?> = lsg40
Enter max string length for sg 103  ?> = 30
Enter new 6 char. name of sg 104 ?> = lsg41
Enter max string length for sg 104  ?> = 30
Enter new 6 char. name of sg 105 ?> = lsg42
Enter max string length for sg 105  ?> = 30
Enter new 6 char. name of sg 106 ?> = lsg43
Enter max string length for sg 106  ?> = 30
Enter new 6 char. name of sg 107 ?> = lsg44
Enter max string length for sg 107  ?> = 30
Enter new 6 char. name of sg 108 ?> = lsg45
Enter max string length for sg 108  ?> = 30
Enter new 6 char. name of sg 109 ?> = lsg46
Enter max string length for sg 109  ?> = 30
Enter new 6 char. name of sg 110 ?> = lsg47
Enter max string length for sg 110  ?> = 30
MUPF?>
 end of file encountered
UPF?>
 end of file encountered
```

TNKFRM

```
UPF Version 6.3-2       05-Dec-94 09:42:07
UPF?>  = de tnkfrm
UPF?>  = cr tnkfrm,*
Enter 4 character name of RPF to associate?> = farm
MUPF?>  = add 496 fg
Enter new 6 char. name of fg 0 ?> = falm0
Enter new 6 char. name of fg 1 ?> = falm1
Enter new 6 char. name of fg 2 ?> = falm2
Enter new 6 char. name of fg 3 ?> = falm3
Enter new 6 char. name of fg 4 ?> = falm4
Enter new 6 char. name of fg 5 ?> = falm5
Enter new 6 char. name of fg 6 ?> = falm6
Enter new 6 char. name of fg 7 ?> = falm7
Enter new 6 char. name of fg 8 ?> = falm8
Enter new 6 char. name of fg 9 ?> = falm9
Enter new 6 char. name of fg 10 ?> = falm10
Enter new 6 char. name of fg 11 ?> = falm11
Enter new 6 char. name of fg 12 ?> = falm12
Enter new 6 char. name of fg 13 ?> = falm13
Enter new 6 char. name of fg 14 ?> = falm14
Enter new 6 char. name of fg 15 ?> = falm15
Enter new 6 char. name of fg 16 ?> = falm16
Enter new 6 char. name of fg 17 ?> = falm17
Enter new 6 char. name of fg 18 ?> = falm18
Enter new 6 char. name of fg 19 ?> = falm19
Enter new 6 char. name of fg 20 ?> = falm20
Enter new 6 char. name of fg 21 ?> = falm21
Enter new 6 char. name of fg 22 ?> = falm22
Enter new 6 char. name of fg 23 ?> = falm23
Enter new 6 char. name of fg 24 ?> = falm24
Enter new 6 char. name of fg 25 ?> = falm25
Enter new 6 char. name of fg 26 ?> = falm26
Enter new 6 char. name of fg 27 ?> = falm27
Enter new 6 char. name of fg 28 ?> = falm28
Enter new 6 char. name of fg 29 ?> = falm29
Enter new 6 char. name of fg 30 ?> = falm30
Enter new 6 char. name of fg 31 ?> = falm31
Enter new 6 char. name of fg 32 ?> = falm32
Enter new 6 char. name of fg 33 ?> = falm33
Enter new 6 char. name of fg 34 ?> = falm34
Enter new 6 char. name of fg 35 ?> = falm35
Enter new 6 char. name of fg 36 ?> = falm36
Enter new 6 char. name of fg 37 ?> = falm37
Enter new 6 char. name of fg 38 ?> = falm38
Enter new 6 char. name of fg 39 ?> = falm39
Enter new 6 char. name of fg 40 ?> = falm40
Enter new 6 char. name of fg 41 ?> = falm41
Enter new 6 char. name of fg 42 ?> = falm42
Enter new 6 char. name of fg 43 ?> = falm43
Enter new 6 char. name of fg 44 ?> = falm44
```

```
Enter new 6 char. name of fg 45 ?> = falm45
Enter new 6 char. name of fg 46 ?> = falm46
Enter new 6 char. name of fg 47 ?> = falm47
Enter new 6 char. name of fg 48 ?> = falm48
Enter new 6 char. name of fg 49 ?> = falm49
Enter new 6 char. name of fg 50 ?> = falm50
Enter new 6 char. name of fg 51 ?> = falm51
Enter new 6 char. name of fg 52 ?> = falm52
Enter new 6 char. name of fg 53 ?> = falm53
Enter new 6 char. name of fg 54 ?> = falm54
Enter new 6 char. name of fg 55 ?> = falm55
Enter new 6 char. name of fg 56 ?> = falm56
Enter new 6 char. name of fg 57 ?> = falm57
Enter new 6 char. name of fg 58 ?> = falm58
Enter new 6 char. name of fg 59 ?> = falm59
Enter new 6 char. name of fg 60 ?> = falm60
Enter new 6 char. name of fg 61 ?> = falm61
Enter new 6 char. name of fg 62 ?> = falm62
Enter new 6 char. name of fg 63 ?> = falm63
Enter new 6 char. name of fg 64 ?> = falm64
Enter new 6 char. name of fg 65 ?> = falm65
Enter new 6 char. name of fg 66 ?> = falm66
Enter new 6 char. name of fg 67 ?> = falm67
Enter new 6 char. name of fg 68 ?> = falm68
Enter new 6 char. name of fg 69 ?> = falm69
Enter new 6 char. name of fg 70 ?> = falm70
Enter new 6 char. name of fg 71 ?> = falm71
Enter new 6 char. name of fg 72 ?> = falm72
Enter new 6 char. name of fg 73 ?> = falm73
Enter new 6 char. name of fg 74 ?> = falm74
Enter new 6 char. name of fg 75 ?> = falm75
Enter new 6 char. name of fg 76 ?> = falm76
Enter new 6 char. name of fg 77 ?> = falm77
Enter new 6 char. name of fg 78 ?> = falm78
Enter new 6 char. name of fg 79 ?> = falm79
Enter new 6 char. name of fg 80 ?> = falm80
Enter new 6 char. name of fg 81 ?> = falm81
Enter new 6 char. name of fg 82 ?> = falm82
Enter new 6 char. name of fg 83 ?> = falm83
Enter new 6 char. name of fg 84 ?> = falm84
Enter new 6 char. name of fg 85 ?> = falm85
Enter new 6 char. name of fg 86 ?> = falm86
Enter new 6 char. name of fg 87 ?> = falm87
Enter new 6 char. name of fg 88 ?> = falm88
Enter new 6 char. name of fg 89 ?> = falm89
Enter new 6 char. name of fg 90 ?> = falm90
Enter new 6 char. name of fg 91 ?> = falm91
Enter new 6 char. name of fg 92 ?> = falm92
Enter new 6 char. name of fg 93 ?> = falm93
Enter new 6 char. name of fg 94 ?> = falm94
```

```
Enter new 6 char. name of fg 95 ?> = falm95
Enter new 6 char. name of fg 96 ?> = falm96
Enter new 6 char. name of fg 97 ?> = falm97
Enter new 6 char. name of fg 98 ?> = falm98
Enter new 6 char. name of fg 99 ?> = falm99
Enter new 6 char. name of fg 100 ?> = fal100
Enter new 6 char. name of fg 101 ?> = fal101
Enter new 6 char. name of fg 102 ?> = fal102
Enter new 6 char. name of fg 103 ?> = fal103
Enter new 6 char. name of fg 104 ?> = fal104
Enter new 6 char. name of fg 105 ?> = fal105
Enter new 6 char. name of fg 106 ?> = fal106
Enter new 6 char. name of fg 107 ?> = fal107
Enter new 6 char. name of fg 108 ?> = fal108
Enter new 6 char. name of fg 109 ?> = fal109
Enter new 6 char. name of fg 110 ?> = fal110
Enter new 6 char. name of fg 111 ?> = fal111
Enter new 6 char. name of fg 112 ?> = fpas0
Enter new 6 char. name of fg 113 ?> = fpas1
Enter new 6 char. name of fg 114 ?> = fpas2
Enter new 6 char. name of fg 115 ?> = fpas3
Enter new 6 char. name of fg 116 ?> = fpas4
Enter new 6 char. name of fg 117 ?> = fpas5
Enter new 6 char. name of fg 118 ?> = fpas6
Enter new 6 char. name of fg 119 ?> = fpas7
Enter new 6 char. name of fg 120 ?> = fpas8
Enter new 6 char. name of fg 121 ?> = fpas9
Enter new 6 char. name of fg 122 ?> = fpas10
Enter new 6 char. name of fg 123 ?> = fpas11
Enter new 6 char. name of fg 124 ?> = fpas12
Enter new 6 char. name of fg 125 ?> = fpas13
Enter new 6 char. name of fg 126 ?> = fpas14
Enter new 6 char. name of fg 127 ?> = fpas15
Enter new 6 char. name of fg 128 ?> = fpas16
Enter new 6 char. name of fg 129 ?> = fpas17
Enter new 6 char. name of fg 130 ?> = fpas18
Enter new 6 char. name of fg 131 ?> = fpas19
Enter new 6 char. name of fg 132 ?> = fpas20
Enter new 6 char. name of fg 133 ?> = fpas21
Enter new 6 char. name of fg 134 ?> = fpas22
Enter new 6 char. name of fg 135 ?> = fpas23
Enter new 6 char. name of fg 136 ?> = fpas24
Enter new 6 char. name of fg 137 ?> = fpas25
Enter new 6 char. name of fg 138 ?> = fpas26
Enter new 6 char. name of fg 139 ?> = fpas27
Enter new 6 char. name of fg 140 ?> = fpas28
Enter new 6 char. name of fg 141 ?> = fpas29
Enter new 6 char. name of fg 142 ?> = fpas30
Enter new 6 char. name of fg 143 ?> = fpas31
Enter new 6 char. name of fg 144 ?> = fpas32
```

```
Enter new 6 char. name of fg 145 ?> = fpas33
Enter new 6 char. name of fg 146 ?> = fpas34
Enter new 6 char. name of fg 147 ?> = fpas35
Enter new 6 char. name of fg 148 ?> = fpas36
Enter new 6 char. name of fg 149 ?> = fpas37
Enter new 6 char. name of fg 150 ?> = fpas38
Enter new 6 char. name of fg 151 ?> = fpas39
Enter new 6 char. name of fg 152 ?> = fpas40
Enter new 6 char. name of fg 153 ?> = fpas41
Enter new 6 char. name of fg 154 ?> = fpas42
Enter new 6 char. name of fg 155 ?> = fpas43
Enter new 6 char. name of fg 156 ?> = fpas44
Enter new 6 char. name of fg 157 ?> = fpas45
Enter new 6 char. name of fg 158 ?> = fpas46
Enter new 6 char. name of fg 159 ?> = fpas47
Enter new 6 char. name of fg 160 ?> = fpas48
Enter new 6 char. name of fg 161 ?> = fpas49
Enter new 6 char. name of fg 162 ?> = fpas50
Enter new 6 char. name of fg 163 ?> = fpas51
Enter new 6 char. name of fg 164 ?> = fpas52
Enter new 6 char. name of fg 165 ?> = fpas53
Enter new 6 char. name of fg 166 ?> = fpas54
Enter new 6 char. name of fg 167 ?> = fpas55
Enter new 6 char. name of fg 168 ?> = fpas56
Enter new 6 char. name of fg 169 ?> = fpas57
Enter new 6 char. name of fg 170 ?> = fpas58
Enter new 6 char. name of fg 171 ?> = fpas59
Enter new 6 char. name of fg 172 ?> = fpas60
Enter new 6 char. name of fg 173 ?> = fpas61
Enter new 6 char. name of fg 174 ?> = fpas62
Enter new 6 char. name of fg 175 ?> = fpas63
Enter new 6 char. name of fg 176 ?> = fpas64
Enter new 6 char. name of fg 177 ?> = fpas65
Enter new 6 char. name of fg 178 ?> = fpas66
Enter new 6 char. name of fg 179 ?> = fpas67
Enter new 6 char. name of fg 180 ?> = fpas68
Enter new 6 char. name of fg 181 ?> = fpas69
Enter new 6 char. name of fg 182 ?> = fpas70
Enter new 6 char. name of fg 183 ?> = fpas71
Enter new 6 char. name of fg 184 ?> = fpas72
Enter new 6 char. name of fg 185 ?> = fpas73
Enter new 6 char. name of fg 186 ?> = fpas74
Enter new 6 char. name of fg 187 ?> = fpas75
Enter new 6 char. name of fg 188 ?> = fpas76
Enter new 6 char. name of fg 189 ?> = fpas77
Enter new 6 char. name of fg 190 ?> = fpas78
Enter new 6 char. name of fg 191 ?> = fpas79
Enter new 6 char. name of fg 192 ?> = fpas80
Enter new 6 char. name of fg 193 ?> = fpas81
Enter new 6 char. name of fg 194 ?> = fpas82
```

```
Enter new 6 char. name of fg 195 ?> = fpas83
Enter new 6 char. name of fg 196 ?> = fpas84
Enter new 6 char. name of fg 197 ?> = fpas85
Enter new 6 char. name of fg 198 ?> = fpas86
Enter new 6 char. name of fg 199 ?> = fpas87
Enter new 6 char. name of fg 200 ?> = fpas88
Enter new 6 char. name of fg 201 ?> = fpas89
Enter new 6 char. name of fg 202 ?> = fpas90
Enter new 6 char. name of fg 203 ?> = fpas91
Enter new 6 char. name of fg 204 ?> = fpas92
Enter new 6 char. name of fg 205 ?> = fpas93
Enter new 6 char. name of fg 206 ?> = fpas94
Enter new 6 char. name of fg 207 ?> = fpas95
Enter new 6 char. name of fg 208 ?> = fpas96
Enter new 6 char. name of fg 209 ?> = fpas97
Enter new 6 char. name of fg 210 ?> = fpas98
Enter new 6 char. name of fg 211 ?> = fpas99
Enter new 6 char. name of fg 212 ?> = fpal00
Enter new 6 char. name of fg 213 ?> = fpal01
Enter new 6 char. name of fg 214 ?> = fpal02
Enter new 6 char. name of fg 215 ?> = fpal03
Enter new 6 char. name of fg 216 ?> = fpal04
Enter new 6 char. name of fg 217 ?> = fpal05
Enter new 6 char. name of fg 218 ?> = fpal06
Enter new 6 char. name of fg 219 ?> = fpal07
Enter new 6 char. name of fg 220 ?> = fpal08
Enter new 6 char. name of fg 221 ?> = fpal09
Enter new 6 char. name of fg 222 ?> = fpal10
Enter new 6 char. name of fg 223 ?> = fpal11
Enter new 6 char. name of fg 224 ?> = fskp0
Enter new 6 char. name of fg 225 ?> = fskp1
Enter new 6 char. name of fg 226 ?> = fskp2
Enter new 6 char. name of fg 227 ?> = fskp3
Enter new 6 char. name of fg 228 ?> = fskp4
Enter new 6 char. name of fg 229 ?> = fskp5
Enter new 6 char. name of fg 230 ?> = fskp6
Enter new 6 char. name of fg 231 ?> = fskp7
Enter new 6 char. name of fg 232 ?> = fskp8
Enter new 6 char. name of fg 233 ?> = fskp9
Enter new 6 char. name of fg 234 ?> = fskp10
Enter new 6 char. name of fg 235 ?> = fskp11
Enter new 6 char. name of fg 236 ?> = fskp12
Enter new 6 char. name of fg 237 ?> = fskp13
Enter new 6 char. name of fg 238 ?> = fskp14
Enter new 6 char. name of fg 239 ?> = fskp15
Enter new 6 char. name of fg 240 ?> = fskp16
Enter new 6 char. name of fg 241 ?> = fskp17
Enter new 6 char. name of fg 242 ?> = fskp18
Enter new 6 char. name of fg 243 ?> = fskp19
Enter new 6 char. name of fg 244 ?> = fskp20
```

```
Enter new 6 char. name of fg 245 ?> = fskp21
Enter new 6 char. name of fg 246 ?> = fskp22
Enter new 6 char. name of fg 247 ?> = fskp23
Enter new 6 char. name of fg 248 ?> = fskp24
Enter new 6 char. name of fg 249 ?> = fskp25
Enter new 6 char. name of fg 250 ?> = fskp26
Enter new 6 char. name of fg 251 ?> = fskp27
Enter new 6 char. name of fg 252 ?> = fskp28
Enter new 6 char. name of fg 253 ?> = fskp29
Enter new 6 char. name of fg 254 ?> = fskp30
Enter new 6 char. name of fg 255 ?> = fskp31
Enter new 6 char. name of fg 256 ?> = fskp32
Enter new 6 char. name of fg 257 ?> = fskp33
Enter new 6 char. name of fg 258 ?> = fskp34
Enter new 6 char. name of fg 259 ?> = fskp35
Enter new 6 char. name of fg 260 ?> = fskp36
Enter new 6 char. name of fg 261 ?> = fskp37
Enter new 6 char. name of fg 262 ?> = fskp38
Enter new 6 char. name of fg 263 ?> = fskp39
Enter new 6 char. name of fg 264 ?> = fskp40
Enter new 6 char. name of fg 265 ?> = fskp41
Enter new 6 char. name of fg 266 ?> = fskp42
Enter new 6 char. name of fg 267 ?> = fskp43
Enter new 6 char. name of fg 268 ?> = fskp44
Enter new 6 char. name of fg 269 ?> = fskp45
Enter new 6 char. name of fg 270 ?> = fskp46
Enter new 6 char. name of fg 271 ?> = fskp47
Enter new 6 char. name of fg 272 ?> = fskp48
Enter new 6 char. name of fg 273 ?> = fskp49
Enter new 6 char. name of fg 274 ?> = fskp50
Enter new 6 char. name of fg 275 ?> = fskp51
Enter new 6 char. name of fg 276 ?> = fskp52
Enter new 6 char. name of fg 277 ?> = fskp53
Enter new 6 char. name of fg 278 ?> = fskp54
Enter new 6 char. name of fg 279 ?> = fskp55
Enter new 6 char. name of fg 280 ?> = fskp56
Enter new 6 char. name of fg 281 ?> = fskp57
Enter new 6 char. name of fg 282 ?> = fskp58
Enter new 6 char. name of fg 283 ?> = fskp59
Enter new 6 char. name of fg 284 ?> = fskp60
Enter new 6 char. name of fg 285 ?> = fskp61
Enter new 6 char. name of fg 286 ?> = fskp62
Enter new 6 char. name of fg 287 ?> = fskp63
Enter new 6 char. name of fg 288 ?> = fskp64
Enter new 6 char. name of fg 289 ?> = fskp65
Enter new 6 char. name of fg 290 ?> = fskp66
Enter new 6 char. name of fg 291 ?> = fskp67
Enter new 6 char. name of fg 292 ?> = fskp68
Enter new 6 char. name of fg 293 ?> = fskp69
Enter new 6 char. name of fg 294 ?> = fskp70
```

```
Enter new 6 char. name of fg 295 ?> = fskp71
Enter new 6 char. name of fg 296 ?> = fskp72
Enter new 6 char. name of fg 297 ?> = fskp73
Enter new 6 char. name of fg 298 ?> = fskp74
Enter new 6 char. name of fg 299 ?> = fskp75
Enter new 6 char. name of fg 300 ?> = fskp76
Enter new 6 char. name of fg 301 ?> = fskp77
Enter new 6 char. name of fg 302 ?> = fskp78
Enter new 6 char. name of fg 303 ?> = fskp79
Enter new 6 char. name of fg 304 ?> = fskp80
Enter new 6 char. name of fg 305 ?> = fskp81
Enter new 6 char. name of fg 306 ?> = fskp82
Enter new 6 char. name of fg 307 ?> = fskp83
Enter new 6 char. name of fg 308 ?> = fskp84
Enter new 6 char. name of fg 309 ?> = fskp85
Enter new 6 char. name of fg 310 ?> = fskp86
Enter new 6 char. name of fg 311 ?> = fskp87
Enter new 6 char. name of fg 312 ?> = fskp88
Enter new 6 char. name of fg 313 ?> = fskp89
Enter new 6 char. name of fg 314 ?> = fskp90
Enter new 6 char. name of fg 315 ?> = fskp91
Enter new 6 char. name of fg 316 ?> = fskp92
Enter new 6 char. name of fg 317 ?> = fskp93
Enter new 6 char. name of fg 318 ?> = fskp94
Enter new 6 char. name of fg 319 ?> = fskp95
Enter new 6 char. name of fg 320 ?> = fskp96
Enter new 6 char. name of fg 321 ?> = fskp97
Enter new 6 char. name of fg 322 ?> = fskp98
Enter new 6 char. name of fg 323 ?> = fskp99
Enter new 6 char. name of fg 324 ?> = fsk100
Enter new 6 char. name of fg 325 ?> = fsk101
Enter new 6 char. name of fg 326 ?> = fsk102
Enter new 6 char. name of fg 327 ?> = fsk103
Enter new 6 char. name of fg 328 ?> = fsk104
Enter new 6 char. name of fg 329 ?> = fsk105
Enter new 6 char. name of fg 330 ?> = fsk106
Enter new 6 char. name of fg 331 ?> = fsk107
Enter new 6 char. name of fg 332 ?> = fsk108
Enter new 6 char. name of fg 333 ?> = fsk109
Enter new 6 char. name of fg 334 ?> = fsk110
Enter new 6 char. name of fg 335 ?> = fsk111
Enter new 6 char. name of fg 336 ?> = fx0
Enter new 6 char. name of fg 337 ?> = fx1
Enter new 6 char. name of fg 338 ?> = fx2
Enter new 6 char. name of fg 339 ?> = fx3
Enter new 6 char. name of fg 340 ?> = fx4
Enter new 6 char. name of fg 341 ?> = fx5
Enter new 6 char. name of fg 342 ?> = fx6
Enter new 6 char. name of fg 343 ?> = fx7
Enter new 6 char. name of fg 344 ?> = fx8
```

```
Enter new 6 char. name of fg 345 ?> = fx9
Enter new 6 char. name of fg 346 ?> = fx10
Enter new 6 char. name of fg 347 ?> = fx11
Enter new 6 char. name of fg 348 ?> = fx12
Enter new 6 char. name of fg 349 ?> = fx13
Enter new 6 char. name of fg 350 ?> = fx14
Enter new 6 char. name of fg 351 ?> = fx15
Enter new 6 char. name of fg 352 ?> = fx16
Enter new 6 char. name of fg 353 ?> = fx17
Enter new 6 char. name of fg 354 ?> = fx18
Enter new 6 char. name of fg 355 ?> = fx19
Enter new 6 char. name of fg 356 ?> = fx20
Enter new 6 char. name of fg 357 ?> = fx21
Enter new 6 char. name of fg 358 ?> = fx22
Enter new 6 char. name of fg 359 ?> = fx23
Enter new 6 char. name of fg 360 ?> = fx24
Enter new 6 char. name of fg 361 ?> = fx25
Enter new 6 char. name of fg 362 ?> = fx26
Enter new 6 char. name of fg 363 ?> = fx27
Enter new 6 char. name of fg 364 ?> = fx28
Enter new 6 char. name of fg 365 ?> = fx29
Enter new 6 char. name of fg 366 ?> = fx30
Enter new 6 char. name of fg 367 ?> = fx31
Enter new 6 char. name of fg 368 ?> = fx32
Enter new 6 char. name of fg 369 ?> = fx33
Enter new 6 char. name of fg 370 ?> = fx34
Enter new 6 char. name of fg 371 ?> = fx35
Enter new 6 char. name of fg 372 ?> = fx36
Enter new 6 char. name of fg 373 ?> = fx37
Enter new 6 char. name of fg 374 ?> = fx38
Enter new 6 char. name of fg 375 ?> = fx39
Enter new 6 char. name of fg 376 ?> = fx40
Enter new 6 char. name of fg 377 ?> = fx41
Enter new 6 char. name of fg 378 ?> = fx42
Enter new 6 char. name of fg 379 ?> = fx43
Enter new 6 char. name of fg 380 ?> = fx44
Enter new 6 char. name of fg 381 ?> = fx45
Enter new 6 char. name of fg 382 ?> = fx46
Enter new 6 char. name of fg 383 ?> = fx47
Enter new 6 char. name of fg 384 ?> = fx48
Enter new 6 char. name of fg 385 ?> = fx49
Enter new 6 char. name of fg 386 ?> = fx50
Enter new 6 char. name of fg 387 ?> = fx51
Enter new 6 char. name of fg 388 ?> = fx52
Enter new 6 char. name of fg 389 ?> = fx53
Enter new 6 char. name of fg 390 ?> = fx54
Enter new 6 char. name of fg 391 ?> = fx55
Enter new 6 char. name of fg 392 ?> = fx56
Enter new 6 char. name of fg 393 ?> = fx57
Enter new 6 char. name of fg 394 ?> = fx58
```

```
Enter new 6 char. name of fg 395 ?> = fx59
Enter new 6 char. name of fg 396 ?> = fx60
Enter new 6 char. name of fg 397 ?> = fx61
Enter new 6 char. name of fg 398 ?> = fx62
Enter new 6 char. name of fg 399 ?> = fx63
Enter new 6 char. name of fg 400 ?> = fx64
Enter new 6 char. name of fg 401 ?> = fx65
Enter new 6 char. name of fg 402 ?> = fx66
Enter new 6 char. name of fg 403 ?> = fx67
Enter new 6 char. name of fg 404 ?> = fx68
Enter new 6 char. name of fg 405 ?> = fx69
Enter new 6 char. name of fg 406 ?> = fx70
Enter new 6 char. name of fg 407 ?> = fx71
Enter new 6 char. name of fg 408 ?> = fx72
Enter new 6 char. name of fg 409 ?> = fx73
Enter new 6 char. name of fg 410 ?> = fx74
Enter new 6 char. name of fg 411 ?> = fx75
Enter new 6 char. name of fg 412 ?> = fx76
Enter new 6 char. name of fg 413 ?> = fx77
Enter new 6 char. name of fg 414 ?> = fx78
Enter new 6 char. name of fg 415 ?> = fx79
Enter new 6 char. name of fg 416 ?> = fx80
Enter new 6 char. name of fg 417 ?> = fx81
Enter new 6 char. name of fg 418 ?> = fx82
Enter new 6 char. name of fg 419 ?> = fx83
Enter new 6 char. name of fg 420 ?> = fx84
Enter new 6 char. name of fg 421 ?> = fx85
Enter new 6 char. name of fg 422 ?> = fx86
Enter new 6 char. name of fg 423 ?> = fx87
Enter new 6 char. name of fg 424 ?> = fx88
Enter new 6 char. name of fg 425 ?> = fx89
Enter new 6 char. name of fg 426 ?> = fx90
Enter new 6 char. name of fg 427 ?> = fx91
Enter new 6 char. name of fg 428 ?> = fx92
Enter new 6 char. name of fg 429 ?> = fx93
Enter new 6 char. name of fg 430 ?> = fx94
Enter new 6 char. name of fg 431 ?> = fx95
Enter new 6 char. name of fg 432 ?> = fx96
Enter new 6 char. name of fg 433 ?> = fx97
Enter new 6 char. name of fg 434 ?> = fx98
Enter new 6 char. name of fg 435 ?> = fx99
Enter new 6 char. name of fg 436 ?> = fx100
Enter new 6 char. name of fg 437 ?> = fx101
Enter new 6 char. name of fg 438 ?> = fx102
Enter new 6 char. name of fg 439 ?> = fx103
Enter new 6 char. name of fg 440 ?> = fx104
Enter new 6 char. name of fg 441 ?> = fx105
Enter new 6 char. name of fg 442 ?> = fx106
Enter new 6 char. name of fg 443 ?> = fx107
Enter new 6 char. name of fg 444 ?> = fx108
```

```
Enter new 6 char. name of fg 445 ?> = fx109
Enter new 6 char. name of fg 446 ?> = fx110
Enter new 6 char. name of fg 447 ?> = fleak0
Enter new 6 char. name of fg 448 ?> = fleak1
Enter new 6 char. name of fg 449 ?> = fleak2
Enter new 6 char. name of fg 450 ?> = fleak3
Enter new 6 char. name of fg 451 ?> = fleak4
Enter new 6 char. name of fg 452 ?> = fleak5
Enter new 6 char. name of fg 453 ?> = fleak6
Enter new 6 char. name of fg 454 ?> = fleak7
Enter new 6 char. name of fg 455 ?> = fsn0
Enter new 6 char. name of fg 456 ?> = fsn1
Enter new 6 char. name of fg 457 ?> = fsn2
Enter new 6 char. name of fg 458 ?> = fsn3
Enter new 6 char. name of fg 459 ?> = fsn4
Enter new 6 char. name of fg 460 ?> = fsn5
Enter new 6 char. name of fg 461 ?> = fsn6
Enter new 6 char. name of fg 462 ?> = fsn7
Enter new 6 char. name of fg 463 ?> = fsl0
Enter new 6 char. name of fg 464 ?> = fsl1
Enter new 6 char. name of fg 465 ?> = fsl2
Enter new 6 char. name of fg 466 ?> = fsl3
Enter new 6 char. name of fg 467 ?> = fsl4
Enter new 6 char. name of fg 468 ?> = fsl5
Enter new 6 char. name of fg 469 ?> = fsl6
Enter new 6 char. name of fg 470 ?> = fsl7
Enter new 6 char. name of fg 471 ?> = ffarm
Enter new 6 char. name of fg 472 ?> = fspit
Enter new 6 char. name of fg 473 ?> = fvlpit
Enter new 6 char. name of fg 474 ?> = fhvac
Enter new 6 char. name of fg 475 ?> = f271
Enter new 6 char. name of fg 476 ?> = fflush
Enter new 6 char. name of fg 477 ?> = fleak
Enter new 6 char. name of fg 478 ?> = fdisp
Enter new 6 char. name of fg 479 ?> = fhold
Enter new 6 char. name of fg 480 ?> = fning
Enter new 6 char. name of fg 481 ?> = fabort
Enter new 6 char. name of fg 482 ?> = fstart
Enter new 6 char. name of fg 483 ?> = fldalm
Enter new 6 char. name of fg 484 ?> = fpass
Enter new 6 char. name of fg 485 ?> = ftrip0
Enter new 6 char. name of fg 486 ?> = ftrip1
Enter new 6 char. name of fg 487 ?> = ftrip2
Enter new 6 char. name of fg 488 ?> = ftrip3
Enter new 6 char. name of fg 489 ?> = ftrip4
Enter new 6 char. name of fg 490 ?> = reset
Enter new 6 char. name of fg 491 ?> = ack
Enter new 6 char. name of fg 492 ?> = falmon
Enter new 6 char. name of fg 493 ?> = fack
Enter new 6 char. name of fg 494 ?> = fclear
```

```
Enter new 6 char. name of fg 495 ?> = flgack
MUPF?>  = add 418 in
Enter new 6 char. name of in 0 ?> = bitval
Enter new 6 char. name of in 1 ?> = lmpval
Enter new 6 char. name of in 2 ?> = iloop
Enter new 6 char. name of in 3 ?> = jloop
Enter new 6 char. name of in 4 ?> = kloop
Enter new 6 char. name of in 5 ?> = lloop
Enter new 6 char. name of in 6 ?> = mloop
Enter new 6 char. name of in 7 ?> = index
Enter new 6 char. name of in 8 ?> = last
Enter new 6 char. name of in 9 ?> = curent
Enter new 6 char. name of in 10 ?> = nxt
Enter new 6 char. name of in 11 ?> = analog
Enter new 6 char. name of in 12 ?> = white
Enter new 6 char. name of in 13 ?> = stat
Enter new 6 char. name of in 14 ?> = remain
Enter new 6 char. name of in 15 ?> = trip
Enter new 6 char. name of in 16 ?> = star
Enter new 6 char. name of in 17 ?> = skdkey
Enter new 6 char. name of in 18 ?> = unitt
Enter new 6 char. name of in 19 ?> = adi
Enter new 6 char. name of in 20 ?> = nadi
Enter new 6 char. name of in 21 ?> = blkval
Enter new 6 char. name of in 22 ?> = sort
Enter new 6 char. name of in 23 ?> = bval
Enter new 6 char. name of in 24 ?> = hldstp
Enter new 6 char. name of in 25 ?> = errn
Enter new 6 char. name of in 26 ?> = errl
Enter new 6 char. name of in 27 ?> = int0
Enter new 6 char. name of in 28 ?> = int1
Enter new 6 char. name of in 29 ?> = int2
Enter new 6 char. name of in 30 ?> = int3
Enter new 6 char. name of in 31 ?> = int4
Enter new 6 char. name of in 32 ?> = int5
Enter new 6 char. name of in 33 ?> = int6
Enter new 6 char. name of in 34 ?> = int7
Enter new 6 char. name of in 35 ?> = int8
Enter new 6 char. name of in 36 ?> = int9
Enter new 6 char. name of in 37 ?> = int10
Enter new 6 char. name of in 38 ?> = int11
Enter new 6 char. name of in 39 ?> = int12
Enter new 6 char. name of in 40 ?> = int13
Enter new 6 char. name of in 41 ?> = int14
Enter new 6 char. name of in 42 ?> = int15
Enter new 6 char. name of in 43 ?> = int16
Enter new 6 char. name of in 44 ?> = int17
Enter new 6 char. name of in 45 ?> = int18
Enter new 6 char. name of in 46 ?> = int19
Enter new 6 char. name of in 47 ?> = int20
```

```
Enter new 6 char. name of in 48 ?> = int21
Enter new 6 char. name of in 49 ?> = int22
Enter new 6 char. name of in 50 ?> = int23
Enter new 6 char. name of in 51 ?> = int24
Enter new 6 char. name of in 52 ?> = int25
Enter new 6 char. name of in 53 ?> = int26
Enter new 6 char. name of in 54 ?> = int27
Enter new 6 char. name of in 55 ?> = int28
Enter new 6 char. name of in 56 ?> = int29
Enter new 6 char. name of in 57 ?> = int30
Enter new 6 char. name of in 58 ?> = int31
Enter new 6 char. name of in 59 ?> = int32
Enter new 6 char. name of in 60 ?> = int33
Enter new 6 char. name of in 61 ?> = int34
Enter new 6 char. name of in 62 ?> = int35
Enter new 6 char. name of in 63 ?> = int36
Enter new 6 char. name of in 64 ?> = int37
Enter new 6 char. name of in 65 ?> = int38
Enter new 6 char. name of in 66 ?> = int39
Enter new 6 char. name of in 67 ?> = int40
Enter new 6 char. name of in 68 ?> = int41
Enter new 6 char. name of in 69 ?> = int42
Enter new 6 char. name of in 70 ?> = int43
Enter new 6 char. name of in 71 ?> = int44
Enter new 6 char. name of in 72 ?> = int45
Enter new 6 char. name of in 73 ?> = int46
Enter new 6 char. name of in 74 ?> = int47
Enter new 6 char. name of in 75 ?> = int48
Enter new 6 char. name of in 76 ?> = int49
Enter new 6 char. name of in 77 ?> = int50
Enter new 6 char. name of in 78 ?> = int51
Enter new 6 char. name of in 79 ?> = int52
Enter new 6 char. name of in 80 ?> = int53
Enter new 6 char. name of in 81 ?> = int54
Enter new 6 char. name of in 82 ?> = int55
Enter new 6 char. name of in 83 ?> = int56
Enter new 6 char. name of in 84 ?> = int57
Enter new 6 char. name of in 85 ?> = int58
Enter new 6 char. name of in 86 ?> = int59
Enter new 6 char. name of in 87 ?> = int60
Enter new 6 char. name of in 88 ?> = int61
Enter new 6 char. name of in 89 ?> = int62
Enter new 6 char. name of in 90 ?> = int63
Enter new 6 char. name of in 91 ?> = int64
Enter new 6 char. name of in 92 ?> = int65
Enter new 6 char. name of in 93 ?> = int66
Enter new 6 char. name of in 94 ?> = int67
Enter new 6 char. name of in 95 ?> = int68
Enter new 6 char. name of in 96 ?> = int69
Enter new 6 char. name of in 97 ?> = int70
```

```
Enter new 6 char. name of in 98 ?> = int71
Enter new 6 char. name of in 99 ?> = int72
Enter new 6 char. name of in 100 ?> = int73
Enter new 6 char. name of in 101 ?> = int74
Enter new 6 char. name of in 102 ?> = int75
Enter new 6 char. name of in 103 ?> = int76
Enter new 6 char. name of in 104 ?> = int77
Enter new 6 char. name of in 105 ?> = int78
Enter new 6 char. name of in 106 ?> = int79
Enter new 6 char. name of in 107 ?> = int80
Enter new 6 char. name of in 108 ?> = int81
Enter new 6 char. name of in 109 ?> = int82
Enter new 6 char. name of in 110 ?> = int83
Enter new 6 char. name of in 111 ?> = int84
Enter new 6 char. name of in 112 ?> = int85
Enter new 6 char. name of in 113 ?> = int86
Enter new 6 char. name of in 114 ?> = int87
Enter new 6 char. name of in 115 ?> = int88
Enter new 6 char. name of in 116 ?> = int89
Enter new 6 char. name of in 117 ?> = int90
Enter new 6 char. name of in 118 ?> = int91
Enter new 6 char. name of in 119 ?> = int92
Enter new 6 char. name of in 120 ?> = int93
Enter new 6 char. name of in 121 ?> = int94
Enter new 6 char. name of in 122 ?> = int95
Enter new 6 char. name of in 123 ?> = int96
Enter new 6 char. name of in 124 ?> = int97
Enter new 6 char. name of in 125 ?> = int98
Enter new 6 char. name of in 126 ?> = int99
Enter new 6 char. name of in 127 ?> = ialm0
Enter new 6 char. name of in 128 ?> = ialm1
Enter new 6 char. name of in 129 ?> = ialm2
Enter new 6 char. name of in 130 ?> = ialm3
Enter new 6 char. name of in 131 ?> = ialm4
Enter new 6 char. name of in 132 ?> = ialm5
Enter new 6 char. name of in 133 ?> = ialm6
Enter new 6 char. name of in 134 ?> = ialm7
Enter new 6 char. name of in 135 ?> = ialm8
Enter new 6 char. name of in 136 ?> = ialm9
Enter new 6 char. name of in 137 ?> = ialm10
Enter new 6 char. name of in 138 ?> = ialm11
Enter new 6 char. name of in 139 ?> = ialm12
Enter new 6 char. name of in 140 ?> = ialm13
Enter new 6 char. name of in 141 ?> = ialm14
Enter new 6 char. name of in 142 ?> = ialm15
Enter new 6 char. name of in 143 ?> = ialm16
Enter new 6 char. name of in 144 ?> = ialm17
Enter new 6 char. name of in 145 ?> = ialm18
Enter new 6 char. name of in 146 ?> = ialm19
Enter new 6 char. name of in 147 ?> = ialm20
```

```
Enter new 6 char. name of in 148 ?> = ialm21
Enter new 6 char. name of in 149 ?> = ialm22
Enter new 6 char. name of in 150 ?> = ialm23
Enter new 6 char. name of in 151 ?> = ialm24
Enter new 6 char. name of in 152 ?> = ialm25
Enter new 6 char. name of in 153 ?> = ialm26
Enter new 6 char. name of in 154 ?> = ialm27
Enter new 6 char. name of in 155 ?> = ialm28
Enter new 6 char. name of in 156 ?> = ialm29
Enter new 6 char. name of in 157 ?> = ialm30
Enter new 6 char. name of in 158 ?> = ialm31
Enter new 6 char. name of in 159 ?> = ialm32
Enter new 6 char. name of in 160 ?> = ialm33
Enter new 6 char. name of in 161 ?> = ialm34
Enter new 6 char. name of in 162 ?> = ialm35
Enter new 6 char. name of in 163 ?> = ialm36
Enter new 6 char. name of in 164 ?> = ialm37
Enter new 6 char. name of in 165 ?> = ialm38
Enter new 6 char. name of in 166 ?> = ialm39
Enter new 6 char. name of in 167 ?> = ialm40
Enter new 6 char. name of in 168 ?> = ialm41
Enter new 6 char. name of in 169 ?> = ialm42
Enter new 6 char. name of in 170 ?> = ialm43
Enter new 6 char. name of in 171 ?> = ialm44
Enter new 6 char. name of in 172 ?> = ialm45
Enter new 6 char. name of in 173 ?> = ialm46
Enter new 6 char. name of in 174 ?> = ialm47
Enter new 6 char. name of in 175 ?> = ialm48
Enter new 6 char. name of in 176 ?> = ialm49
Enter new 6 char. name of in 177 ?> = ialm50
Enter new 6 char. name of in 178 ?> = ialm51
Enter new 6 char. name of in 179 ?> = ialm52
Enter new 6 char. name of in 180 ?> = ialm53
Enter new 6 char. name of in 181 ?> = ialm54
Enter new 6 char. name of in 182 ?> = ialm55
Enter new 6 char. name of in 183 ?> = ialm56
Enter new 6 char. name of in 184 ?> = ialm57
Enter new 6 char. name of in 185 ?> = ialm58
Enter new 6 char. name of in 186 ?> = ialm59
Enter new 6 char. name of in 187 ?> = ialm60
Enter new 6 char. name of in 188 ?> = ialm61
Enter new 6 char. name of in 189 ?> = ialm62
Enter new 6 char. name of in 190 ?> = ialm63
Enter new 6 char. name of in 191 ?> = ialm64
Enter new 6 char. name of in 192 ?> = ialm65
Enter new 6 char. name of in 193 ?> = ialm66
Enter new 6 char. name of in 194 ?> = ialm67
Enter new 6 char. name of in 195 ?> = ialm68
Enter new 6 char. name of in 196 ?> = ialm69
Enter new 6 char. name of in 197 ?> = ialm70
```

```
Enter new 6 char. name of in 198 ?> = ialm71
Enter new 6 char. name of in 199 ?> = ialm72
Enter new 6 char. name of in 200 ?> = ialm73
Enter new 6 char. name of in 201 ?> = ialm74
Enter new 6 char. name of in 202 ?> = ialm75
Enter new 6 char. name of in 203 ?> = ialm76
Enter new 6 char. name of in 204 ?> = ialm77
Enter new 6 char. name of in 205 ?> = ialm78
Enter new 6 char. name of in 206 ?> = ialm79
Enter new 6 char. name of in 207 ?> = ialm80
Enter new 6 char. name of in 208 ?> = ialm81
Enter new 6 char. name of in 209 ?> = ialm82
Enter new 6 char. name of in 210 ?> = ialm83
Enter new 6 char. name of in 211 ?> = ialm84
Enter new 6 char. name of in 212 ?> = ialm85
Enter new 6 char. name of in 213 ?> = ialm86
Enter new 6 char. name of in 214 ?> = ialm87
Enter new 6 char. name of in 215 ?> = ialm88
Enter new 6 char. name of in 216 ?> = ialm89
Enter new 6 char. name of in 217 ?> = ialm90
Enter new 6 char. name of in 218 ?> = ialm91
Enter new 6 char. name of in 219 ?> = ialm92
Enter new 6 char. name of in 220 ?> = ialm93
Enter new 6 char. name of in 221 ?> = ialm94
Enter new 6 char. name of in 222 ?> = ialm95
Enter new 6 char. name of in 223 ?> = ialm96
Enter new 6 char. name of in 224 ?> = ialm97
Enter new 6 char. name of in 225 ?> = ialm98
Enter new 6 char. name of in 226 ?> = ialm99
Enter new 6 char. name of in 227 ?> = skid0
Enter new 6 char. name of in 228 ?> = skid1
Enter new 6 char. name of in 229 ?> = skid2
Enter new 6 char. name of in 230 ?> = skid3
Enter new 6 char. name of in 231 ?> = skid4
Enter new 6 char. name of in 232 ?> = skid5
Enter new 6 char. name of in 233 ?> = skid6
Enter new 6 char. name of in 234 ?> = skid7
Enter new 6 char. name of in 235 ?> = skid8
Enter new 6 char. name of in 236 ?> = skid9
Enter new 6 char. name of in 237 ?> = skid10
Enter new 6 char. name of in 238 ?> = skid11
Enter new 6 char. name of in 239 ?> = skid12
Enter new 6 char. name of in 240 ?> = skid13
Enter new 6 char. name of in 241 ?> = skid14
Enter new 6 char. name of in 242 ?> = skid15
Enter new 6 char. name of in 243 ?> = skid16
Enter new 6 char. name of in 244 ?> = skid17
Enter new 6 char. name of in 245 ?> = skid18
Enter new 6 char. name of in 246 ?> = skid19
Enter new 6 char. name of in 247 ?> = skid20
```

```
Enter new 6 char. name of in 248 ?> = skid21
Enter new 6 char. name of in 249 ?> = skid22
Enter new 6 char. name of in 250 ?> = skid23
Enter new 6 char. name of in 251 ?> = skid24
Enter new 6 char. name of in 252 ?> = skid25
Enter new 6 char. name of in 253 ?> = skid26
Enter new 6 char. name of in 254 ?> = skid27
Enter new 6 char. name of in 255 ?> = skid28
Enter new 6 char. name of in 256 ?> = skid29
Enter new 6 char. name of in 257 ?> = skid30
Enter new 6 char. name of in 258 ?> = skid31
Enter new 6 char. name of in 259 ?> = skid32
Enter new 6 char. name of in 260 ?> = skid33
Enter new 6 char. name of in 261 ?> = skid34
Enter new 6 char. name of in 262 ?> = skid35
Enter new 6 char. name of in 263 ?> = skid36
Enter new 6 char. name of in 264 ?> = skid37
Enter new 6 char. name of in 265 ?> = skid38
Enter new 6 char. name of in 266 ?> = skid39
Enter new 6 char. name of in 267 ?> = skid40
Enter new 6 char. name of in 268 ?> = skid41
Enter new 6 char. name of in 269 ?> = skid42
Enter new 6 char. name of in 270 ?> = skid43
Enter new 6 char. name of in 271 ?> = skid44
Enter new 6 char. name of in 272 ?> = skid45
Enter new 6 char. name of in 273 ?> = skid46
Enter new 6 char. name of in 274 ?> = skid47
Enter new 6 char. name of in 275 ?> = skid48
Enter new 6 char. name of in 276 ?> = skid49
Enter new 6 char. name of in 277 ?> = skid50
Enter new 6 char. name of in 278 ?> = skid51
Enter new 6 char. name of in 279 ?> = skid52
Enter new 6 char. name of in 280 ?> = skid53
Enter new 6 char. name of in 281 ?> = skid54
Enter new 6 char. name of in 282 ?> = skid55
Enter new 6 char. name of in 283 ?> = skid56
Enter new 6 char. name of in 284 ?> = skid57
Enter new 6 char. name of in 285 ?> = skid58
Enter new 6 char. name of in 286 ?> = skid59
Enter new 6 char. name of in 287 ?> = skid60
Enter new 6 char. name of in 288 ?> = skid61
Enter new 6 char. name of in 289 ?> = skid62
Enter new 6 char. name of in 290 ?> = skid63
Enter new 6 char. name of in 291 ?> = skid64
Enter new 6 char. name of in 292 ?> = skid65
Enter new 6 char. name of in 293 ?> = skid66
Enter new 6 char. name of in 294 ?> = skid67
Enter new 6 char. name of in 295 ?> = skid68
Enter new 6 char. name of in 296 ?> = skid69
Enter new 6 char. name of in 297 ?> = skid70
```

```
Enter new 6 char. name of in 298 ?> = skid71
Enter new 6 char. name of in 299 ?> = skid72
Enter new 6 char. name of in 300 ?> = skid73
Enter new 6 char. name of in 301 ?> = skid74
Enter new 6 char. name of in 302 ?> = skid75
Enter new 6 char. name of in 303 ?> = skid76
Enter new 6 char. name of in 304 ?> = skid77
Enter new 6 char. name of in 305 ?> = skid78
Enter new 6 char. name of in 306 ?> = skid79
Enter new 6 char. name of in 307 ?> = skid80
Enter new 6 char. name of in 308 ?> = skid81
Enter new 6 char. name of in 309 ?> = skid82
Enter new 6 char. name of in 310 ?> = skid83
Enter new 6 char. name of in 311 ?> = skid84
Enter new 6 char. name of in 312 ?> = skid85
Enter new 6 char. name of in 313 ?> = skid86
Enter new 6 char. name of in 314 ?> = skid87
Enter new 6 char. name of in 315 ?> = skid88
Enter new 6 char. name of in 316 ?> = skid89
Enter new 6 char. name of in 317 ?> = skid90
Enter new 6 char. name of in 318 ?> = skid91
Enter new 6 char. name of in 319 ?> = skid92
Enter new 6 char. name of in 320 ?> = skid93
Enter new 6 char. name of in 321 ?> = skid94
Enter new 6 char. name of in 322 ?> = skid95
Enter new 6 char. name of in 323 ?> = skid96
Enter new 6 char. name of in 324 ?> = skid97
Enter new 6 char. name of in 325 ?> = skid98
Enter new 6 char. name of in 326 ?> = skid99
Enter new 6 char. name of in 327 ?> = dev0
Enter new 6 char. name of in 328 ?> = dev1
Enter new 6 char. name of in 329 ?> = dev2
Enter new 6 char. name of in 330 ?> = dev3
Enter new 6 char. name of in 331 ?> = dev4
Enter new 6 char. name of in 332 ?> = dev5
Enter new 6 char. name of in 333 ?> = dev6
Enter new 6 char. name of in 334 ?> = dev7
Enter new 6 char. name of in 335 ?> = dev8
Enter new 6 char. name of in 336 ?> = dev9
Enter new 6 char. name of in 337 ?> = dev10
Enter new 6 char. name of in 338 ?> = dev11
Enter new 6 char. name of in 339 ?> = dev12
Enter new 6 char. name of in 340 ?> = dev13
Enter new 6 char. name of in 341 ?> = dev14
Enter new 6 char. name of in 342 ?> = dev15
Enter new 6 char. name of in 343 ?> = dev16
Enter new 6 char. name of in 344 ?> = dev17
Enter new 6 char. name of in 345 ?> = dev18
Enter new 6 char. name of in 346 ?> = dev19
Enter new 6 char. name of in 347 ?> = farm
```

```
Enter new 6 char. name of in 348 ?> = serpit
Enter new 6 char. name of in 349 ?> = vlvpit
Enter new 6 char. name of in 350 ?> = hvac
Enter new 6 char. name of in 351 ?> = bld271
Enter new 6 char. name of in 352 ?> = flspit
Enter new 6 char. name of in 353 ?> = lekpit
Enter new 6 char. name of in 354 ?> = drapit
Enter new 6 char. name of in 355 ?> = wspit
Enter new 6 char. name of in 356 ?> = yspit
Enter new 6 char. name of in 357 ?> = rspit
Enter new 6 char. name of in 358 ?> = wvlvpt
Enter new 6 char. name of in 359 ?> = yvlvpt
Enter new 6 char. name of in 360 ?> = rvlvpt
Enter new 6 char. name of in 361 ?> = whvac
Enter new 6 char. name of in 362 ?> = yhvac
Enter new 6 char. name of in 363 ?> = rhvac
Enter new 6 char. name of in 364 ?> = wbld
Enter new 6 char. name of in 365 ?> = ybld
Enter new 6 char. name of in 366 ?> = rbld
Enter new 6 char. name of in 367 ?> = wpump
Enter new 6 char. name of in 368 ?> = ypump
Enter new 6 char. name of in 369 ?> = rpump
Enter new 6 char. name of in 370 ?> = leak0
Enter new 6 char. name of in 371 ?> = leak1
Enter new 6 char. name of in 372 ?> = leak2
Enter new 6 char. name of in 373 ?> = leak3
Enter new 6 char. name of in 374 ?> = leak4
Enter new 6 char. name of in 375 ?> = leak5
Enter new 6 char. name of in 376 ?> = hour
Enter new 6 char. name of in 377 ?> = min
Enter new 6 char. name of in 378 ?> = sec
Enter new 6 char. name of in 379 ?> = dom
Enter new 6 char. name of in 380 ?> = mon
Enter new 6 char. name of in 381 ?> = year
Enter new 6 char. name of in 382 ?> = alm0
Enter new 6 char. name of in 383 ?> = alm1
Enter new 6 char. name of in 384 ?> = alm2
Enter new 6 char. name of in 385 ?> = alm3
Enter new 6 char. name of in 386 ?> = alm4
Enter new 6 char. name of in 387 ?> = alm5
Enter new 6 char. name of in 388 ?> = alm6
Enter new 6 char. name of in 389 ?> = alm7
Enter new 6 char. name of in 390 ?> = alm8
Enter new 6 char. name of in 391 ?> = alm9
Enter new 6 char. name of in 392 ?> = ack0
Enter new 6 char. name of in 393 ?> = ack1
Enter new 6 char. name of in 394 ?> = ack2
Enter new 6 char. name of in 395 ?> = ack3
Enter new 6 char. name of in 396 ?> = ack4
Enter new 6 char. name of in 397 ?> = ack5
```

```
Enter new 6 char. name of in 398 ?> = ack6
Enter new 6 char. name of in 399 ?> = ack7
Enter new 6 char. name of in 400 ?> = ack8
Enter new 6 char. name of in 401 ?> = ack9
Enter new 6 char. name of in 402 ?> = color0
Enter new 6 char. name of in 403 ?> = color1
Enter new 6 char. name of in 404 ?> = color2
Enter new 6 char. name of in 405 ?> = color3
Enter new 6 char. name of in 406 ?> = color4
Enter new 6 char. name of in 407 ?> = color5
Enter new 6 char. name of in 408 ?> = color6
Enter new 6 char. name of in 409 ?> = color7
Enter new 6 char. name of in 410 ?> = grafx0
Enter new 6 char. name of in 411 ?> = grafx1
Enter new 6 char. name of in 412 ?> = grafx2
Enter new 6 char. name of in 413 ?> = grafx3
Enter new 6 char. name of in 414 ?> = grafx4
Enter new 6 char. name of in 415 ?> = grafx5
Enter new 6 char. name of in 416 ?> = grafx6
Enter new 6 char. name of in 417 ?> = grafx7
MUPF?>  = add 20 fl
Enter new 6 char. name of fl 0 ?> = ai0
Enter new 6 char. name of fl 1 ?> = ai1
Enter new 6 char. name of fl 2 ?> = ai2
Enter new 6 char. name of fl 3 ?> = ai3
Enter new 6 char. name of fl 4 ?> = ai4
Enter new 6 char. name of fl 5 ?> = ai5
Enter new 6 char. name of fl 6 ?> = ai6
Enter new 6 char. name of fl 7 ?> = ai7
Enter new 6 char. name of fl 8 ?> = ai8
Enter new 6 char. name of fl 9 ?> = ai9
Enter new 6 char. name of fl 10 ?> = ai10
Enter new 6 char. name of fl 11 ?> = ai11
Enter new 6 char. name of fl 12 ?> = ai12
Enter new 6 char. name of fl 13 ?> = ai13
Enter new 6 char. name of fl 14 ?> = ai14
Enter new 6 char. name of fl 15 ?> = ai15
Enter new 6 char. name of fl 16 ?> = ai16
Enter new 6 char. name of fl 17 ?> = ai17
Enter new 6 char. name of fl 18 ?> = ai18
Enter new 6 char. name of fl 19 ?> = ai19
MUPF?>  = add 13 pt
Enter new 6 char. name of pt 0 ?> = pepn
Enter new 6 char. name of pt 1 ?> = pdin
Enter new 6 char. name of pt 2 ?> = pdev
Enter new 6 char. name of pt 3 ?> = pai
Enter new 6 char. name of pt 4 ?> = pskid
Enter new 6 char. name of pt 5 ?> = pstat
Enter new 6 char. name of pt 6 ?> = ptype
Enter new 6 char. name of pt 7 ?> = prev
```

```
Enter new 6 char. name of pt 8 ?> = psn
Enter new 6 char. name of pt 9 ?> = pmsg0
Enter new 6 char. name of pt 10 ?> = pmsg1
Enter new 6 char. name of pt 11 ?> = ppass
Enter new 6 char. name of pt 12 ?> = pack
MUPF?>  = add 61 sg
Enter new 6 char. name of sg 0 ?> = swhen0
Enter max string length for sg 0  ?> = 8
Enter new 6 char. name of sg 1Enter new 6 char. name of sg 1 ?> = swhen1
Enter max string length for sg 1  ?> = 8
Enter new 6 char. name of sg 2 ?> = swhen2
Enter max string length for sg 2  ?> = 8
Enter new 6 char. name of sg 3 ?> = swhen3
Enter max string length for sg 3  ?> = 8
Enter new 6 char. name of sg 4 ?> = swhen4
Enter max string length for sg 4  ?> = 8
Enter new 6 char. name of sg 5 ?> = swhen5
Enter max string length for sg 5  ?> = 8
Enter new 6 char. name of sg 6 ?> = swhen6
Enter max string length for sg 6  ?> = 8
Enter new 6 char. name of sg 7 ?> = swhen7
Enter max string length for sg 7  ?> = 8
Enter new 6 char. name of sg 8 ?> = ssng0
Enter max string length for sg 8  ?> = 8
Enter new 6 char. name of sg 9 ?> = ssng1
Enter max string length for sg 9  ?> = 8
Enter new 6 char. name of sg 10 ?> = ssng2
Enter max string length for sg 10  ?> = 8
Enter new 6 char. name of sg 11 ?> = ssng3
Enter max string length for sg 11  ?> = 8
Enter new 6 char. name of sg 12 ?> = ssng4
Enter max string length for sg 12  ?> = 8
Enter new 6 char. name of sg 13 ?> = ssng5
Enter max string length for sg 13  ?> = 8
Enter new 6 char. name of sg 14 ?> = ssng6
Enter max string length for sg 14  ?> = 8
Enter new 6 char. name of sg 15 ?> = ssng7
Enter max string length for sg 15  ?> = 8
Enter new 6 char. name of sg 16 ?> = sdate0
Enter max string length for sg 16  ?> = 10
Enter new 6 char. name of sg 17 ?> = sdate1
Enter max string length for sg 17  ?> = 10
Enter new 6 char. name of sg 18 ?> = sdate2
Enter max string length for sg 18  ?> = 10
Enter new 6 char. name of sg 19 ?> = sdate3
Enter max string length for sg 19  ?> = 10
Enter new 6 char. name of sg 20 ?> = sdate4
Enter max string length for sg 20  ?> = 10
Enter new 6 char. name of sg 21 ?> = sdate5
Enter max string length for sg 21  ?> = 10
```

```
Enter new 6 char. name of sg 22 ?> = sdate6
Enter max string length for sg 22  ?> = 10
Enter new 6 char. name of sg 23 ?> = sdate7
Enter max string length for sg 23  ?> = 10
Enter new 6 char. name of sg 24 ?> = epns0
Enter max string length for sg 24  ?> = 10
Enter new 6 char. name of sg 25 ?> = epns1
Enter max string length for sg 25  ?> = 10
Enter new 6 char. name of sg 26 ?> = epns2
Enter max string length for sg 26  ?> = 10
Enter new 6 char. name of sg 27 ?> = epns3
Enter max string length for sg 27  ?> = 10
Enter new 6 char. name of sg 28 ?> = epns4
Enter max string length for sg 28  ?> = 10
Enter new 6 char. name of sg 29 ?> = epns5
Enter max string length for sg 29  ?> = 10
Enter new 6 char. name of sg 30 ?> = epns6
Enter max string length for sg 30  ?> = 10
Enter new 6 char. name of sg 31 ?> = epns7
Enter max string length for sg 31  ?> = 10
Enter new 6 char. name of sg 32 ?> = salm0
Enter max string length for sg 32  ?> = 20
Enter new 6 char. name of sg 33 ?> = salm1
Enter max string length for sg 33  ?> = 20
Enter new 6 char. name of sg 34 ?> = salm2
Enter max string length for sg 34  ?> = 20
Enter new 6 char. name of sg 35 ?> = salm3
Enter max string length for sg 35  ?> = 20
Enter new 6 char. name of sg 36 ?> = salm4
Enter max string length for sg 36  ?> = 20
Enter new 6 char. name of sg 37 ?> = salm5
Enter max string length for sg 37  ?> = 20
Enter new 6 char. name of sg 38 ?> = salm6
Enter max string length for sg 38  ?> = 20
Enter new 6 char. name of sg 39 ?> = salm7
Enter max string length for sg 39  ?> = 20
Enter new 6 char. name of sg 40 ?> = msg0
Enter max string length for sg 40  ?> = 30
Enter new 6 char. name of sg 41 ?> = msg1
Enter max string length for sg 41  ?> = 30
Enter new 6 char. name of sg 42 ?> = msg2
Enter max string length for sg 42  ?> = 30
Enter new 6 char. name of sg 43 ?> = msg3
Enter max string length for sg 43  ?> = 30
Enter new 6 char. name of sg 44 ?> = msg4
Enter max string length for sg 44  ?> = 30
Enter new 6 char. name of sg 45 ?> = msg5
Enter max string length for sg 45  ?> = 30
Enter new 6 char. name of sg 46 ?> = msg6
Enter max string length for sg 46  ?> = 30
```

```
Enter new 6 char. name of sg 47 ?> = msg7
Enter max string length for sg 47  ?> = 30
Enter new 6 char. name of sg 48 ?> = lsg0
Enter max string length for sg 48  ?> = 30
Enter new 6 char. name of sg 49 ?> = lsg1
Enter max string length for sg 49  ?> = 30
Enter new 6 char. name of sg 50 ?> = lsg2
Enter max string length for sg 50  ?> = 30
Enter new 6 char. name of sg 51 ?> = lsg3
Enter max string length for sg 51  ?> = 30
Enter new 6 char. name of sg 52 ?> = lsg4
Enter max string length for sg 52  ?> = 30
Enter new 6 char. name of sg 53 ?> = lsg5
Enter max string length for sg 53  ?> = 30
Enter new 6 char. name of sg 54 ?> = lsg6
Enter max string length for sg 54  ?> = 30
Enter new 6 char. name of sg 55 ?> = lsg7
Enter max string length for sg 55  ?> = 30
Enter new 6 char. name of sg 56 ?> = strng0
Enter max string length for sg 56  ?> = 30
Enter new 6 char. name of sg 57 ?> = strng1
Enter max string length for sg 57  ?> = 30
Enter new 6 char. name of sg 58 ?> = strng2
Enter max string length for sg 58  ?> = 30
Enter new 6 char. name of sg 59 ?> = strng3
Enter max string length for sg 59  ?> = 30
Enter new 6 char. name of sg 60 ?> = strng4
Enter max string length for sg 60  ?> = 30
MUPF?>  =
UPF?>
 end of file encountered
```

BLANKK

```
UPF Version 6.3-2       05-Dec-94 09:39:00
UPF?>  = de blankk
UPF?>  = cr blankk,*
Enter 4 character name of RPF to associate?> = emty
MUPF?>  = add 192 fg
Enter new 6 char. name of fg 0 ?> = falm0
Enter new 6 char. name of fg 1 ?> = falm1
Enter new 6 char. name of fg 2 ?> = falm2
Enter new 6 char. name of fg 3 ?> = falm3
Enter new 6 char. name of fg 4 ?> = falm4
Enter new 6 char. name of fg 5 ?> = falm5
Enter new 6 char. name of fg 6 ?> = falm6
Enter new 6 char. name of fg 7 ?> = falm7
Enter new 6 char. name of fg 8 ?> = falm8
Enter new 6 char. name of fg 9 ?> = falm9
Enter new 6 char. name of fg 10 ?> = falm10
Enter new 6 char. name of fg 11 ?> = falm11
Enter new 6 char. name of fg 12 ?> = falm12
Enter new 6 char. name of fg 13 ?> = falm13
Enter new 6 char. name of fg 14 ?> = falm14
Enter new 6 char. name of fg 15 ?> = falm15
Enter new 6 char. name of fg 16 ?> = falm16
Enter new 6 char. name of fg 17 ?> = falm17
Enter new 6 char. name of fg 18 ?> = falm18
Enter new 6 char. name of fg 19 ?> = falm19
Enter new 6 char. name of fg 20 ?> = falm20
Enter new 6 char. name of fg 21 ?> = falm21
Enter new 6 char. name of fg 22 ?> = falm22
Enter new 6 char. name of fg 23 ?> = falm23
Enter new 6 char. name of fg 24 ?> = falm24
Enter new 6 char. name of fg 25 ?> = falm25
Enter new 6 char. name of fg 26 ?> = falm26
Enter new 6 char. name of fg 27 ?> = falm27
Enter new 6 char. name of fg 28 ?> = falm28
Enter new 6 char. name of fg 29 ?> = falm29
Enter new 6 char. name of fg 30 ?> = falm30
Enter new 6 char. name of fg 31 ?> = falm31
Enter new 6 char. name of fg 32 ?> = falm32
Enter new 6 char. name of fg 33 ?> = falm33
Enter new 6 char. name of fg 34 ?> = falm34
Enter new 6 char. name of fg 35 ?> = falm35
Enter new 6 char. name of fg 36 ?> = falm36
Enter new 6 char. name of fg 37 ?> = falm37
Enter new 6 char. name of fg 38 ?> = falm38
Enter new 6 char. name of fg 39 ?> = falm39
Enter new 6 char. name of fg 40 ?> = falm40
Enter new 6 char. name of fg 41 ?> = falm41
Enter new 6 char. name of fg 42 ?> = falm42
Enter new 6 char. name of fg 43 ?> = falm43
Enter new 6 char. name of fg 44 ?> = falm44
```

```
Enter new 6 char. name of fg 45 ?> = falm45
Enter new 6 char. name of fg 46 ?> = falm46
Enter new 6 char. name of fg 47 ?> = falm47
Enter new 6 char. name of fg 48 ?> = falm48
Enter new 6 char. name of fg 49 ?> = falm49
Enter new 6 char. name of fg 50 ?> = falm50
Enter new 6 char. name of fg 51 ?> = falm51
Enter new 6 char. name of fg 52 ?> = falm52
Enter new 6 char. name of fg 53 ?> = falm53
Enter new 6 char. name of fg 54 ?> = falm54
Enter new 6 char. name of fg 55 ?> = falm55
Enter new 6 char. name of fg 56 ?> = falm56
Enter new 6 char. name of fg 57 ?> = falm57
Enter new 6 char. name of fg 58 ?> = falm58
Enter new 6 char. name of fg 59 ?> = falm59
Enter new 6 char. name of fg 60 ?> = falm60
Enter new 6 char. name of fg 61 ?> = falm61
Enter new 6 char. name of fg 62 ?> = falm62
Enter new 6 char. name of fg 63 ?> = falm63
Enter new 6 char. name of fg 64 ?> = falm64
Enter new 6 char. name of fg 65 ?> = falm65
Enter new 6 char. name of fg 66 ?> = falm66
Enter new 6 char. name of fg 67 ?> = falm67
Enter new 6 char. name of fg 68 ?> = falm68
Enter new 6 char. name of fg 69 ?> = falm69
Enter new 6 char. name of fg 70 ?> = falm70
Enter new 6 char. name of fg 71 ?> = falm71
Enter new 6 char. name of fg 72 ?> = falm72
Enter new 6 char. name of fg 73 ?> = falm73
Enter new 6 char. name of fg 74 ?> = falm74
Enter new 6 char. name of fg 75 ?> = falm75
Enter new 6 char. name of fg 76 ?> = falm76
Enter new 6 char. name of fg 77 ?> = falm77
Enter new 6 char. name of fg 78 ?> = falm78
Enter new 6 char. name of fg 79 ?> = falm79
Enter new 6 char. name of fg 80 ?> = flg0
Enter new 6 char. name of fg 81 ?> = flg1
Enter new 6 char. name of fg 82 ?> = flg2
Enter new 6 char. name of fg 83 ?> = flg3
Enter new 6 char. name of fg 84 ?> = flg4
Enter new 6 char. name of fg 85 ?> = flg5
Enter new 6 char. name of fg 86 ?> = flg6
Enter new 6 char. name of fg 87 ?> = flg7
Enter new 6 char. name of fg 88 ?> = flg8
Enter new 6 char. name of fg 89 ?> = flg9
Enter new 6 char. name of fg 90 ?> = flg10
Enter new 6 char. name of fg 91 ?> = flg11
Enter new 6 char. name of fg 92 ?> = flg12
Enter new 6 char. name of fg 93 ?> = flg13
Enter new 6 char. name of fg 94 ?> = flg14
```

```
Enter new 6 char. name of fg 95 ?> = flg15
Enter new 6 char. name of fg 96 ?> = flg16
Enter new 6 char. name of fg 97 ?> = flg17
Enter new 6 char. name of fg 98 ?> = flg18
Enter new 6 char. name of fg 99 ?> = flg19
Enter new 6 char. name of fg 100 ?> = flg20
Enter new 6 char. name of fg 101 ?> = flg21
Enter new 6 char. name of fg 102 ?> = flg22
Enter new 6 char. name of fg 103 ?> = flg23
Enter new 6 char. name of fg 104 ?> = flg24
Enter new 6 char. name of fg 105 ?> = flg25
Enter new 6 char. name of fg 106 ?> = flg26
Enter new 6 char. name of fg 107 ?> = flg27
Enter new 6 char. name of fg 108 ?> = flg28
Enter new 6 char. name of fg 109 ?> = flg29
Enter new 6 char. name of fg 110 ?> = flg30
Enter new 6 char. name of fg 111 ?> = flg31
Enter new 6 char. name of fg 112 ?> = flg32
Enter new 6 char. name of fg 113 ?> = flg33
Enter new 6 char. name of fg 114 ?> = flg34
Enter new 6 char. name of fg 115 ?> = flg35
Enter new 6 char. name of fg 116 ?> = flg36
Enter new 6 char. name of fg 117 ?> = flg37
Enter new 6 char. name of fg 118 ?> = flg38
Enter new 6 char. name of fg 119 ?> = flg39
Enter new 6 char. name of fg 120 ?> = flg40
Enter new 6 char. name of fg 121 ?> = flg41
Enter new 6 char. name of fg 122 ?> = flg42
Enter new 6 char. name of fg 123 ?> = flg43
Enter new 6 char. name of fg 124 ?> = flg44
Enter new 6 char. name of fg 125 ?> = flg45
Enter new 6 char. name of fg 126 ?> = flg46
Enter new 6 char. name of fg 127 ?> = flg47
Enter new 6 char. name of fg 128 ?> = flg48
Enter new 6 char. name of fg 129 ?> = flg49
Enter new 6 char. name of fg 130 ?> = flg50
Enter new 6 char. name of fg 131 ?> = flg51
Enter new 6 char. name of fg 132 ?> = flg52
Enter new 6 char. name of fg 133 ?> = flg53
Enter new 6 char. name of fg 134 ?> = flg54
Enter new 6 char. name of fg 135 ?> = flg55
Enter new 6 char. name of fg 136 ?> = flg56
Enter new 6 char. name of fg 137 ?> = flg57
Enter new 6 char. name of fg 138 ?> = flg58
Enter new 6 char. name of fg 139 ?> = flg59
Enter new 6 char. name of fg 140 ?> = flg60
Enter new 6 char. name of fg 141 ?> = flg61
Enter new 6 char. name of fg 142 ?> = flg62
Enter new 6 char. name of fg 143 ?> = flg63
Enter new 6 char. name of fg 144 ?> = flg64
```

```
Enter new 6 char. name of fg 145 ?> = flg65
Enter new 6 char. name of fg 146 ?> = flg66
Enter new 6 char. name of fg 147 ?> = flg67
Enter new 6 char. name of fg 148 ?> = flg68
Enter new 6 char. name of fg 149 ?> = flg69
Enter new 6 char. name of fg 150 ?> = flg70
Enter new 6 char. name of fg 151 ?> = flg71
Enter new 6 char. name of fg 152 ?> = flg72
Enter new 6 char. name of fg 153 ?> = flg73
Enter new 6 char. name of fg 154 ?> = flg74
Enter new 6 char. name of fg 155 ?> = flg75
Enter new 6 char. name of fg 156 ?> = flg76
Enter new 6 char. name of fg 157 ?> = flg77
Enter new 6 char. name of fg 158 ?> = flg78
Enter new 6 char. name of fg 159 ?> = flg79
Enter new 6 char. name of fg 160 ?> = fx0
Enter new 6 char. name of fg 161 ?> = fx1
Enter new 6 char. name of fg 162 ?> = fx2
Enter new 6 char. name of fg 163 ?> = fx3
Enter new 6 char. name of fg 164 ?> = fx4
Enter new 6 char. name of fg 165 ?> = fx5
Enter new 6 char. name of fg 166 ?> = fx6
Enter new 6 char. name of fg 167 ?> = fx7
Enter new 6 char. name of fg 168 ?> = fsl0
Enter new 6 char. name of fg 169 ?> = fsl1
Enter new 6 char. name of fg 170 ?> = fsl2
Enter new 6 char. name of fg 171 ?> = fsl3
Enter new 6 char. name of fg 172 ?> = fsl4
Enter new 6 char. name of fg 173 ?> = fsl5
Enter new 6 char. name of fg 174 ?> = fsl6
Enter new 6 char. name of fg 175 ?> = fsl7
Enter new 6 char. name of fg 176 ?> = fhold
Enter new 6 char. name of fg 177 ?> = fning
Enter new 6 char. name of fg 178 ?> = fabort
Enter new 6 char. name of fg 179 ?> = fstart
Enter new 6 char. name of fg 180 ?> = fpass
Enter new 6 char. name of fg 181 ?> = ftrip0
Enter new 6 char. name of fg 182 ?> = ftrip1
Enter new 6 char. name of fg 183 ?> = ftrip2
Enter new 6 char. name of fg 184 ?> = ftrip3
Enter new 6 char. name of fg 185 ?> = ftrip4
Enter new 6 char. name of fg 186 ?> = fspar0
Enter new 6 char. name of fg 187 ?> = fspar1
Enter new 6 char. name of fg 188 ?> = fspar2
Enter new 6 char. name of fg 189 ?> = fspar3
Enter new 6 char. name of fg 190 ?> = fspar4
Enter new 6 char. name of fg 191 ?> = fspar5
MUPF?>  = add 92 in
Enter new 6 char. name of in 0 ?> = bitval
Enter new 6 char. name of in 1 ?> = iloop
```

```
Enter new 6 char. name of in 2 ?> = index
Enter new 6 char. name of in 3 ?> = remain
Enter new 6 char. name of in 4 ?> = trip
Enter new 6 char. name of in 5 ?> = nadi
Enter new 6 char. name of in 6 ?> = blkval
Enter new 6 char. name of in 7 ?> = sort
Enter new 6 char. name of in 8 ?> = bval
Enter new 6 char. name of in 9 ?> = hldstp
Enter new 6 char. name of in 10 ?> = errn
Enter new 6 char. name of in 11 ?> = errl
Enter new 6 char. name of in 12 ?> = int0
Enter new 6 char. name of in 13 ?> = int1
Enter new 6 char. name of in 14 ?> = int2
Enter new 6 char. name of in 15 ?> = int3
Enter new 6 char. name of in 16 ?> = int4
Enter new 6 char. name of in 17 ?> = int5
Enter new 6 char. name of in 18 ?> = int6
Enter new 6 char. name of in 19 ?> = int7
Enter new 6 char. name of in 20 ?> = int8
Enter new 6 char. name of in 21 ?> = int9
Enter new 6 char. name of in 22 ?> = int10
Enter new 6 char. name of in 23 ?> = int11
Enter new 6 char. name of in 24 ?> = int12
Enter new 6 char. name of in 25 ?> = int13
Enter new 6 char. name of in 26 ?> = int14
Enter new 6 char. name of in 27 ?> = int15
Enter new 6 char. name of in 28 ?> = int16
Enter new 6 char. name of in 29 ?> = int17
Enter new 6 char. name of in 30 ?> = int18
Enter new 6 char. name of in 31 ?> = int19
Enter new 6 char. name of in 32 ?> = int20
Enter new 6 char. name of in 33 ?> = int21
Enter new 6 char. name of in 34 ?> = int22
Enter new 6 char. name of in 35 ?> = int23
Enter new 6 char. name of in 36 ?> = int24
Enter new 6 char. name of in 37 ?> = int25
Enter new 6 char. name of in 38 ?> = int26
Enter new 6 char. name of in 39 ?> = int27
Enter new 6 char. name of in 40 ?> = int28
Enter new 6 char. name of in 41 ?> = int29
Enter new 6 char. name of in 42 ?> = int30
Enter new 6 char. name of in 43 ?> = int31
Enter new 6 char. name of in 44 ?> = int32
Enter new 6 char. name of in 45 ?> = int33
Enter new 6 char. name of in 46 ?> = int34
Enter new 6 char. name of in 47 ?> = int35
Enter new 6 char. name of in 48 ?> = int36
Enter new 6 char. name of in 49 ?> = int37
Enter new 6 char. name of in 50 ?> = int38
Enter new 6 char. name of in 51 ?> = int39
```

```
Enter new 6 char. name of in 52 ?> = int40
Enter new 6 char. name of in 53 ?> = int41
Enter new 6 char. name of in 54 ?> = int42
Enter new 6 char. name of in 55 ?> = int43
Enter new 6 char. name of in 56 ?> = int44
Enter new 6 char. name of in 57 ?> = int45
Enter new 6 char. name of in 58 ?> = int46
Enter new 6 char. name of in 59 ?> = int47
Enter new 6 char. name of in 60 ?> = int48
Enter new 6 char. name of in 61 ?> = int49
Enter new 6 char. name of in 62 ?> = int50
Enter new 6 char. name of in 63 ?> = int51
Enter new 6 char. name of in 64 ?> = int52
Enter new 6 char. name of in 65 ?> = int53
Enter new 6 char. name of in 66 ?> = int54
Enter new 6 char. name of in 67 ?> = int55
Enter new 6 char. name of in 68 ?> = int56
Enter new 6 char. name of in 69 ?> = int57
Enter new 6 char. name of in 70 ?> = int58
Enter new 6 char. name of in 71 ?> = int59
Enter new 6 char. name of in 72 ?> = int60
Enter new 6 char. name of in 73 ?> = int61
Enter new 6 char. name of in 74 ?> = int62
Enter new 6 char. name of in 75 ?> = int63
Enter new 6 char. name of in 76 ?> = int64
Enter new 6 char. name of in 77 ?> = int65
Enter new 6 char. name of in 78 ?> = int66
Enter new 6 char. name of in 79 ?> = int67
Enter new 6 char. name of in 80 ?> = int68
Enter new 6 char. name of in 81 ?> = int69
Enter new 6 char. name of in 82 ?> = int70
Enter new 6 char. name of in 83 ?> = int71
Enter new 6 char. name of in 84 ?> = int72
Enter new 6 char. name of in 85 ?> = int73
Enter new 6 char. name of in 86 ?> = int74
Enter new 6 char. name of in 87 ?> = int75
Enter new 6 char. name of in 88 ?> = int76
Enter new 6 char. name of in 89 ?> = int77
Enter new 6 char. name of in 90 ?> = int78
Enter new 6 char. name of in 91 ?> = int79
MUPF?>  = add 30 fl
Enter new 6 char. name of fl 0 ?> = float0
Enter new 6 char. name of fl 1 ?> = float1
Enter new 6 char. name of fl 2 ?> = float2
Enter new 6 char. name of fl 3 ?> = float3
Enter new 6 char. name of fl 4 ?> = float4
Enter new 6 char. name of fl 5 ?> = float5
Enter new 6 char. name of fl 6 ?> = float6
Enter new 6 char. name of fl 7 ?> = float7
Enter new 6 char. name of fl 8 ?> = float8
```

```
Enter new 6 char. name of fl 9 ?> = float9
Enter new 6 char. name of fl 10 ?> = ai0
Enter new 6 char. name of fl 11 ?> = ai1
Enter new 6 char. name of fl 12 ?> = ai2
Enter new 6 char. name of fl 13 ?> = ai3
Enter new 6 char. name of fl 14 ?> = ai4
Enter new 6 char. name of fl 15 ?> = ai5
Enter new 6 char. name of fl 16 ?> = ai6
Enter new 6 char. name of fl 17 ?> = ai7
Enter new 6 char. name of fl 18 ?> = ai8
Enter new 6 char. name of fl 19 ?> = ai9
Enter new 6 char. name of fl 20 ?> = ai10
Enter new 6 char. name of fl 21 ?> = ai11
Enter new 6 char. name of fl 22 ?> = ai12
Enter new 6 char. name of fl 23 ?> = ai13
Enter new 6 char. name of fl 24 ?> = ai14
Enter new 6 char. name of fl 25 ?> = ai15
Enter new 6 char. name of fl 26 ?> = ai16
Enter new 6 char. name of fl 27 ?> = ai17
Enter new 6 char. name of fl 28 ?> = ai18
Enter new 6 char. name of fl 29 ?> = ai19
MUPF?>  = add 6 pt
Enter new 6 char. name of pt 0 ?> = pepn
Enter new 6 char. name of pt 1 ?> = pdin
Enter new 6 char. name of pt 2 ?> = pdev
Enter new 6 char. name of pt 3 ?> = pai
Enter new 6 char. name of pt 4 ?> = palarm
Enter new 6 char. name of pt 5 ?> = pspar0
MUPF?>  = add 38 sg
Enter new 6 char. name of sg 0 ?> = lsg0
Enter max string length for sg 0  ?> = 30
Enter new 6 char. name of sg 1 ?> = lsg1
Enter max string length for sg 1  ?> = 30
Enter new 6 char. name of sg 2 ?> = lsg2
Enter max string length for sg 2  ?> = 30
Enter new 6 char. name of sg 3 ?> = lsg3
Enter max string length for sg 3  ?> = 30
Enter new 6 char. name of sg 4 ?> = lsg4
Enter max string length for sg 4  ?> = 30
Enter new 6 char. name of sg 5 ?> = lsg5
Enter max string length for sg 5  ?> = 30
Enter new 6 char. name of sg 6 ?> = lsg6
Enter max string length for sg 6  ?> = 30
Enter new 6 char. name of sg 7 ?> = lsg7
Enter max string length for sg 7  ?> = 30
Enter new 6 char. name of sg 8 ?> = unit0
Enter max string length for sg 8  ?> = 30
Enter new 6 char. name of sg.9 ?> = unit1
Enter max string length for sg 9  ?> = 30
Enter new 6 char. name of sg 10 ?> = unit2
```

```
Enter max string length for sg 10  ?> = 30
Enter new 6 char. name of sg 11 ?> = unit3
Enter max string length for sg 11  ?> = 30
Enter new 6 char. name of sg 12 ?> = unit4
Enter max string length for sg 12  ?> = 30
Enter new 6 char. name of sg 13 ?> = unit5
Enter max string length for sg 13  ?> = 30
Enter new 6 char. name of sg 14 ?> = unit6
Enter max string length for sg 14  ?> = 30
Enter new 6 char. name of sg 15 ?> = unit7
Enter max string length for sg 15  ?> = 30
Enter new 6 char. name of sg 16 ?> = unit8
Enter max string length for sg 16  ?> = 30
Enter new 6 char. name of sg 17 ?> = unit9
Enter max string length for sg 17  ?> = 30
Enter new 6 char. name of sg 18 ?> = unit10
Enter max string length for sg 18  ?> = 30
Enter new 6 char. name of sg 19 ?> = unit11
Enter max string length for sg 19  ?> = 30
Enter new 6 char. name of sg 20 ?> = unit12
Enter max string length for sg 20  ?> = 30
Enter new 6 char. name of sg 21 ?> = unit13
Enter max string length for sg 21  ?> = 30
Enter new 6 char. name of sg 22 ?> = unit14
Enter max string length for sg 22  ?> = 30
Enter new 6 char. name of sg 23 ?> = unit15
Enter max string length for sg 23  ?> = 30
Enter new 6 char. name of sg 24 ?> = unit16
Enter max string length for sg 24  ?> = 30
Enter new 6 char. name of sg 25 ?> = unit17
Enter max string length for sg 25  ?> = 30
Enter new 6 char. name of sg 26 ?> = unit18
Enter max string length for sg 26  ?> = 30
Enter new 6 char. name of sg 27 ?> = unit19
Enter max string length for sg 27  ?> = 30
Enter new 6 char. name of sg 28 ?> = unit20
Enter max string length for sg 28  ?> = 30
Enter new 6 char. name of sg 29 ?> = unit21
Enter max string length for sg 29  ?> = 30
Enter new 6 char. name of sg 30 ?> = unit22
Enter max string length for sg 30  ?> = 30
Enter new 6 char. name of sg 31 ?> = unit23
Enter max string length for sg 31  ?> = 30
Enter new 6 char. name of sg 32 ?> = unit24
Enter max string length for sg 32  ?> = 30
Enter new 6 char. name of sg 33 ?> = unit25
Enter max string length for sg 33  ?> = 30
Enter new 6 char. name of sg 34 ?> = unit26
Enter max string length for sg 34  ?> = 30
Enter new 6 char. name of sg 35 ?> = unit27
```

```
Enter max string length for sg 35  ?> = 30
Enter new 6 char. name of sg 36 ?> = unit28
Enter max string length for sg 36  ?> = 30
Enter new 6 char. name of sg 37 ?> = unit29
Enter max string length for sg 37  ?> = 30
MUPF?>
 end of file encountered
UPF?>
 end of file encountered
```

APPENDIX E - SEQUENCE AND BATCH LANGUAGE (SABL)

224BATCH

Sequence Instruction Compiler Listing 15:31:59 24-Feb-95 Page 1

```
;       2 4 4 B A T C H . S E Q                                 VERSION 2
;
;
;       Function:       Sequence program for Lamp function,
;                       message display on Graphic number 340,
;                       and printer messages.
;
;       Written By      : R. N. Kersey
;       Date            : 05-31-92
;       Facility        : 242-A Evaporator Bldg.
;       Machine         : VAX 4104-A
;       Op. System      : VAX/VMS 5.5-2AH
;       D/3 Version     : 6.3-2 BASELINE
;       Compiler        : SIC
;       Build command   : @[320,20]244BATCH
;
;       UNIT = 1, SKDKEY = 19
;
;       Assoc. UPF:     [300,1]_SHOTRO.___      SHORTO.LST
;       Assoc. RPF:     [300,1]_FARM.___        FARM.LST
;       SDB:            [300,1]PCMOASDB.___     PCMOASDB.CMD
;
;       POINTER   SDB VAR.      FUNCTION
;       -------   ---------     ----------------------------------------
;       PSTAT     INTn          Current EPN alarm state and is integer tied to
;                               graphic flashing string of value:
;       .         (n=0-100)             0 = previously not alarmed
;       .                               1 = previously alarmed, not ack
;       .                               2 = previously alarmed, is ack
;       .                               3 = alarm is inhibited
;       .                               4 = input is off scan
;       .                               5 = input is bad
;       .         LEAKn         Special INT variable for multiple white alarms on an EPN
;       PDIN      'EPN'         ---
;       PTYPE     TYPEn         alarm type 0 = none, 1 = white, 2 = yellow,
;                                  3 = yellow and yellow, 4 = red,
;                                  6 = yellow and red
;       PREV      FRVn          Reverse acting (not fail safe) flag (1=set)
;       PSKID     SKIDn         Used for analog alarms
;       PPASS     FPASn         Turn lamp off when alarm is ACKnowledged
;       PALARM    FALMn         Analog alarm flag
;       ---       FNING         Unused error flag
;
;  MODIFICATION LOG:
;
;       09-JUN-93       Added comments and baselined for        M. Teats
;                       D/3 version 3.3.  CCN# 93JUN01-01.
;
;       15-DEC-94       Revised header to indicate the new VMS          J. Berger
;                       system. Rebase line document. Removed
;                       obsolite Select() statements.
;
;  END MOD LOG
;
UNIT    A244
```

```
 56   0000            NOP
 57   0001            STAR = 20                       ; Start message
 58   0002            SKDKEY =19                      ; SKID key
 59   0003            ADI = 0                         ; Alarm Destination Index = 0
 60   0004            FSNO = 0                        ; No alarm yet
 61   0005            TIMEROFF STMR0                  ; Alarm timer off
 62   0006            TIMERCLR STMR0                  ; Alarm timer clear
 63   0007            STRNG0 = " "                    ; Blank 30 Character string
 64   0008            ERRN = 0                        ; Set error number to zero
 65   0009            ERRL = 0                        ; Set error line to zero
 66   0010            ASSIGN PDEV = '244ALARM' ; Alarm device
 67   0011            LET FTRIP4 = 0                  ; FTRIP4 = 1 when audible horn is to be turned on
 68   0012            GETUNIT TRIP                    ; Which sequence unit is program running in
 69   0013            IF TRIP != 1 THEN               ; Abort if running in wrong seq. unit
 70                       BEGIN
 71   0014                FSTART = 1
 72   0015                ILOOP = STAR + 1
 73   0016                GOSUB ACKK
 74   0017                FSTART = 1
 75   0018                PRINT ADI," >>>>>  ATTEMPT TO RUN 244BATCH IN WRONG UNIT"
 76   0019                GOTO TERM
 77                       END
 78            ;
 79            ;       B E G I N   I N I T I A L I Z I N G
 80            ;
 81   0020            SSNG0 = "INIT"
 82   0021    INIT:   LET FABORT = 0                  ; If FABORT = 1 then abort program
 83   0022            LET FSTART = 0                  ; Start Flag = 0
 84   0023            LAMP ADI,SKDKEY,0               ; Turn off the annunciator key
 85   0024            LET LSG0 = " "                  ; Clear the string variables
 86   0025            LET LSG1 = " "
 87   0026            LET LSG2 = " "
 88   0027            LET LSG3 = " "
 89   0028            LET LSG4 = " "
 90   0029            PDEV:DEV_TMP0 = 1                           ; Set alarm device to non-alarm state
 91   0030            IF AND(PDEV:DEV_ALMCT,16) THEN              ; If alarm device is inhibited, remove the inhibit
 92   0031              PDEV:DEV_ALMCT=XOR(PDEV:DEV_ALMCT,16)
 93   0032            IF AND(PDEV:DEV_ALMCT,32) THEN              ; If alarm device is acknowledged, clear it
 94   0033               PDEV:DEV_ALMCT = XOR(PDEV:DEV_ALMCT,32)
 95            ;
 96            ;       First, set all flags and integers:
 97            ;
 98            ; EPN           INT     TYPE    FRV
 99            ; ---------     ---     ----    ---
100            ; YS244A1IL     0       2       0
101            ; YS-P244A1     1       0       -
102            ; YS244A2IL     2       2       0
103            ; YS-P244A2     3       0       -
104            ; YS-P244A3     4       0       -
105            ; YS-A-244A     5       0       -
106            ; ZS-MOV-2F     6       0       -
107            ; ZS-MOV2DR     7       0       -
108            ; ZS-244A11     8       0       -
109            ; ZS-244A12     9       0       -
110            ; ZS-244A13     10      0       -
```

```
111           ; ZS-244A14      11      0       -
112           ; PS-244A11      12      0       -
113           ; PS-244A12      13      0       -
114           ; PS-244A13      14      0       -
115           ; PSH-SN215      15      2       0
116           ; PSH-SN216      16      2       0
117           ; LDS-PP 17      2       0
118           ; RSH-SP         18      4       0
119           ; PSH-FP 19      1       1
120           ;
121    0034            ASSIGN PDIN = 'YS244A1IL'
122    0035            ASSIGN PSTAT = INTO
123    0036            ASSIGN PTYPE = TYPEO
124    0037            ASSIGN PREV = FRVO
125    0038            ASSIGN PSKID = SKIDO
126    0039            ASSIGN PPASS = FPASO
127           ;
128    0040            FOR ILOOP = 0 TO 31
129    0041            LET PPASS = 0
130    0042            LET PREV = 0
131    0043            LET PTYPE = 0
132    0044            LET PSTAT = 0
133    0045            LET PSKID = 0
134    0046            IF ILOOP < 20 THEN                              ; For DINs
135                     BEGIN
136    0047             IF (AND(PDIN:DIN_DFLG,3)) THEN                 ; Check for alarms (must be red or yellow)
137                       BEGIN
138    0048               LET PTYPE = 2                                ; Default to yellow for DINs
139    0050               IF AND(PDIN:DIN_DFLG,1) THEN LET PREV = 1    ; Reverse acting
140    0052               IF AND(PDIN:DIN_DFLG,4) THEN LET PTYPE = 4   ; Set to critical alarm
141                       END
142    0053             STEP PDIN BY 1                                 ; Point to next DIN
143                     END
144    0055            IF ILOOP > 30 THEN LET PTYPE = 1
145    0056    SKP:    STEP PSTAT BY 1
146    0057            STEP PTYPE BY 1
147    0058            STEP PREV BY 1
148    0059            STEP PPASS BY 1
149    0060            STEP PSKID BY 1
150    0061            NEXT ILOOP
151           ;
152    0062            LET TYPE0 = 2           ; YS244A1IL
153    0063            LET TYPE2 = 2           ; YS244A2IL
154    0064            LET TYPE15 = 0          ; This EPN now alarms in Unit XYZFRM on Graphic 321
155    0065            LET TYPE16 = 0          ; This EPN now alarms in Unit XYZFRM on Graphic 321
156    0066            LET TYPE17 = 2          ; LDS-PP
157    0067            LET TYPE18 = 4          ; RSH-SP
158    0068            LET TYPE19 = 1          ; PSH-FP
159    0069            LET TYPE24 = 2          ; WFI-244SP HI
160    0070            LET TYPE26 = 6          ; LI-244TK LOLO, LO, HI, HIHI
161    0071            LET TYPE28 = 6          ; WFI-244TK LOLO, LO, HI, HIHI
162           ;
163    0072            LET FRV15 = 0           ; This EPN now alarms in Unit XYZFRM on Graphic 321
164    0073            LET FRV16 = 0           ; This EPN now alarms in Unit XYZFRM on Graphic 321
165    0074            LET FRV17 = 0           ; LDS-PP
```

```
166  0075          LET FRV18 = 0            ; RSH-SP
167  0076          LET FRV19 = 1            ; PSH-FP
168          ;
169  0077          LET AI0 = 200.0  ; TI-244-TK TEMP HI
170          ;
171          ;     E N D   I N I T I A L I Z I N G
172          ;
173          ;     S E N D   P R O G R A M   S T A R T   M E S S A G E
174          ;
175  0078          PTYPE = 0
176  0079          ILOOP = STAR
177  0080          FSTART = 1
178  0081          GOSUB ACKK
179  0082          ON ERROR GOSUB ERRSUB
180          ;
181          ;     B E G I N   P R O G R A M   L O O P
182          ;
183  0083  AGAIN: MARK STEP "WTUNTL",0,0
184          ;     WAITUNTIL SELECT() ¦ FABORT      ; Wait until micro is selected or program aborted
185          ;     IF (FABORT) THEN GOTO TERM       ; Terminate program if FABORT flag is set
186  0084          MARK STEP "RNNING",0,0
187  0085          WAIT 1
188  0086          IF (FSN0) THEN                   ; If alarm flag set
189  0087             IF STMR0 > 60 THEN            ; If alarm flag active for 1 minute
190                                                ; Reset timer and turn off flag
191  0088                BEGIN
192  0089                TIMEROFF STMR0
193  0090                TIMERCLR STMR0
194  0091                LET FSN0 = 0
195  0092                LET ERRN = 0
196                      LET ERRL = 0
197                      END
198          ;
199          ;     If 244ALARM $T1 is alarmed and it's ALMCT indicates alarms are acknowledged:
200          ;
201  0093          IF (PDEV:DEV_TMP0 = 2 ) & AND(PDEV:DEV_ALMCT,32) THEN
202                   BEGIN
203  0094             PDEV:DEV_TMP0 = 1             ; Put device in acknowledge state
204  0095             LET FTRIP4 = 1                ; Set for Unit alarm acknowledgement, system to graphic
205  0096             LET ACK = 1                   ; Set for Unit alarm acknowledgement, graphic to system
206  0097             CRIT
207  0098             LAMP ADI,SKDKEY,1
208                   END
209          ;
210          ;     BEGIN DINs:
211          ;
212  0099          ASSIGN PDIN = 'YS244A11L'
213  0100          ASSIGN PTYPE = TYPE0
214  0101          ASSIGN PREV = FRV0
215  0102          ASSIGN PSTAT = INT0
216  0103          ASSIGN PPASS = FPAS0
217  0104          CURENT = 0                       ; Used to set LAMP status
218  0105          LAST = 0
219  0106          NXT = 0
220  0107          LMPVAR = 0
221  0108          BITVAL = 0
```

```
221             ;
222     0109    LOP:    FOR ILOOP = 0 TO 19
223             ;       IF !SELECT() THEN GOTO AGAIN
224     0110            IF ( PTYPE > 0 ) THEN                                   ; If some type of alarm
225                        BEGIN                                                ; be1
226     0112               IF AND(PDIN:DIN_ALMCT,16) THEN LET PSTAT = 3         ; If EPN inhibited
227     0114               IF AND(PDIN:DIN_ALMCT,4) THEN LET PSTAT = 5                  ; If EPN is bad input
228     0116               IF !AND(PDIN:DIN_FLG,8) THEN LET PSTAT = 4                   ; If EPN is disabled
229     0117               IF ( PSTAT < 3 ) THEN                                ; If not inhibited
230                           BEGIN                                             ; be2
231     0118                  IF PSTAT = 1 THEN
232                              BEGIN                                          ; be3
233     0119                     IF PTYPE != 1 THEN
234     0121                        IF AND(PDIN:DIN_ALMCT,32) THEN GOSUB ACK1
235     0123                     IF (FTRIP4) THEN GOSUB ACK1
236                              END                                            ; en3
237     0126                  IF PSTAT = 0 THEN IF(PDIN:DIN_VAL = PREV) THEN GOSUB ALM
238     0127                  IF (PSTAT) THEN
239                                 BEGIN                                       ; be4
240     0129                        IF PDIN:DIN_VAL != PREV THEN GOSUB CLR
241     0130                        NXT = PSTAT
242     0132                        IF (PPASS) & (PSTAT = 2 ) THEN NXT = 0
243     0133                        LAST = OR(LAST,NXT)
244                                 END                                         ; en4
245                           END                                               ; en2
246     0135                ELSE IF !AND(PDIN:DIN_ALMCT,16)&!AND(PDIN:DIN_ALMCT,4)&AND(PDIN:DIN_FLG,8) THEN LET PSTAT = 0
247                         END                                               ; en1
248     0137            IF ILOOP < 19 THEN STEP PDIN BY 1
249     0138            STEP PTYPE BY 1
250     0139            STEP PREV BY 1
251     0140            STEP PSTAT BY 1
252     0141            STEP PPASS BY 1
253     0142            NEXT ILOOP
254             ;
255             ;       END DINs
256             ;
257             ;       BEGIN ANALOG RED AND YELLOWS:
258             ;
259     0143            ANALOG = 0
260     0144            ASSIGN PAI = 'II-P244A1'
261     0145            ASSIGN PSKID = SKID20
262     0146            ASSIGN PTYPE = TYPE20
263     0147            ASSIGN PREV = FRV20
264     0148            FOR JLOOP = 20 TO 30
265             ;       IF !SELECT() THEN GOTO AGAIN
266     0149            IF (PTYPE) THEN                          ; If some alarm exists
267                        BEGIN                          ; be1
268     0150               BITVAL = PAI:AI_ALMCT
269     0151               IF !AND(BITVAL,16) THEN
270                           BEGIN                       ; be2
271     0152                  IF AND(BITVAL,15) THEN             ; If alarm exists and it is current
272                              BEGIN                    ; be3
273     0154                     IF AND(BITVAL,32) THEN NXT = 2   ; If alarm is acknowledged
274                              ELSE
275                               BEGIN                   ; be4
```

```
276   0155                    IF PSKID !=BITVAL THEN
277                              BEGIN                ; be5
278   0157                       IF BITVAL < 3 THEN FALMON = 1
279   0158                       LET PSKID = BITVAL
280   0159                       LET PDEV:DEV_TMPO = 2
281   0160                       NXT = 1
282   0161                       LAMP ADI,SKDKEY,2
283   0162                       CRIT
284                              END                  ; en5
285   0163                     ELSE NXT = 1
286                            END                    ; en4
287                         END                       ; en3
288                      ELSE                                ; Else there is no alarm
289                         BEGIN
290   0164                  NXT = 0
291   0165                  LET PSKID = 0
292                         END
293                      END                          ; en2
294                   ELSE                                   ; Else alarm is inhibited
295                      BEGIN
296   0166               NXT = 0
297   0167               LET PSKID = 0
298                      END
299   0168            ANALOG = OR(ANALOG,NXT)
300                   END                             ; en1
301   0169          STEP PSKID BY 1
302   0170          STEP PTYPE BY 1
303   0171          STEP PREV BY 1
304   0172          STEP PAI BY 1
305   0173          NEXT JLOOP
306         ;
307         ;       END ANALOG RED AND YELLOWS
308         ;
309         ;       BEGIN ANALOG WHITE ALARMS:
310         ;
311   0174  WHT:   WHITE = 0
312   0175         SSNGO = "RUN"
313   0176         ASSIGN PSTAT = INT31
314   0177         ASSIGN PTYPE = TYPE31
315   0178         ASSIGN PREV = FRV31
316   0179         ASSIGN PPASS = FPAS31
317   0180         ASSIGN PALARM = AIO
318   0181         ILOOP = 31
319   0182         ASSIGN PAI = 'TI-244-TK'
320   0183         GOSUB ANLG
321         ;
322         ;       END ANALOG WHITE ALARMS
323         ;
324   0184         MARK STEP "BELOW",0,0
325   0185         LET FTRIP4 = 0
326   0187         IF LAST > 3 THEN PRINT ADI," LAMP VARIABLE IS GREATER THAN 3!"
327   0188         CURENT = OR(LAST,WHITE)
328   0189         CURENT = OR(CURENT,ANALOG)
329   0191         IF AND(CURENT,1)THEN LMPVAR = 2
330   0193            ELSE IF AND(CURENT,2)THEN LMPVAR = 1
```

```
331   0194                    ELSE LMPVAR = 0
332   0195          LAMP ADI,SKDKEY,LMPVAR
333   0196          MARK STEP "BOTTOM",0,0
334   0197          NOCRIT
335         ;
336         ;       BEGIN DEVICES FOR GRAPHICS:
337         ;
338   0198          FLEAK0 = AND('P-244-A1':DEV_STAT,4)
339   0199          FLEAK1 = AND('P-244-A1':DEV_STAT,16)
340   0200          FLEAK2 = AND('P-244-A1':DEV_STAT,32)
341   0201          FLEAK3 = AND('P-244-A2':DEV_STAT,4)
342   0202          FLEAK4 = AND('P-244-A2':DEV_STAT,16)
343   0203          FLEAK5 = AND('P-244-A2':DEV_STAT,32)
344   0204          FLEAK6 = AND('A-244-A':DEV_STAT,4)
345   0205          FLEAK7 = AND('A-244-A':DEV_STAT,16)
346   0206          FRV39 = AND('A-244-A':DEV_STAT,32)
347   0207          FRV36 = AND('P-244-A3':DEV_STAT,4)
348   0208          FRV37 = AND('P-244-A3':DEV_STAT,16)
349   0209          FRV38 = AND('P-244-A3':DEV_STAT,32)
350                 ;
351                 ;       FOR INTERLOCK GRAPHIC:
352                 ;
353   0210          FSN1 = AND('P-244-A1':DEV_TMP1,128)
354   0211          FSN2 = AND('P-244-A1':DEV_TMP1,256)
355   0212          FSN3 = AND('P-244-A1':DEV_TMP1,512)
356   0213          FSN4 = AND('P-244-A1':DEV_TMP1,1024)
357   0214          FSN5 = AND('P-244-A1':DEV_TMP1,2048)
358   0215          FSN6 = AND('P-244-A1':DEV_TMP1,4096)
359   0216          FSN7 = AND('P-244-A1':DEV_TMP1,8192)
360   0217          FSL1 = AND('P-244-A1':DEV_TMP1,32768)
361   0218          FSL2 = 'YS-P244A1':DIN_VAL
362   0219          FSL3 = 'YS-P244A2':DIN_VAL
363   0220          FSL4 = YS244A1IL:DIN_VAL
364         ;
365         ;       END DEVICES FOR GRAPHICS
366         ;
367   0221          GOTO AGAIN
368         ;
369         ;       E N D   P R O G R A M   L O O P
370         ;
371         ;       BEGIN ANALOG WHITE ALARM SUBROUTINE:
372         ;
373         ; ANLG: IF !SELECT() THEN GOTO AGAIN
374   0222  ANLG:   IF PTYPE = 1 THEN                          ; If some alarm
375                   BEGIN                                   ; be0
376   0223            BITVAL = PAI:AI_ALMCT
377   0224            IF !(AND(BITVAL,16)) THEN               ; If not inhibited
378                     BEGIN                                 ; be1
379   0225              IF (PREV) THEN                        ; If rev acting, LOLO, or LO
380                       BEGIN                               ; be2
381   0226                IF (PSTAT = 0) THEN                 ; If not previously alarmed
382                         BEGIN                             ; be3
383   0228                    IF PAI:AI_INVL <= PALARM THEN GOSUB ALM
384                         END                               ; en3
385                       ELSE
```

```
386                              BEGIN                              ; be4
387    0229                      IF PSTAT = 1 THEN                          ; If previously alarmed
388                                 BEGIN                           ; be5
389    0231                         IF (FTRIP4) THEN GOSUB ACK1             ; If ACK flag set, then ack it
390    0234                         IF PTYPE !=1 THEN IF AND(PAI:AI_ALMCT,32) THEN GOSUB ACK1
391                                 END                             ; en5
392    0236                      IF PAI:AI_INVL >= PALARM + PAI:AI_DEAD THEN GOSUB CLR
393                              END                                ; en4
394                           END                                   ; en2
395                        ELSE                                             ; If fail-safe, HIHI, or HI
396                           BEGIN                                 ; be2.1
397    0237                   IF PSTAT = 0  THEN
398                              BEGIN
399    0239                      IF PAI:AI_INVL >= PALARM THEN GOSUB ALM
400                              END
401                           ELSE
402                              BEGIN
403    0240                      IF PSTAT = 1 THEN
404                                 BEGIN
405    0242                         IF (FTRIP4) THEN GOSUB ACK1
406    0245                         IF PTYPE != 1 THEN IF AND(PAI:AI_ALMCT,32) THEN GOSUB ACK1
407                                 END
408    0247                      IF PAI:AI_INVL <= PALARM - PAI:AI_DEAD THEN GOSUB CLR
409                              END
410                           END                                   ; en2.1
411    0249                IF PSTAT = 3 THEN PSTAT = 0                      ; If INHA, then there is no alarm
412    0251                IF (PPASS) & (PSTAT = 2 ) THEN NXT = 0
413    0252                   ELSE NXT = PSTAT
414                        END
415                        ELSE
416                          BEGIN
417    0253                  LET PSTAT = 3
418    0254                  NXT = 0
419                          END
420    0255             WHITE = OR(WHITE,NXT)
421                      END
422    0256              ELSE LET PSTAT = 0
423    0257              STEP PALARM BY 1
424    0258              STEP PSTAT BY 1
425    0259              STEP PTYPE BY 1
426    0260              STEP PREV BY 1
427    0261              STEP PPASS BY 1
428    0262              ILOOP = ILOOP + 1
429    0263              RETURN
430             ;
431             ;        END ANALOG WHITE ALARM SUBROUTINE
432             ;
433             ;        BEGIN CLEAR, ALARM, ACKNOWLEDGE, MESSAGE, ERROR, AND TERMINATE SUBROUTINES:
434             ;
435    0264     CLR:     LET PSTAT = 0                                              ; Int set to zero
436                      ;
437                      ; If not white then unack:
438                      ;
439    0267              IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
440    0268              LET FCLEAR = 1                                             ; Clear flag
```

```
441  0269          GOSUB ACKK                                          ; Print message
442  0270          RETURN
443           ;
444  0271    ALM:  LET PSTAT = 1                                       ; Int set to one
445                 ;
446                 ; If not white then unack:
447                 ;
448  0274           IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
449                 ;
450                 ; Turn audible alarm on if white or yellow:
451                 ;
452  0276           IF AND(PTYPE,3) THEN FALMON = 1
453  0277           LET PDEV:DEV_TMP0 = 2
454  0278           CRIT
455  0279           LAMP ADI,SKDKEY,2                                   ; Blink the LAMP
456  0280           GOSUB ACKK                                          ; Print message
457  0281           RETURN
458           ;
459  0282    ACK1: LET PSTAT = 2                                       ; Acknowledge alarm
460  0283          LET FACK = 1                                        ; Set ACK flag for C program
461           ;
462           ;       ALARM TYPER STRINGS:
463           ;
464  0285    ACKK:  IF ILOOP = 0 THEN STRNG0 = "244A-P1 HRDWR INTRLK  "
465  0287           IF ILOOP = 2 THEN STRNG0 = "244A-P2 HRDWR INTRLK  "
466  0289           IF ILOOP = 17 THEN STRNG0 = "244A PUMP PIT LEAK    "
467  0291           IF ILOOP = 18 THEN STRNG0 = "244A SER PIT RADN HI  "
468  0293           IF ILOOP = 19 THEN STRNG0 = "244A FLSH PIT PRES HI "
469  0295           IF ILOOP = 20 THEN STRNG0 = "244BATCH PGM STARTING "
470  0297           IF ILOOP = 21 THEN STRNG0 = "244BATCH WRONG UNIT   "
471  0299           IF ILOOP = 22 THEN STRNG0 = "244BATCH TERMINATING  "
472  0301           IF ILOOP = 31 THEN STRNG0 = "244A LIQUID TEMP HI   "
473           ;
474           ; If acknowledging alarm, then:
475           ;
476  0302    AMSG:  IF (FACK) THEN
477                    BEGIN
478                    ;
479                    ; If white, print it is acknowledged and reset the flag:
480                    ;
481  0304              IF PTYPE = 1 THEN PRINT ADI,"%p ALARM ACKNOWLEDGE",STRNG0
482  0305              LET FACK = 0
483  0306              GOTO NXT
484                    END
485           ;
486           ; If CLEAR flag is set, print it is clear and reset the flag:
487           ;
488  0307           IF (FCLEAR) THEN
489                    BEGIN
490  0308              PRINT ADI,"%p ALARM CLEAR",STRNG0
491  0309              LET FCLEAR = 0
492  0310              GOTO NXT
493                    END
494           ;       INO = TIMSTR(SSNG0)
495           ;       IF INO != 8 THEN SSNG0 = "TIMEFAIL"
```

```
496             ;       INO = APPEND(STRNG1,STRNG0,SSNG0)
497             ;       IF INO < 0 THEN STRNG1 = "APPEND FAILED!!"
498             ;
499             ; If white alarm, print:
500             ;
501     0312            IF PTYPE = 1 THEN PRINT ADI,"WHITE ALARM %p",STRNG0
502             ;
503             ; If STARTUP, then print startup message:
504             ;
505     0313    NXT:    IF FSTART = 1 THEN
506                        BEGIN
507     0314               LET FSTART = 0
508     0315               PRINT ADI,"%p",STRNG0
509                        END
510     0316            RETURN
511             ;
512             ; If ERROR - print message, start Btimer, and set flag:
513             ;
514     0317    ERRSUB: IF ERRN != ERRNUM() THEN
515                        BEGIN
516     0318               TIMERON STMR0
517     0319               TIMERCLR STMR0
518     0320               LET FSN0 = 1
519     0321               LET ERRN = ERRNUM()
520     0322               LET ERRL = ERRLIN()
521     0323               PRINT ADI,"ERROR # %i AT LINE %i",errn,errl
522                        END
523     0324            ERSCLR
524     0325            GOTO AGAIN
525             ;
526             ; If FABORT = 1, then abort program:
527             ;
528             ; TERM:  LET FABORT = 0
529     0326    TERM:   LET STRNG0 = "244BATCH TERMINATED    "
530     0327            LET ILOOP = STAR + 2
531     0328            LET FSTART = 1
532     0329            LET SSNG0 = "TERM"
533     0330            GOSUB ACKK
534                     END

      Total Errors Detected      0     Total Warnings      0
          Sequence Program Size = 6272 Bytes, 392 Pages
               21 Forward Reference(s)
```

244HVAC

```
;       2 4 4 H V A C . S E Q                                   VERSION 2
;
;       Function:       Sequence program for Lamp function,
;                       message display on Graphic number 341,
;                       and printer messages.
;
;       Written By      : R. N. Kersey
;       Date            : 05-31-92
;       Facility        : 242-A Evaporator Bldg.
;       Machine         : VAX 4104-A
;       Op. System      : VAX/VMS 5.5-2AH
;       D/3 Version     : 6.3-2 BASELINE
;       Compiler        : SIC
;       Build command   : @[320,20]244HVAC
;
;       UNIT = 2, SKDKEY = 18
;
;       Assoc. UPF:     [300,1]_SHORTO.___      SHORTO.LST
;       Assoc. RPF:     [300,1]_FARM.___        FARM.LST
;       SDB:            [300,1]PCMOASDB.___     PCMOASDB.CMD
;
;       POINTER   SDB VAR.      FUNCTION
;       -------   ---------     ----------------------------------------
;       PSTAT     INTn          Current EPN alarm state is an integer tied to
;                               graphic flashing string of value:
;       .         (n=0-100)             0 = previously not alarmed
;       .                               1 = previously alarmed, not ack
;       .                               2 = previously alarmed, is ack
;       .                               3 = alarm is inhibited
;       .                               4 = input is off scan
;       .                               5 = input is bad
;       .         LEAKn         Special INT variable for multiple white alarms on an EPN
;       PDIN      'EPN'         ---
;       PTYPE     TYPEn         alarm type 0 = none, 1 = white, 2 = yellow,
;                                  3 = yellow and yellow, 4 = red,
;                                  6 = yellow and red
;       PREV      FRVn          Reverse acting (not fail safe) flag (1=set)
;       PSKID     SKIDn         Used for analog alarms
;       PPASS     FPASn         Turn lamp off when alarm is ACKnowledged
;       PALARM    FALMn         Analog alarm flag
;       ---       FNING         Unused error flag
;
; MODIFICATION LOG:
;
;       09-JUN-93       Added comments and baselined for        M. Teats
;                       D/3 version 3.3.  CCN# 93JUN01-01.
;
;       15-DEC-94       Revised header to indicate the new VMS      J. Berger
;                       system. Rebase line document. Removed
;                       obsolite Select() statements.
;
; END MOD LOG
;
UNIT    HVC244
;
```

```
 56   0000            STAR = 8                        ; Start message
 57   0001            SKDKEY = 18                     ; SKID Key
 58   0002            ADI = 0                         ; Alarm destination index
 59   0003            FSNO = 0                        ; No alarm yet
 60   0004            TIMEROFF STMRO                  ; Alarm timer off
 61   0005            TIMERCLR STMRO                  ; Alarm timer clear
 62   0006            STRNGO = " "                    ; Blank 30 character string
 63   0007            ERRN = 0                        ; Set error number to zero
 64   0008            ERRL = 0                        ; Set error line to zero
 65   0009            ASSIGN PDEV = 'HVAC244'         ; Alarm device
 66   0010            FTRIP4 = 0                      ; FTRIPP4 = 1 when audible horn is to be turned on
 67   0011            GETUNIT TRIP                    ; Which sequence unit is program running in
 68   0012            IF TRIP != 2 THEN               ; Abort if running in wrong seq. unit
 69                       BEGIN
 70   0013                FSTART = 1
 71   0014                ILOOP = STAR + 1
 72   0015                GOSUB ACKK
 73   0016                FSTART = 1
 74   0017                PRINT ADI," >>>>>  ATTEMPT TO RUN 244BATCH IN WRONG UNIT"
 75   0018                GOTO TERM
 76                       END
 77            ;       IF !SELECT() THEN GOTO AGAIN
 78            ;
 79            ;       B E G I N  I N I T I A L I Z I N G
 80            ;
 81   0019            SSNGO = "INIT"
 82   0020    INIT:   FABORT = 0                      ; If FABORT = 1 then abort program
 83   0021            FSTART = 0                      ; Program start flag = 0
 84   0022            LAMP ADI,SKDKEY,0               ; Turn off the annunciator key
 85   0023            LSGO = " "                      ; Clear the string variables
 86   0024            LSG1 = " "
 87   0025            LSG2 = " "
 88   0026            LSG3 = " "
 89   0027            LSG4 = " "
 90   0028            PDEV:DEV_TMPO = 1                           ; Set alarm device to non-alarm state
 91   0029            IF AND(PDEV:DEV_ALMCT,16) THEN              ; If alarm device is inhibited, remove the inhibit
 92   0030              PDEV:DEV_ALMCT=XOR(PDEV:DEV_ALMCT,16)
 93   0031            IF AND(PDEV:DEV_ALMCT,32) THEN              ; If alarm device is acknowledged, clear it
 94   0032               PDEV:DEV_ALMCT = XOR(PDEV:DEV_ALMCT,32)
 95            ;
 96            ;       First, set all flags and integers:
 97            ;
 98            ; EPN           TYPE    INT     FRV     LIMIT          DESCRIPTOR
 99            ; ---------     ----    ---     ---     -----   ------------------------------
100            ; PDSL-EF-2     2       0       1               244-A HEPA DP FAN FAIL EXH RADN
101            ; PDSL-EF-1     2       1       0               244-A HEPA DELTA P HIGH
102            ; RSH-A         4       2       0               244-A ANNULUS RADN HIGH
103            ; RSH-EXH-1     4       3       0               244-A EXH STAK RADN HIGH
104            ; QXS-EXH-1     2       4       0               244-A STACK MON SYS FAIL
105            ;
106            ; RI-244ANN     4       5       -               244-A LIFT STA ANNULUS RADN
107            ; FI-244EXH     0       6       -               244-A EXHAUST STACK FLOW
108            ; RI-244EXH     4       7       -               244-A LIFT STA EXH STACK RADN
109            ;
110            ; FI-244EXH     1       8       1       100 LO  244-A EXHAUST STACK FLOW
```

```
111          ; FI-244EXH      1       9       0       200 HI  244-A EXHAUST STACK FLOW
112          ;
113    0033          ASSIGN PDIN = 'PDSL-EF-2'                    ; Assign pointer to first DIN
114    0034          ASSIGN PSTAT = INTO
115    0035          ASSIGN PTYPE = TYPEO
116    0036          ASSIGN PREV = FRVO
117    0037          ASSIGN PPASS = FPASO
118    0038          ASSIGN PSKID = SKIDO
119    0039          ASSIGN PALARM = FDISO
120          ;
121    0040          FOR ILOOP = 0 TO 9
122    0041          PPASS = 0
123    0042          PREV = 0
124    0043          PTYPE = 2
125    0044          PSTAT = 0
126    0045          PSKID = 0
127    0046          IF ILOOP < 5 THEN
128                   BEGIN
129    0047           IF (AND(PDIN:DIN_DFLG,3)) THEN              ; Check for alarms (must be red or yellow)
130                     BEGIN
131    0048             PTYPE = 2                                 ; Default to yellow
132    0050             IF AND(PDIN:DIN_DFLG,1) THEN PREV = 1     ; Reverse acting
133    0052             IF AND(PDIN:DIN_DFLG,4) THEN PTYPE = 4    ; Set to critical alarm
134                     END
135    0054           IF ILOOP < 4 THEN STEP PDIN BY 1            ; Point to next DIN
136                   END
137    0056          IF ILOOP > 7 THEN PTYPE = 1
138    0057    SKP:  STEP PSTAT BY 1
139    0058          STEP PTYPE BY 1
140    0059          STEP PREV BY 1
141    0060          STEP PPASS BY 1
142    0061          STEP PSKID BY 1
143    0062          STEP PALARM BY 1
144    0063          NEXT ILOOP
145          ;
146    0064          FRV0 = 1              ; PDSL-EF-2
147    0065          FRV1 = 0              ; PSDH-EF-1
148    0066          FRV2 = 0              ; RSH-A
149    0067          FRV3 = 0              ; RSH-EXH-1
150    0068          FRV4 = 0              ; QXS-EXH-1
151    0069          FRV8 = 1              ; FI-244EXH LO
152          ;
153    0070          TYPE5 = 4             ; RI-244ANN
154    0071          TYPE6 = 0             ; RI-244EXH
155    0072          TYPE7 = 4             ; RI-244EXH
156          ;
157    0073          AI0 = 100.0           ; FI-244EXH LO
158    0074          AI1 = 200.0           ; FI-244EXH HI
159    0075          ANALOG = 0
160          ;
161          ;       E N D   I N I T I A L I Z I N G
162          ;
163          ;       S E N D   P R O G R A M   S T A R T   M E S S A G E
164          ;
165          ;
```

```
166  0076            ILOOP = STAR
167  0077            FSTART = 1
168  0078            GOSUB ACKK
169  0079            ON ERROR GOSUB ERRSUB
170          ;
171          ;       B E G I N   P R O G R A M   L O O P
172          ;
173  0080    AGAIN:  MARK STEP "AGAIN",0,0
174  0081            SSNG0 = "WAIT"
175          ;       WAITUNTIL SELECT() | FABORT      ; Wait until micro is selected or program aborted
176          ;       IF (FABORT) THEN GOTO TERM       ; Terminate program if FABORT flag is set
177  0082            WAIT 1
178  0083            IF (FSN0) THEN                   ; If alarm flag set
179  0084               IF STMR0 > 60 THEN            ; If alarm flag active for 1 minute
180                      BEGIN                      ; Reset timer and turn off flag
181  0085                  TIMEROFF STMR0
182  0086                  TIMERCLR STMR0
183  0087                  FSN0 = 0
184  0088                  ERRN = 0
185  0089                  ERRL = 0
186                      END
187          ;
188          ;       If HVAC244 $T1 is alarmed and it's ALMCT indicates alarms are acknowledged:
189          ;
190  0090            IF (PDEV:DEV_TMP0 = 2) & AND(PDEV:DEV_ALMCT,32) THEN
191                  BEGIN
192  0091               PDEV:DEV_TMP0 = 1             ; Put device in ack state
193  0092               FTRIP4 = 1                    ; Set for unit alarm ack, system to graphic
194  0093               ACK = 1                       ; Set for unit alarm ack, graphic to system
195  0094               CRIT
196  0095               LAMP ADI,SKDKEY,1
197                     END
198          ;
199          ;       BEGIN DINs:
200          ;
201  0096            ASSIGN PDIN = 'PDSL-EF-2'
202  0097            ASSIGN PTYPE = TYPE0
203  0098            ASSIGN PREV = FRV0
204  0099            ASSIGN PSTAT = INT0
205  0100            ASSIGN PPASS = FPAS0
206  0101            CURENT = 0                       ; Used to set LAMP status
207  0102            LAST = 0
208  0103            ANALOG = 0
209  0104            NXT = 0
210  0105            LMPVAR = 0
211  0106            BITVAL = 0
212          ;
213  0107    LOP:    FOR ILOOP = 0 TO 4
214  0108            IF ( PTYPE > 0 ) THEN                          ; If some type of alarm
215                     BEGIN                                  ;be1
216  0110               IF AND(PDIN:DIN_ALMCT,16) THEN PSTAT = 3    ; If EPN inhibited
217  0112               IF AND(PDIN:DIN_ALMCT,4) THEN PSTAT = 5     ; If EPN is bad input
218  0114               IF !AND(PDIN:DIN_FLG,8) THEN PSTAT = 4      ; If EPN is disabled
219  0115               IF ( PSTAT < 3 ) THEN                  ; If not inhibited
220                          BEGIN                           ;be2
```

```
221  0116                  IF PSTAT = 1 THEN
222                           BEGIN                          ;be3
223  0118                     IF AND(PDIN:DIN_ALMCT,32) THEN GOSUB ACK1
224  0120                     IF (FTRIP4) THEN GOSUB ACK1
225                           END                            ;en3
226  0123                  IF PSTAT = 0 THEN IF(PDIN:DIN_VAL = PREV) THEN GOSUB ALM
227  0124                  IF(PSTAT) THEN
228                             BEGIN                        ;be4
229  0126                       IF PDIN:DIN_VAL != PREV THEN GOSUB CLR
230  0127                       NXT = PSTAT
231  0129                       IF (PPASS) & (PSTAT = 2) THEN NXT = 0
232  0130                       LAST = OR(LAST,NXT)
233                             END                          ;en4
234                        END                               ;en2
235  0132               ELSE IF !AND(PDIN:DIN_ALMCT,16)&!AND(PDIN:DIN_ALMCT,4)&AND(PDIN:DIN_FLG,8)THEN PSTAT = 0
236                     END                                  ;en1
237  0134            IF ILOOP < 4 THEN STEP PDIN BY 1
238  0135            STEP PTYPE BY 1
239  0136            STEP PREV BY 1
240  0137            STEP PSTAT BY 1
241  0138            STEP PPASS BY 1
242  0139            NEXT ILOOP
243                  ;
244                  ;       END DINs:
245                  ;
246                  ;       BEGIN ANALOG RED AND YELLOWS:
247                  ;
248  0140            ASSIGN PSKID = SKID5
249  0141            ASSIGN PTYPE = TYPE5
250  0142            ASSIGN PREV = FRV5
251  0143            ASSIGN PAI = 'RI-244ANN'
252  0144            FOR JLOOP = 5 TO 7
253  0145            IF(PTYPE) THEN                          ; If some alarm exists
254                     BEGIN                          ;be1
255  0146               BITVAL = PAI:AI_ALMCT
256  0147               IF !AND(BITVAL,16) THEN
257                        BEGIN                       ;be2
258  0148                  IF AND(BITVAL,15) THEN            ; If alarm exists and alarm is current
259                           BEGIN                    ;be3
260  0150                     IF AND(BITVAL,32) THEN NXT = 2  ; If alarm is ack
261                           ELSE
262                              BEGIN                 ;be4
263  0151                        IF PSKID != BITVAL THEN
264                                 BEGIN              ;be5
265  0152                           PSKID = BITVAL
266  0153                           PDEV:DEV_TMPO = 2
267  0155                           IF BITVAL < 3 THEN FALMON = 1
268  0156                           NXT = 1
269  0157                           LAMP ADI,SKDKEY,2
270  0158                           CRIT
271                                 END                ;en5
272  0159                        ELSE NXT = 1
273                              END                   ;en4
274                        END                         ;en3
275                     ELSE                                ; Else there is no alarm
```

```
276                              BEGIN
277     0160                     NXT = 0
278     0161                     PSKID = 0
279                              END
280                           END                          ;en2
281                        ELSE                                  ; Else alarm is inhibited
282                           BEGIN
283     0162                  NXT = 0
284     0163                  PSKID = 0
285                           END
286     0164                  ANALOG = OR(ANALOG,NXT)
287                           END                          ;en1
288     0165               STEP PSKID BY 1
289     0166               STEP PTYPE BY 1
290     0167               STEP PREV BY 1
291     0169               IF JLOOP < 3 THEN STEP PAI BY 1
292     0170               NEXT JLOOP
293             ;
294             ;          END ANALOG RED AND YELLOWS
295             ;
296             ;          BEGIN ANALOG WHITE ALARMS:
297             ;
298     0171    WHT:       WHITE = 0
299     0172               SSNG0 = "RUN"
300     0173               ASSIGN PALARM = AI0
301     0174               ASSIGN PSTAT = INT8
302     0175               ASSIGN PTYPE = TYPE8
303     0176               ASSIGN PREV = FRV8
304     0177               ASSIGN PPASS = FPAS8
305     0178               ILOOP = 8
306     0179               ASSIGN PAI = 'FI-244EXH'
307     0180               GOSUB ANLG
308     0181               GOSUB ANLG
309     0182               CURENT = OR(CURENT,ANALOG)
310     0183               MARK STEP "BELOW",0,0
311                        ;
312                        ;  Use LEAK0 with string packet 78 in graphic:
313                        ;
314     0185               IF ((INT9)&INT9<3) THEN LEAK0 = INT9 + 4
315     0187                  ELSE IF((INT8) & (INT8 < 3)) THEN LEAK0 = INT8 + 2
316     0189                       ELSE IF INT9 = 3 ¦ INT8 = 3 THEN LEAK0 = 9
317     0190                            ELSE LEAK0 = 0
318             ;
319             ;          END ANALOG WHITE ALARMS
320             ;
321     0191               FTRIP4 = 0
322     0193               IF CURENT > 3 THEN PRINT ADI," LAMP VARIABLE IS GREATER THAN 3!"
323     0194               CURENT = OR(LAST,WHITE)
324     0195               CURENT = OR(CURENT,ANALOG)
325     0197               IF AND(CURENT,1)THEN LMPVAR = 2
326     0199                  ELSE IF AND(CURENT,2)THEN LMPVAR = 1
327     0200                          ELSE LMPVAR = 0
328     0201               LAMP ADI,SKDKEY,LMPVAR
329     0202               MARK STEP "BOTTOM",0,0
330     0203               NOCRIT
```

```
331  0204          GOTO AGAIN
332          ;
333          ;     E N D   P R O G R A M   L O O P
334          ;
335          ;     BEGIN ANALOG WHITE ALARM SUBROUTINE:
336          ;
337  0205    ANLG: IF PTYPE = 1 THEN                              ; If some alarm
338                   BEGIN                                  ;be0
339  0206             BITVAL = PAI:AI_ALMCT
340  0207             IF !(AND(BITVAL,16) & (PTYPE)) THEN        ; If not inhibited and alarm enabled
341                     BEGIN                                  ;be1
342  0209               IF PSTAT = 3 THEN PSTAT = 0
343  0210               IF (PREV) THEN                         ; If rev acting, LOLO, or LO
344                        BEGIN                               ;be2
345  0211                  IF !(PSTAT) THEN                    ; If not previously alarmed
346                           BEGIN                            ;be3
347  0213                     IF PAI:AI_INVL <= PALARM THEN GOSUB ALM
348                           END                              ;en3
349                     ELSE
350                        BEGIN                               ;be4
351  0214                  IF PSTAT = 1 THEN                   ; If previously alarmed
352                           BEGIN                            ;be5
353  0216                     IF (FTRIP4) THEN GOSUB ACK1      ; If ACK flag set, then ack it
354  0219                     IF PTYPE !=1 THEN IF AND(PAI:AI_ALMCT,32) THEN GOSUB ACK1
355                           END                              ;en5
356  0221                  IF PAI:AI_INVL >= PALARM + PAI:AI_DEAD THEN GOSUB CLR
357                        END                                 ;en4
358                     END                                    ;en2
359                  ELSE                                      ; If fail-safe, HIHI, or HI
360                     BEGIN                                  ;be2.1
361  0222               IF PSTAT = 0  THEN
362                        BEGIN
363  0224                  IF PAI:AI_INVL >= PALARM THEN GOSUB ALM
364                        END
365                     ELSE
366                        BEGIN
367  0225                  IF PSTAT = 1 THEN
368                           BEGIN
369  0227                     IF (FTRIP4) THEN GOSUB ACK1
370  0230                     IF PTYPE != 1 THEN IF AND(PAI:AI_ALMCT,32) THEN GOSUB ACK1
371                           END
372  0232                  IF PAI:AI_INVL <= PALARM - PAI:AI_DEAD THEN GOSUB CLR
373                        END
374                     END                                    ;en2.1
375  0234            IF PSTAT = 3 THEN PSTAT = 0              ; If INHA then there is no alarm
376  0236            IF (PPASS) & (PSTAT = 2 ) THEN NXT = 0
377  0237               ELSE NXT = PSTAT
378  0238            NXT = PSTAT
379                  END
380                  ELSE
381                    BEGIN
382  0239              PSTAT = 3
383  0240              NXT = 0
384                    END
385  0241          WHITE = OR(WHITE,NXT)
```

```
386                     END
387     0242            ELSE PSTAT = 0
388     0243            STEP PALARM BY 1
389     0244            STEP PSTAT BY 1
390     0245            STEP PTYPE BY 1
391     0246            STEP PREV BY 1
392     0247            STEP PPASS BY 1
393     0248            ILOOP = ILOOP + 1
394     0249            RETURN
395             ;
396             ;       END ANALOG WHITE ALARM SUBROUTINE
397             ;
398             ;       BEGIN CLEAR, ALARM, ACKNOWLEDGE, MESSAGE, ERROR, AND TERMINATE SUBROUTINES:
399             ;
400     0250    CLR:    PSTAT = 0                                               ; Int set to zero
401     0253            IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
402     0254            FCLEAR = 1                                              ; Clear flag
403     0256            IF ILOOP = 9 THEN INT8 = 0                              ; Reset the other white alarm
404     0257            GOSUB ACKK                                              ; Print message
405     0258            RETURN
406             ;
407     0259    ALM:    PSTAT = 1                                               ; Int set to one
408                     ;
409                     ; If not white then unack:
410                     ;
411     0262            IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
412                     ;
413                     ; Turn audible alarm on if white or yellow:
414                     ;
415     0264            IF AND(PTYPE,3) THEN FALMON = 1
416     0265            PDEV:DEV_TMP0 = 2
417     0266            CRIT
418     0267            LAMP ADI,SKDKEY,2                                       ; Blink the LAMP
419     0268            GOSUB ACKK                                              ; Print message
420     0269            RETURN
421             ;
422     0270    ACK1:   PSTAT = 2                                               ; Ack the alarm
423     0271            FACK = 1                                                ; Set ACK flag for C program
424             ;
425             ;       ALARM TYPER STRINGS:
426             ;
427     0273    ACKK:   IF ILOOP = 0 THEN STRNG0 = "DP FAN/FAIL EXH RADN  "
428     0275           IF ILOOP = 1 THEN STRNG0 = "244-A HEPA DLTA P HI  "
429     0277            IF ILOOP = 2 THEN STRNG0 = "244-A ANN EXH RAD HI  "
430     0279            IF ILOOP = 3 THEN STRNG0 = "244-A EXH STK RAD HI  "
431     0281            IF ILOOP = 4 THEN STRNG0 = "244-A STK MN SYS FAIL "
432     0283            IF ILOOP = 8 THEN STRNG0 = "244-A EXH STK FLO LO  "
433     0285            IF ILOOP = 9 THEN STRNG0 = "244-A EXH STK FLO HI  "
434     0287            IF ILOOP = 20 THEN STRNG0 = "START PGM 244HVAC.SEQ "
435     0289            IF ILOOP = 21 THEN STRNG0 = "WRONG UNIT 244HVAC.SEQ"
436     0291            IF ILOOP = 22 THEN STRNG0 = "244HVAC TERMINATED   "
437             ;
438             ; If acknowledging the alarm, then:
439             ;
440     0292    AMSG:   IF (FACK) THEN
```

```
441                         BEGIN
442                         ;
443                         ; If white, print it is ack and reset the flag:
444                         ;
445     0293                PRINT ADI,"%p ALARM ACKNOWLEDGE",STRNG0
446     0294                FACK = 0
447     0295                GOTO NXT
448                         END
449             ;
450             ; If CLEAR flag is set, print it is clear and reset the flag:
451             ;
452     0296            IF (FCLEAR) THEN
453                         BEGIN
454     0297                PRINT ADI,"%p ALARM CLEAR",STRNG0
455     0298                FCLEAR = 0
456     0299                GOTO NXT
457                         END
458             ;       REMAIN= TIMSTR(SSNG0)
459             ;       IF REMAIN != 8 THEN SSNG0 = "TIMEFAIL"
460             ;       REMAIN = APPEND(STRNG1,STRNG0,SSNG0)
461             ;       IF REMAIN < 0 THEN STRNG1 = "APPEND FAILED!!"
462             ;
463             ; If white alarm, print:
464             ;
465     0301            IF PTYPE = 1 THEN PRINT ADI,"WHITE ALARM %p",STRNG1
466             ;
467             ; If STARTUP, then print startup message:
468             ;
469     0302    NXT:    IF FSTART = 1 THEN
470                         BEGIN
471     0303                FSTART = 0
472     0304                PRINT ADI,"%p",STRNG0
473                         END
474     0305            RETURN
475             ;
476             ; If ERROR - print message, start Btimer, and set flag:
477             ;
478     0307    ERRSUB: IF(ERRNUM() = -187) THEN ERSRET
479                     ELSE
480                       BEGIN
481     0308              IF ERRN != ERRNUM() THEN
482                               BEGIN
483     0309                      TIMERON STMR0
484     0310                      TIMERCLR STMR0
485     0311                      FSN0 = 1
486     0312                      ERRN = ERRNUM()
487     0313                      ERRL = ERRLIN()
488     0314                      PRINT ADI,"ERROR # %i AT LINE %i",ERRN,ERRL
489     0315                      ERSCLR
490                               END
491                       END
492     0316            GOTO AGAIN
493             ;
494             ; If FABORT = 1, then abort the program:
495             ;
```

```
496             ; TERM:  FABORT = 0
497     0317    TERM:    STRNGO = "244BATCH TERMINATED    "
498     0318             ILOOP = STAR + 2
499     0319             FSTART = 1
500     0320             SSNGO = "TERM"
501     0321             GOSUB ACKK
502                      END

       Total Errors Detected      0     Total Warnings     0
            Sequence Program Size = 6048 Bytes, 378 Pages
                  22 Forward Reference(s)
```

WSTXFR

```
1           ;       W S T X F R . S E Q                              VERSION 2
2           ;
3           ;
4           ;       Function:       Sequence program for Lamp function,
5           ;                       message display on Graphic number 344,
6           ;                       and printer messages.
7           ;
8           ;       Written By      : R. N. Kersey
9           ;       Date            : 05-31-92
10          ;       Facility        : 242-A Evaporator Bldg.
11          ;       Machine         : VAX 4104-A
12          ;       Op. System      : VAX/VMS 5.5-2AH
13          ;       D/3 Version     : 6.3-2 BASELINE
14          ;       Compiler        : SIC
15          ;       Build command   : @[320,20]WSTXFR
16          ;
17          ;       UNIT = 3, SKDKEY = 16
18          ;
19          ;       Assoc. UPF:     [300,1]_SHORTO.___      SHORTO.LST
20          ;       Assoc. RPF:     [300,1]_FARM.___        FARM.LST
21          ;       SDB:            [300,1]PCMOASDB.___     PCMOASDB.CMD
22          ;
23          ;       POINTER   SDB VAR.      FUNCTION
24          ;       -------   ---------     ----------------------------------------
25          ;       PSTAT     INTn          Current EPN alarm state is an integer tied to
26          ;                               graphic flashing string of value:
27          ;       .         (n=0-100)             0 = previously not alarmed
28          ;       .                               1 = previously alarmed, not ack
29          ;       .                               2 = previously alarmed, is ack
30          ;       .                               3 = alarm is inhibited
31          ;       .                               4 = input is off scan
32          ;       .                               5 = input is bad
33          ;       .         LEAKn         Special INT variable for multiple white alarms on an EPN
34          ;       PDIN      'EPN'         ---
35          ;       PTYPE     TYPEn         alarm type 0 = none, 1 = white, 2 = yellow,
36          ;                                 3 = yellow and yellow, 4 = red,
37          ;                                 6 = yellow and red
38          ;       PREV      FRVn          Reverse acting (not fail safe) flag (1=set)
39          ;       PSKID     SKIDn         Used for analog alarms
40          ;       PPASS     FPASn         Turn lamp off when alarm is ACKnowledged
41          ;       PALARM    FALMn         Analog alarm flag
42          ;       ---       FNING         Unused error flag
43          ;
44          ;  MODIFICATION LOG:
45          ;
46          ;       04-AUG-92       Set FRV9 and FRV10 to 0 for Fail Safe          M. Teats
47          ;                       per ECN 186678.
48          ;
49          ;       30-MAR-93       Modified for redundancy.              M. Teats
50          ;
51          ;       09-JUN-93       Added comments, CCN# 93JUN01-01.               M. Teats
52          ;
53          ;-mct1+  13-JUL-93      Add LDS216A40 PCM1 DIN section, and CLR2,      M. Teats
54          ;                       ALM2, ACK2 subroutines per CCN 93JUL13-01.
55          ;                       Baselined for D/3 version 3.3.
```

```
56          ;
57          ;       15-DEC-94       Revised header to indicate the new VMS          J. Berger
58          ;                       system. Rebase line document. Removed
59          ;                       obsolite Select() statements.
60          ;
61          ;  END MOD LOG
62          ;
63          UNIT    WASTE                           ; Is in PCM0
64                  COMMON WHOLE1                   ; Is in PCM1, -mct1+
65          ;
66                  LDS1 == 'LDS216A40'             ;-mct1+
67    0000          NOP
68    0001          STAR = 20                       ; Start message
69    0002          SKDKEY = 16                     ; SKID Key
70    0003          ADI = 0                         ; Alarm Destination Index
71    0004          FSN0 = 0                        ; No alarm yet
72    0005          TIMEROFF STMR0                  ; Alarm timer off
73    0006          TIMERCLR STMR0                  ; Alarm timer clear
74    0007          STRNG0 = " "                    ; Blank 30 character string
75    0008          ERRN = 0                        ; Set error number to zero
76    0009          ERRL = 0                        ; Set error line to zero
77    0010          ASSIGN PDEV = 'WSTXFR'          ; Alarm Device
78    0011          FTRIP4 = 0                      ; FTRIP4 = 1 when audible horn is to be turned on
79    0012          GETUNIT TRIP                    ; Which sequence unit is program running in
80    0013          IF TRIP != 3 THEN               ; Abort if running in the wrong seq. unit
81                     BEGIN
82    0014             FSTART = 1
83    0015             ILOOP = STAR + 1
84    0016             FSTART = 1
85    0017             GOSUB ACKK
86    0018             PRINT ADI," >>>>>  ATTEMPT TO RUN WSTXFR IN WRONG UNIT"
87    0019             GOTO TERM
88                     END
89          ;
90          ;       B E G I N   I N I T I A L I Z I N G
91          ;
92    0020          SSNG0 = "INIT"
93    0021    INIT: LET FABORT = 0                  ; If FABORT = 1 then abort program
94    0022          LET FSTART = 0                  ; Start Flag = 0
95    0023          LAMP ADI,SKDKEY,0               ; Turn off the annunciator key
96    0024          LET LSG0 = " "                  ; Clear the string variables
97    0025          LET LSG1 = " "
98    0026          LET LSG2 = " "
99    0027          LET LSG3 = " "
100   0028          LET LSG4 = " "
101   0029          PDEV:DEV_TMP0 = 1                             ; Set alarm device to non-alarm state
102   0030          IF AND(PDEV:DEV_ALMCT,16) THEN                ; If alarm device is inhibited, remove the inhibit
103   0031            PDEV:DEV_ALMCT=XOR(PDEV:DEV_ALMCT,16)
104   0032          IF AND(PDEV:DEV_ALMCT,32) THEN                ; If alarm device is acknowledged, clear it
105   0033             PDEV:DEV_ALMCT = XOR(PDEV:DEV_ALMCT,32)
106         ;
107         ;       First, set all flags and integers:
108         ;
109         ;       EPN             INT     TYPE    FRV
110         ;       ---------       ---     ----    ---
```

```
111             ;       LDS-AR151       0       2       0
112             ;       LDS-ER151       1       2       0
113             ;       PSHSNV229       2       2       0
114             ;       LDS-ER152       3       2       0
115             ;       ZSER15212       4       0       -
116             ;       ZSER15211       5       0       -
117             ;       PSER15212       6       0       -
118             ;       PSER15211       7       0       -
119             ;       LDS-ER153       8       2       0
120             ;       PSHPSN232       9       2       0
121             ;       PSHPSN234       10      2       0
122             ;       ZSER15312       11      0       -
123             ;       ZSER15311       12      0       -
124             ;       PSER15312       13      0       -
125             ;       PSER15311       14      0       -
126             ;       LDS-AX155       15      2       0
127             ;       LDS-DR713       16      2       0
128             ;       LDS-DR717       17      2       0
129             ;       LDS216A40       18      2       0       ; IPN 243 in PCM1, -mct1+
130             ;
131     0034            ASSIGN PDIN = 'LDS-AR151'               ; Assign pointer to first DIN
132     0035            ASSIGN PSTAT = INTO
133     0036            ASSIGN PTYPE = TYPE0
134     0037            ASSIGN PREV = FRV0
135     0038            ASSIGN PPASS = FPAS0
136             ;
137     0039            FOR ILOOP = 0 TO 17
138     0040            LET PPASS = 0
139     0041            LET PREV = 0
140     0042            LET PTYPE = 0
141     0043            LET PSTAT = 0
142     0044            IF (AND(PDIN:DIN_DFLG,3)) THEN          ; Check for alarms (must be red or yellow)
143                        BEGIN
144     0045               LET PTYPE = 2                        ; Default to yellow
145     0047               IF AND(PDIN:DIN_DFLG,1) THEN LET PREV = 1    ; Reverse acting
146     0049               IF AND(PDIN:DIN_DFLG,4) THEN LET PTYPE = 4   ; Set to critical alarm
147                        END
148     0051            IF ILOOP < 17 THEN STEP PDIN BY 1       ; Point to next DIN
149     0052    SKP:    STEP PSTAT BY 1
150     0053            STEP PTYPE BY 1
151     0054            STEP PREV BY 1
152     0055            STEP PPASS BY 1
153     0056            NEXT ILOOP
154             ;
155     0057            LET TYPE4 = 0           ; ZSER15212
156     0058            LET TYPE5 = 0           ; ZSER15211
157     0059            LET TYPE6 = 0           ; PSER15212
158     0060            LET TYPE7 = 0           ; PSER15211
159     0061            LET TYPE11 = 0          ; ZSER15312
160     0062            LET TYPE12 = 0          ; ZSER15311
161     0063            LET TYPE13 = 0          ; PSER15312
162     0064            LET TYPE14 = 0          ; PSER15311
163     0065            LET TYPE18 = 2          ; LDS216A40, -mct1+
164             ;
165     0066            LET FRV0 = 0            ; LDS-AR151
```

```
166  0067          LET FRV1 = 0            ; LDS-ER151
167  0068          LET FRV2 = 0            ; PSHSNV229
168  0069          LET FRV3 = 0            ; LDS-ER152
169  0070          LET FRV8 = 0            ; LDS-ER153
170  0071          LET FRV9 = 0            ; PSHPSN232
171  0072          LET FRV10 = 0           ; PSHPSN234
172  0073          LET FRV15 = 0           ; LDS-AX155
173  0074          LET FRV16 = 0           ; LDS-DR713
174  0075          LET FRV17 = 0           ; LDS-DR717
175  0076          LET FRV18 = 0           ; LDS216A40, -mct1+
176          ;
177  0077          LET INT18 = 0           ; LDS216A40, -mct1+
178  0078          LET FPAS18 = 0          ; LDS216A40, -mct1+
179          ;
180          ;     E N D   I N I T I A L I Z I N G
181          ;
182          ;     S E N D   P R O G R A M   S T A R T   M E S S A G E
183          ;
184  0079          ANALOG = 0
185  0080          ILOOP = STAR
186  0081          FSTART = 1
187  0082          GOSUB ACKK
188  0083          ON ERROR GOSUB ERRSUB
189          ;
190          ;     B E G I N   P R O G R A M   L O O P
191          ;
192  0084    AGAIN:MARK STEP "WTUNTL",0,0
193          ;     WAITUNTIL SELECT() ¦ FABORT     ; Wait until micro is selected or program aborted
194          ;     IF (FABORT) THEN GOTO TERM      ; Terminate program if FABORT flag is set
195  0085          MARK STEP "RNNING",0,0
196  0086          WAIT 1
197  0087          IF (FSNO) THEN                  ; If alarm flag set
198  0088             IF STMR0 > 60 THEN           ; If alarm flag active for 1 minute
199                         BEGIN                ; Reset timer and turn off flag
200  0089                   TIMEROFF STMR0
201  0090                   TIMERCLR STMR0
202  0091                   LET FSNO = 0
203  0092                   LET ERRN = 0
204  0093                   LET ERRL = 0
205                         END
206          ;
207          ;     If WSTXFR $T1 is alarmed and it's ALMCT indicates alarms are acknowledged:
208          ;
209  0094          IF (PDEV:DEV_TMP0 = 2 ) & AND(PDEV:DEV_ALMCT,32) THEN
210                   BEGIN
211  0095             PDEV:DEV_TMP0 = 1            ; Put device in acknowledge state
212  0096             LET FTRIP4 = 1               ; Set for unit alarm acknowledgement, system to graphic
213  0097             LET ACK = 1                  ; Set for unit alarm acknowledgement, graphic to system
214  0098             CRIT
215  0099             LAMP ADI,SKDKEY,1
216                   END
217          ;
218          ;     BEGIN DINs:
219          ;
220  0100          ASSIGN PDIN = 'LDS-AR151'
```

```
221  0101            ASSIGN PSTAT = INTO
222  0102            ASSIGN PTYPE = TYPEO
223  0103            ASSIGN PREV = FRVO
224  0104            ASSIGN PPASS = FPASO
225  0105            CURENT = 0                          ; Used to set LAMP status
226  0106            LAST = 0
227  0107            ANALOG = 0
228  0108            NXT = 0
229  0109            WHITE = 0
230  0110            LMPVAR = 0
231  0111            BITVAL = 0
232                  ;
233  0112       LOP: FOR ILOOP = 0 TO 17
234                  ;       IF !SELECT() THEN GOTO AGAIN
235  0113            IF ( PTYPE > 0 ) THEN                                   ; If some type of alarm
236                     BEGIN                                                ; be1
237  0115               IF AND(PDIN:DIN_ALMCT,16) THEN LET PSTAT = 3       ; If EPN inhibited
238  0117               IF AND(PDIN:DIN_ALMCT,4) THEN LET PSTAT = 5              ; If EPN is bad input
239  0119               IF !AND(PDIN:DIN_FLG,8) THEN LET PSTAT = 4               ; If EPN is disabled
240  0120               IF ( PSTAT < 3 ) THEN                                ; If not inhibited
241                          BEGIN                                          ; be2
242  0121                    IF PSTAT = 1 THEN
243                             BEGIN                                       ; be3
244  0122                       IF PTYPE != 1 THEN
245  0124                          IF AND(PDIN:DIN_ALMCT,32) THEN GOSUB ACK1
246  0126                       IF (FTRIP4) THEN GOSUB ACK1
247                             END                                         ; en3
248  0129                    IF PSTAT = 0 THEN IF(PDIN:DIN_VAL = PREV) THEN GOSUB ALM
249  0130                    IF (PSTAT) THEN
250                                BEGIN                                    ; be4
251  0132                          IF PDIN:DIN_VAL != PREV THEN GOSUB CLR
252  0133                          NXT = PSTAT
253  0135                          IF (PPASS) & (PSTAT = 2) THEN NXT = 0
254  0136                          LAST = OR(LAST,NXT)
255                                END                                      ; en4
256                          END                                            ; en2
257  0138                 ELSE IF !AND(PDIN:DIN_ALMCT,16)&!AND(PDIN:DIN_ALMCT,4)&AND(PDIN:DIN_FLG,8) THEN LET PSTAT = 0
258                      END                                                ; en1
259  0140            IF ILOOP < 17 THEN STEP PDIN BY 1
260  0141            STEP PSTAT BY 1
261  0142            STEP PTYPE BY 1
262  0143            STEP PREV BY 1
263  0144            STEP PPASS BY 1
264  0145            NEXT ILOOP
265                  ;
266                  ;       Begin PCM1 DIN (LDS216A40) -mct1+:
267                  ;
268  0146            ILOOP = 18
269                  ;       IF !SELECT() THEN GOTO AGAIN
270  0147            IF ( TYPE18 > 0 ) THEN                                  ; If some type of alarm
271                     BEGIN                                                ; be1
272  0149               IF AND('LDS1':DIN_ALMCT,16) THEN LET INT18 = 3     ; If EPN inhibited
273  0151                  IF (ERRNUM() = -187) THEN GOTO WHT               ; Skip LDS216A40 if PCM1 micros down
274  0153               IF AND('LDS1':DIN_ALMCT,4) THEN LET INT18 = 5      ; If EPN is bad input
275  0155               IF !AND('LDS1':DIN_FLG,8) THEN LET INT18 = 4       ; If EPN is disabled
```

```
276  0156              IF ( INT18 < 3 ) THEN                          ; If not inhibited
277                         BEGIN                                     ; be2
278  0157                   IF INT18 = 1 THEN
279                            BEGIN                                  ; be3
280  0158                      IF TYPE18 != 1 THEN
281  0160                         IF AND('LDS1':DIN_ALMCT,32) THEN GOSUB ACK2
282  0162                      IF (FTRIP4) THEN GOSUB ACK2
283                            END                                    ; en3
284  0165                   IF INT18 = 0 THEN IF('LDS1':DIN_VAL = FRV18) THEN GOSUB ALM2
285  0166                   IF (INT18) THEN
286                               BEGIN                               ; be4
287  0168                         IF 'LDS1':DIN_VAL != FRV18 THEN GOSUB CLR2
288  0169                         NXT = INT18
289  0171                         IF (FPAS18) & (INT18 = 2) THEN NXT = 0
290  0172                         LAST = OR(LAST,NXT)
291                               END                                 ; en4
292                         END                                       ; en2
293  0174                ELSE IF !AND('LDS1':DIN_ALMCT,16)&!AND('LDS1':DIN_ALMCT,4)&AND('LDS1':DIN_FLG,8) THEN LET INT18 = 0
294                      END                                          ; en1
295                 ;
296                 ;       END PCM1 DIN
297          ;
298          ;       END DINs
299          ;
300  0175    WHT:    WHITE = 0
301  0176            SSNG0 = "RUN"
302  0177            MARK STEP "BELOW",0,0
303  0178            LET FTRIP4 = 0
304  0180            IF LAST > 3 THEN PRINT ADI," LAMP VARIABLE IS GREATER THAN 3!"
305  0181            CURENT = OR(LAST,WHITE)
306  0182            CURENT = OR(CURENT,ANALOG)
307  0184            IF AND(CURENT,1) THEN LMPVAR = 2
308  0186               ELSE IF AND(CURENT,2) THEN LMPVAR = 1
309  0187                       ELSE LMPVAR = 0
310  0188            LAMP ADI,SKDKEY,LMPVAR
311  0189            MARK STEP "BOTTOM",0,0
312  0190            NOCRIT
313  0191            GOTO AGAIN
314          ;
315          ;       E N D   P R O G R A M   L O O P
316          ;
317          ;       BEGIN CLEAR, ALARM, ACKNOWLEDGE, MESSAGE, ERROR, AND TERMINATE SUBROUTINES:
318          ;
319  0192    CLR:    LET PSTAT = 0                                          ; INT set to zero
320                  ;
321                  ; If not white then unack:
322                  ;
323  0195            IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
324  0196            LET FCLEAR = 1                                         ; Clear Flag
325  0197            GOSUB ACKK
326  0198            RETURN
327          ;
328  0199    CLR2:   LET INT18 = 0                                          ; PCM1 DIN, -mct1+
329                  ;
330                  ; If not white, then unack:
```

```
331                     ;
332     0202            IF TYPE18 != 1 THEN IF AND('LDS1':DIN_ALMCT,32)THEN 'LDS1':DIN_ALMCT = XOR('LDS1':DIN_ALMCT,32)
333     0203            LET FCLEAR = 1
334     0204            GOSUB ACKK
335     0205            RETURN
336             ;
337     0206    ALM:    LET PSTAT = 1                                   ; INT set to one
338                     ;
339                     ; If not white then unack:
340                     ;
341     0209            IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
342                     ;
343                     ; Turn audible alarm on if white or yellow:
344                     ;
345     0211            IF AND(PTYPE,3) THEN FALMON = 1
346     0212            LET PDEV:DEV_TMPO = 2
347     0213            CRIT
348     0214            LAMP ADI,SKDKEY,2                               ; Blink the LAMP
349     0215            GOSUB ACKK                                      ; Print message
350     0216            RETURN
351             ;
352     0217    ALM2:   LET INT18 = 1                                   ; PCM1 DIN, -mct1+
353                     ;
354                     ; If not white, then unack:
355                     ;
356     0220            IF TYPE18 != 1 THEN IF AND('LDS1':DIN_ALMCT,32)THEN 'LDS1':DIN_ALMCT = XOR('LDS1':DIN_ALMCT,32)
357                     ;
358                     ; Turn audible alarm on if white or yellow:
359                     ;
360     0221            FALMON = 1
361     0222            LET 'WSTXFR':DEV_TMPO = 2
362     0223            CRIT
363     0224            LAMP ADI,SKDKEY,2
364     0225            GOSUB ACKK
365     0226            RETURN
366             ;
367     0227    ACK1:   LET PSTAT = 2                                   ; Acknowledge alarm
368     0228            LET FACK = 1                                    ; Set ACK flag for C program
369             ;
370     0229    ACK2:   IF ILOOP = 18 THEN                              ; PCM1 DIN, -mct1+
371                     BEGIN
372     0230            LET INT18 = 2
373     0231            LET FACK = 1
374                     END
375             ;
376             ;       ALARM TYPER STRINGS:
377             ;
378     0233    ACKK:   IF ILOOP = 0 THEN STRNGO = "151-AR DIVISION BOX LEAK"
379     0235            IF ILOOP = 1 THEN STRNGO = "1-1-ER DIVISION BOX LEAK"
380     0237            IF ILOOP = 2 THEN STRNGO = "151-ER LINE V229 LEAK"
381     0239            IF ILOOP = 3 THEN STRNGO = "152-ER DIVISION BOX LEAK"
382     0241            IF ILOOP = 8 THEN STRNGO = "153-ER DIVISION BOX LEAK"
383     0243            IF ILOOP = 9 THEN STRNGO = "153-ER LINE V232 LEAK"
384     0245            IF ILOOP = 10 THEN STRNGO = "153-ER LINE V234 LEAK"
385     0247            IF ILOOP = 15 THEN STRNGO = "155-AX DIVISION BOX LEAK"
```

```
386  0249            IF ILOOP = 16 THEN STRNG0 = " DRAIN DR-713 LINE LEAK"
387  0251            IF ILOOP = 17 THEN STRNG0 = " DRAIN DR-717 LINE LEAK"
388  0253            IF ILOOP = 18 THEN STRNG0 = "LDS216A40 LEAK DET "        ;-mct1+
389  0255            IF ILOOP = 20 THEN STRNG0 = "WSTXFR PGM STARTING "
390  0257            IF ILOOP = 21 THEN STRNG0 = "WSTXFR WRONG UNIT   "
391  0259            IF ILOOP = 22 THEN STRNG0 = "WSTXFR TERMINATING  "
392          ;
393          ; If acknowledging alarm, then:
394          ;
395  0260    AMSG:    IF (FACK) THEN
396                      BEGIN
397                      ;
398                      ; If white, print it is acknowledged and reset the flag
399                      ;
400  0262                IF PTYPE = 1 THEN PRINT ADI,"%p ALARM ACKNOWLEDGE",STRNG0
401  0263                LET FACK = 0
402  0264                GOTO NXT
403                      END
404          ;
405          ; If CLEAR flag is set, print it is clear and reset the flag:
406          ;
407  0265            IF (FCLEAR) THEN
408                      BEGIN
409  0266                PRINT ADI,"%p ALARM CLEAR",STRNG0
410  0267                LET FCLEAR = 0
411  0268                GOTO NXT
412                      END
413          ;       INO = TIMSTR(SSNG0)
414          ;       IF INO != 8 THEN SSNG0 = "TIMEFAIL"
415          ;       INO = APPEND(STRNG1,STRNG0,SSNG0)
416          ;       IF INO < 0 THEN STRNG1 = "APPEND FAILED!!"
417          ;
418          ; If white alarm, print:
419          ;
420  0270            IF PTYPE = 1 THEN PRINT ADI,"WHITE ALARM %p",STRNG0
421          ;
422          ; If STARTUP, then print startup message:
423          ;
424  0271    NXT:     IF FSTART = 1 THEN
425                      BEGIN
426  0272                LET FSTART = 0
427  0273                PRINT ADI,"%p",STRNG0
428                      END
429  0274            RETURN
430          ;
431          ; If ERROR - print message, start timer, and set flag:
432          ;
433  0275    ERRSUB:  IF ERRN != ERRNUM() THEN
434                      BEGIN
435  0276                TIMERON STMR0
436  0277                TIMERCLR STMR0
437  0278                LET FSN0 = 1
438  0279                LET ERRN = ERRNUM()
439  0280                LET ERRL = ERRLIN()
440  0281                PRINT ADI,"ERROR # %i AT LINE %i",errn,errl
```

```
441                         END
442    0282              ERSCLR
443    0283              GOTO AGAIN
444             ;
445             ; If FABORT = 1, then abort program:
446             ;
447             ; TERM:  LET FABORT = 0
448    0284     TERM:    LET STRNG0 = "WSTXFR TERMINATED    "
449    0285              LET ILOOP = STAR + 2
450    0286              LET FSTART = 1
451    0287              LET SSNG0 = "TERM"
452    0288              GOSUB ACKK
453                       END

       Total Errors Detected      0     Total Warnings      0
            Sequence Program Size = 5600 Bytes, 350 Pages
                 19 Forward Reference(s)
```

350BATCH

```
1           ;       3 5 0 B A T C H . S E Q                                 VERSION 3
2           ;
3           ;       Function:       Sequence program for Lamp function,
4           ;                       message display on Graphic number 343,
5           ;                       and printer messages.
6           ;
7           ;       Written By      : R. N. Kersey
8           ;       Date            : 05-31-92
9           ;       Facility        : 242-A Evaporator Bldg.
10          ;       Machine         : VAX 4104-A
11          ;       Op. System      : VAX/VMS 5.5-2AH
12          ;       D/3 Version     : 6.3-2 BASILINE
13          ;       Compiler        : SIC
14          ;       Build command   : @[320,20]350BATCH
15          ;
16          ;       UNIT = 4, SKDKEY = 17
17          ;
18          ;       Assoc. UPF:     [300,1]_SHORT0.___      SHORT0.LST
19          ;       Assoc. RPF:     [300,1]_FARM.___        FARM.LST
20          ;       SDB:            [300,1]PCMOASDB.___     PCMOASDB.CMD
21          ;
22          ;       POINTER   SDB VAR.      FUNCTION
23          ;       -------   ---------     ----------------------------------------
24          ;       PSTAT     INTn          Current EPN alarm state is an integer tied to
25          ;                               graphic flashing string of value:
26          ;       .         (n=0-100)             0 = previously not alarmed
27          ;       .                               1 = previously alarmed, not ack
28          ;       .                               2 = previously alarmed, is ack
29          ;       .                               3 = alarm is inhibited
30          ;       .                               4 = input is off scan
31          ;       .                               5 = input is bad
32          ;       .         LEAKn         Special INT variable for multiple white alarms on an EPN
33          ;       PDIN      'EPN'         ---
34          ;       PTYPE     TYPEn         alarm type 0 = none, 1 = white, 2 = yellow,
35          ;                                  3 = yellow and yellow, 4 = red,
36          ;                                  6 = yellow and red
37          ;       PREV      FRVn          Reverse acting (not fail safe) flag (1=set)
38          ;       PSKID     SKIDn         Used for analog alarms
39          ;       PPASS     FPASn         Turn lamp off when alarm is ACKnowledged
40          ;       PALARM    FALMn         Analog alarm flag
41          ;       ---       FNING         Unused error flag
42          ;
43          ; MODIFICATION LOG:
44          ;
45          ;       09-JUN-93       Added comments and baselined for D/3            M. Teats
46          ;                       version 3.3.  CCN# 93JUN01-01 and
47          ;                       CCN# 93JUN01-02.
48          ;
49          ;       04-JAN-94       Changed variable FRV2=1 to FRV2=0 per       R. Persin
50          ;                       ECN 603359 and CCN 94JAN04-01
51          ;
52          ;       15-DEC-94       Revised header to indicate the new VMS          J. Berger
53          ;                       system. Rebase line document. Removed
54          ;                       obsolite Select() statements.
55          ;
```

```
 56           ;  END MOD LOG
 57           ;
 58           UNIT     A350
 59           ;
 60    0000            STAR = 13                        ; Start message
 61    0001            SKDKEY = 17                      ; SKID Key
 62    0002            ADI = 0                          ; Alarm destination index
 63    0003            FSNO = 0                         ; No alarm yet
 64    0004            TIMEROFF STMRO                   ; Alarm timer off
 65    0005            TIMERCLR STMRO                   ; Alarm timer clear
 66    0006            STRNGO = " "                     ; Blank 30 character string
 67    0007            ERRN = 0                         ; Set error number to zero
 68    0008            ERRL = 0                         ; Set error line to zero
 69    0009            ASSIGN PDEV = '350ALARM' ; Alarm device
 70    0010            FTRIP4 = 0                       ; FTRIP4 = 1 when audible horn is to be turned on
 71    0011            GETUNIT TRIP                     ; Which sequence unit is program running in
 72    0012            IF TRIP != 4 THEN                ; Abort if running in wrong seq. unit
 73                       BEGIN
 74    0013               FSTART = 1
 75    0014               ILOOP = STAR + 1
 76    0015               ASSIGN PTYPE = TYPE14
 77    0016               PTYPE = 0
 78    0017               GOSUB ACKK
 79    0018               GOTO TERM
 80                       END
 81           ;       IF !SELECT() THEN GOTO AGAIN
 82           ;
 83           ;       B E G I N   I N I T I A L I Z I N G
 84           ;
 85    0019            SSNGO = "INIT"
 86    0020    INIT:   FABORT = 0                       ; If FABORT = 1 then abort program
 87    0021            FSTART = 0                       ; Start flag = 0
 88    0022            LAMP ADI,SKDKEY,0                ; Turn off the annunciator key
 89    0023            LSGO = " "                       ; Clear the string variables
 90    0024            LSG1 = " "
 91    0025            LSG2 = " "
 92    0026            LSG3 = " "
 93    0027            LSG4 = " "
 94    0028            PDEV:DEV_TMPO = 1                          ; Set alarm device to non-alarm state
 95    0029            IF AND(PDEV:DEV_ALMCT,16) THEN             ; If alarm device is inhibited, remove the inhibit
 96    0030              PDEV:DEV_ALMCT=XOR(PDEV:DEV_ALMCT,16)
 97    0031            IF AND(PDEV:DEV_ALMCT,32) THEN             ; If alarm device is ack, clear it
 98    0032               PDEV:DEV_ALMCT = XOR(PDEV:DEV_ALMCT,32)
 99           ;
100           ;       First, set all flags and integers:
101           ;
102    0033            ASSIGN PDIN = 'LSH-350-2'                  ; Assign pointer to first DIN
103    0034            ASSIGN PSTAT = INT1
104    0035            ASSIGN PTYPE = TYPE1
105    0036            ASSIGN PREV = FRV1
106    0037            ASSIGN PSKID = SKID1
107    0038            ASSIGN PPASS = FPAS1
108    0039            ASSIGN PALARM = FDIS1
109           ;
110    0040            FOR ILOOP = 1 TO 10
```

```
111  0041            PPASS = 0
112  0042            PREV = 0
113  0043            PTYPE = 0
114  0044            PSTAT = 0
115  0045            PSKID = 0
116  0046            PALARM = 0
117  0047            IF ILOOP < 4 THEN                                ; For DINs
118                   BEGIN
119  0048             IF (AND(PDIN:DIN_DFLG,3)) THEN                  ; Check for alarms (must be red or yellow)
120                     BEGIN
121  0049               PTYPE = 2                                     ; Default to yellow
122  0051               IF AND(PDIN:DIN_DFLG,1) THEN PREV = 1         ; Reverse acting
123  0053               IF AND(PDIN:DIN_DFLG,4) THEN PTYPE = 4        ; Set to critical alarm
124                     END
125  0055             IF ILOOP < 3 THEN STEP PDIN BY 1                ; Point to next DIN
126                   END
127  0056  SKP:      STEP PSTAT BY 1
128  0057            STEP PTYPE BY 1
129  0058            STEP PREV BY 1
130  0059            STEP PPASS BY 1
131  0060            STEP PSKID BY 1
132  0061            STEP PALARM BY 1
133  0062            NEXT ILOOP
134          ;
135  0063            FRV1 = 0                ; LSH-350-2
136  0064            FRV2 = 0                ; LDS-350-2
137  0065            FRV3 = 0                ; LDS-PW481
138          ;
139  0066            TYPE7 = 6               ; WFI-350TK
140          ;
141  0067            ANALOG = 0
142          ;
143          ;       E N D   I N I T I A L I Z I N G
144          ;
145          ;       S E N D   P R O G R A M   S T A R T   M E S S A G E
146          ;
147  0068            PTYPE = 0
148  0069            FCLEAR = 0
149  0070            ILOOP = STAR
150  0071            FSTART = 1
151  0072            GOSUB ACKK
152  0073            ON ERROR GOSUB ERRSUB
153          ;
154          ;       B E G I N   P R O G R A M   L O O P
155          ;
156  0074  AGAIN:   MARK STEP "AGAIN",0,0
157  0075            SSNGO = "WAIT"
158          ;       WAITUNTIL SELECT() ¦ FABORT     ; Wait until micro is selected or program aborted
159          ;       IF (FABORT) THEN GOTO TERM      ; Terminate program if FABORT flag is set
160  0076            WAIT 1
161  0077            IF (FSNO) THEN                  ; If alarm flag set
162  0078               IF STMRO > 60 THEN           ; If alarm flag active for 1 minute
163                       BEGIN                     ; Reset timer and turn off flag
164  0079                 TIMEROFF STMRO
165  0080                 TIMERCLR STMRO
```

```
166  0081                FSNO = 0
167  0082                ERRN = 0
168  0083                ERRL = 0
169                      END
170          ;
171          ;       If 350ALARM $T1 is alarmed and it's ALMCT indicates alarms are acknowledged:
172          ;
173  0084            IF (PDEV:DEV_TMP0 = 2 ) & AND(PDEV:DEV_ALMCT,32) THEN
174                    BEGIN
175  0085              PDEV:DEV_TMP0 = 1             ; Put device in ack state
176  0086              FTRIP4 = 1                    ; Set for unit alarm ack, system to graphic
177  0087              ACK = 1                       ; Set for unit alarm ack, graphic to system
178  0088              CRIT
179  0089              LAMP AD1,SKDKEY,1
180                    END
181          ;
182          ;       BEGIN DINs:
183          ;
184  0090            ASSIGN PDIN = 'LSH-350-2'
185  0091            ASSIGN PTYPE = TYPE1
186  0092            ASSIGN PREV = FRV1
187  0093            ASSIGN PSTAT = INT1
188  0094            ASSIGN PPASS = FPAS1
189  0095            LET LAST = 0
190  0096            LET NXT = 0
191  0097            LMPVAR = 0
192  0098            BITVAL = 0
193          ;
194  0099    LOP:    FOR ILOOP = 1 TO 3
195  0100            IF ( PTYPE > 0 ) THEN                           ; If some type of alarm
196                    BEGIN                                 ;be1
197  0102              IF AND(PDIN:DIN_ALMCT,16) THEN PSTAT = 3      ; If EPN inhibited
198  0104              IF AND(PDIN:DIN_ALMCT,4) THEN PSTAT = 5       ; If EPN is bad input
199  0106              IF !AND(PDIN:DIN_FLG,8) THEN PSTAT = 4        ; If EPN is disabled
200  0107              IF ( PSTAT < 3 ) THEN                 ; If not inhibited
201                         BEGIN                            ;be2
202  0108                  IF PSTAT = 1 THEN
203                           BEGIN                          ;be3
204  0109                     IF PTYPE != 1 THEN
205  0111                        IF AND(PDIN:DIN_ALMCT,32) THEN GOSUB ACK1
206  0113                     IF (FTRIP4) THEN GOSUB ACK1
207                           END                            ;en3
208  0116                  IF PSTAT = 0 THEN IF(PDIN:DIN_VAL = PREV) THEN GOSUB ALM
209  0117                  IF (PSTAT) THEN
210                              BEGIN                       ;be4
211  0119                        IF PDIN:DIN_VAL != PREV THEN GOSUB CLR
212  0120                        NXT = PSTAT
213  0122                        IF (PPASS) & (PSTAT = 2 ) THEN NXT = 0
214  0123                        LAST = OR(LAST,NXT)
215                              END                         ;en4
216                       END                               ;en2
217  0125                ELSE IF !AND(PDIN:DIN_ALMCT,16)&!AND(PDIN:DIN_ALMCT,4)&AND(PDIN:DIN_FLG,8)THEN PSTAT = 0
218                       END                               ;en1
219  0126            STEP PDIN BY 1
220  0127            STEP PTYPE BY 1
```

```
221   0128          STEP PREV BY 1
222   0129          STEP PSTAT BY 1
223   0130          STEP PPASS BY 1
224   0131          NEXT ILOOP
225           ;
226           ;     END DINs:
227           ;
228           ;     BEGIN ANALOG RED AND YELLOWS:
229           ;
230   0132          LET ANALOG = 0
231   0133          ASSIGN PAI = 'II-350-1'
232   0134          ASSIGN PSKID = SKID4
233   0135          ASSIGN PTYPE = TYPE4
234   0136          ASSIGN PREV = FRV4
235   0137          FOR JLOOP = 4 TO 10
236   0138          IF (PTYPE) THEN                          ; If some alarm exists
237                    BEGIN                          ;be1
238   0139             BITVAL = PAI:AI_ALMCT
239   0140             IF !AND(BITVAL,16) THEN
240                         BEGIN                     ;be2
241   0141                  IF AND(BITVAL,15) THEN              ; If alarm exists and it is current
242                              BEGIN                ;be3
243   0143                       IF AND(BITVAL,32) THEN NXT = 2  ; If alarm is ack
244                              ELSE
245                                 BEGIN             ;be4
246   0144                          IF PSKID !=BITVAL THEN
247                                       BEGIN       ;be5
248   0146                                IF BITVAL < 3 THEN FALMON = 1
249   0147                                PSKID = BITVAL
250   0148                                PDEV:DEV_TMP0 = 2
251   0149                                NXT = 1
252   0150                                LAMP ADI,SKDKEY,2
253   0151                                CRIT
254                                       END         ;en5
255   0152                          ELSE NXT = 1
256                                 END               ;en4
257                         END                       ;en3
258                      ELSE                                ; Else there is no alarm
259                         BEGIN
260   0153                  NXT = 0
261   0154                  PSKID = 0
262                         END
263                      END                          ;en2
264                   ELSE                                   ; Else alarm is inhibited
265                      BEGIN
266   0155               NXT = 0
267   0156               PSKID = 0
268                      END
269   0157             ANALOG = OR(ANALOG,NXT)
270                   END                             ;en1
271   0158          STEP PSKID BY 1
272   0159          STEP PTYPE BY 1
273   0160          STEP PREV BY 1
274   0161          STEP PAI BY 1
275   0162          NEXT JLOOP
```

```
276             ;
277             ;       END ANALOG RED AND YELLOWS
278             ;
279     0163    WHT:    WHITE = 0
280     0164            SSNG0 = "RUN"
281     0165            MARK STEP "BELOW",0,0
282     0166            FTRIP4 = 0
283     0168            IF LAST > 3 THEN PRINT ADI," LAMP VARIABLE IS GREATER THAN 3!"
284     0169            CURENT = OR(LAST,WHITE)
285     0170            CURENT = OR(CURENT,ANALOG)
286     0172            IF AND(CURENT,1)THEN LMPVAR = 2
287     0174               ELSE IF AND(CURENT,2)THEN LMPVAR = 1
288     0175                       ELSE LMPVAR = 0
289     0176            LAMP ADI,SKDKEY,LMPVAR
290     0177            MARK STEP "BOTTOM",0,0
291     0178            NOCRIT
292     0179            GOTO AGAIN
293             ;
294             ;       E N D   P R O G R A M   L O O P
295             ;
296             ;       BEGIN CLEAR, ALARM, ACKNOWLEDGE, MESSAGE, ERROR, AND TERMINATE SUBROUTINES:
297             ;
298     0180    CLR:    PSTAT = 0                                               ; Int set to zero
299                     ;
300                     ; If not white, then unack:
301                     ;
302     0183            IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
303     0184            FCLEAR = 1                                              ; Clear flag
304     0185            GOSUB ACKK
305     0186            RETURN
306             ;
307     0187    ALM:    PSTAT = 1                                               ; Int set to one
308                     ;
309                     ; If not white, then unack:
310                     ;
311     0190            IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
312                     ;
313                     ; Turn audible alarm on if white or yellow:
314                     ;
315     0192            IF AND(PTYPE,3) THEN FALMON = 1
316     0193            PDEV:DEV_TMP0 = 2
317     0194            NOCRIT
318     0195            CRIT
319     0196            LAMP ADI,SKDKEY,2                                       ; Blink LAMP
320     0197            GOSUB ACKK                                              ; Print message
321     0198            RETURN
322             ;
323     0199    ACK1:   PSTAT = 2                                               ; Ack the alarm
324     0200            FACK = 1                                                ; Set ACK flag for C program
325             ;
326             ;       ALARM TYPER STRINGS:
327             ;
328     0202    ACKK:   IF ILOOP = 1 THEN STRNG0 = "A-350 TANK LEVEL HIGH"
329     0204            IF ILOOP = 2 THEN STRNG0 = "A-350 LIFT STATION LEAK"
330     0206            IF ILOOP = 3 THEN STRNG0 = "PW-481 WASTE LINE LEAK"
```

```
331   0208            IF ILOOP = 13 THEN STRNGO = "350BATCH PGM STARTING "
332   0210            IF ILOOP = 14 THEN STRNGO = "350BATCH WRONG UNIT   "
333   0212            IF ILOOP = 15 THEN STRNGO = "350BATCH TERMINATING  "
334           ;
335           ; If acknowledging the alarm, then:
336           ;
337   0213    AMSG:     IF PTYPE = 1 THEN
338                        BEGIN
339   0214                 IF (FACK) THEN
340                          BEGIN
341                          ;
342                          ; If white, print it is ack and reset the flag:
343                          ;
344   0215                   PRINT ADI,"%p ALARM ACKNOWLEDGE",STRNGO
345   0216                   FACK = 0
346                          END.
347   0217                 ELSE PRINT ADI,"WHITE ALARM %p",STRNGO
348                        END
349           ;
350           ; If CLEAR flag is set, print it is clear and reset the flag:
351           ;
352   0218              IF (FCLEAR) THEN
353                        BEGIN
354   0219                 PRINT ADI,"%p ALARM CLEAR",STRNGO
355   0220                 FCLEAR = 0
356                        END
357           ;
358           ;  If STARTUP, then print startup message:
359           ;
360   0221              IF (FSTART) THEN
361                        BEGIN
362   0222                 FSTART = 0
363   0223                 PRINT ADI,"%p",STRNGO
364                        END
365   0224              RETURN
366           ;
367           ; If ERROR - print message, start timer, and set flag:
368           ;
369   0226    ERRSUB:   IF(ERRNUM() = -187) THEN ERSRET
370                     ELSE
371                      BEGIN
372   0227               IF ERRN != ERRNUM() THEN
373                        BEGIN
374   0228                 TIMERON STMR0
375   0229                 TIMERCLR STMR0
376   0230                 FSN0 = 1
377   0231                 ERRN = ERRNUM()
378   0232                 ERRL = ERRLIN()
379   0233                 PRINT ADI,"ERROR # %i AT LINE %i",errn,errl
380   0234                 ERSCLR
381                        END
382                   END
383   0235            GOTO AGAIN
384           ;
385           ; If FABORT = 1, then abort program:
```

```
386             ;
387             ; TERM:  FABORT = 0
388     0236    TERM:    STRNG0 = "350BATCH TERMINATED    "
389     0237             ILOOP = STAR + 2
390     0238             FSTART = 1
391     0239             SSNG0 = "TERM"
392     0240             GOSUB ACKK
393                      END

       Total Errors Detected      0     Total Warnings      0
            Sequence Program Size = 4416 Bytes, 276 Pages
                 10 Forward Reference(s)
```

FEEDBATCH

```
1           ;       F E E D B A T C H . S E Q                        VERSION 3
2           ;
3           ;       Function:       Sequence program for Lamp function,
4           ;                       message display on Graphic number 301,
5           ;                       and printer messages.
6           ;
7           ;       Written By      : R. N. Kersey
8           ;       Date            : 05-31-92
9           ;       Facility        : 242-A Evaporator Bldg.
10          ;       Machine         : VAX 4104-A
11          ;       Op. System      : VAX/VMS 5.5-2AH
12          ;       D/3 Version     : 6.3-2 BASELINE
13          ;       Compiler        : SIC
14          ;       Build command   : @[320,20]FEEDBATCH
15          ;
16          ;       NOTE:  lines commented out with ';OLD' are temporary changes to be
17          ;              reimplemented when operational testing is done.  Related
18          ;              changes are in PCBATCH.
19          ;
20          ;       UNIT = 5, SKDKEY = 31
21          ;
22          ;       Assoc. UPF:     [300,1]_PRCESS.___      PRCESS.LST
23          ;       Assoc. RPF:     [300,1]_242A.___        242A.LST
24          ;       SDB:            [300,1]PCM0A.SDB        PCM0ASDB.CMD
25          ;
26          ;       POINTER   SDB VAR.      FUNCTION
27          ;       -------   ---------     ----------------------------------------
28          ;       PSTAT     INTn          Current EPN alarm state is an integer tied to
29          ;                               graphic flashing string of value:
30          ;       .         (n=0-100)             0 = previously not alarmed
31          ;       .                               1 = previously alarmed, not ack
32          ;       .                               2 = previously alarmed, is ack
33          ;       .                               3 = alarm is inhibited
34          ;       .                               4 = input is off scan
35          ;       .                               5 = input is bad
36          ;       .         LEAKn         Special INT variable for multiple white alarms on an EPN
37          ;       PDIN      'EPN'         ---
38          ;       PTYPE     TYPEn         alarm type 0 = none, 1 = white, 2 = yellow,
39          ;                                 3 = yellow and yellow, 4 = red,
40          ;                                 6 = yellow and red
41          ;       PREV      FRVn          Reverse acting (not fail safe) flag (1=set)
42          ;       PSKID     SKIDn         Used for analog alarms
43          ;       PPASS     FPASn         Turn lamp off when alarm is ACKnowledged
44          ;       PALARM    FALMn         Analog alarm flag
45          ;       ---       FNING         Unused error flag
46          ;
47          ;  MODIFICATION LOG:
48          ;
49          ;       16-NOV-92       Added O/C display bar feature.          M. Teats
50          ;
51          ;       23-NOV-92       Added 102AW pressure signals, analog red M. Teats
52          ;                       and yellow section.
53          ;
54          ;       30-DEC-92       Fixed white alarm print message.        M. Teats
55          ;
```

```
56          ;         13-MAY-93      Removed II-AW-102 hi white alarm per CCN#        M. Teats
57          ;                        93MAY13-01.
58          ;
59          ;-mct1+  25-MAY-93       In ALM routine, changed FACK=1 to FALMON=1       M. Teats
60          ;                        to prevent "alarm ack" message printout
61          ;                        when alarm occurs.  CCN# 93MAY25-01.
62          ;
63          ;-mct2+  01-JUN-93       Add code to put FIC-CA1-1 in manual and set      M. Teats
64          ;                        its output to zero when HV-CA1-1 closes.
65          ;                        CCN# 93JUN01-01.
66          ;
67          ;         07-JUN-93      Added comments.  CCN# 93JUN01-01.                M. Teats
68          ;
69          ;         20-JUL-93      Commented and baselined for D/3 version 3.3.     M. Teats
70          ;
71          ;-mct3+   12-JAN-94      Delete LO white alarm on II-P-C100 per           M. Teats
72          ;                        CCN# 94JAN12-01.
73          ;
74          ;         15-DEC-94      Revised header to indicate the new VMS           J. Berger
75          ;                        system. Rebase line document. Removed
76          ;                        obsolite Select() statements.
77          ;
78          ;  END OF MOD LOG
79          ;
80          UNIT     FEED
81          ;
82                   COMMON WHOLE0,WHOLE1,CA1          ; CA1 for synch PID
83                   FLAGBITS == PSPAR0
84                   HV_TMP0  == 'VCA1-INLK':DEV_TMP0
85                   HV_TMP1  == 'HV-CA1-1':DEV_TMP1
86                   PAW_TMP1 == 'P-AW-102':DEV_TMP1
87                   PAW_STAT == 'P-AW-102':DEV_STAT
88                   FSN0 == FSPAR0
89                   FIC1SP == 'FIC-CA1-1':PID_SPVL:3 ; For synch PID
90          ;
91    0000           NOP
92    0001           STAR = 24                         ; Start message
93    0002           LET SKDKEY = 31          ; SKID Key
94    0003           LET UNITT = 5                     ; SEQ Unit number
95    0004           ASSIGN PDEV = 'PAW102ALM'         ; Alarm device
96    0005           ASSIGN PTYPE = TYPE24
97    0006           LET ADI = 0                       ; Alarm destination index
98    0007           LET FSN0 = 0                      ; No alarm yet
99    0008           TIMEROFF STIME0          ; Alarm timer off
100   0009           TIMERCLR STIME0          ; Alarm timer clear
101   0010           LET STRNG0 = " "                  ; Blank 30 character string
102   0011           LET ERRN = 0                      ; Set error number to zero
103   0012           LET ERRL = 0                      ; Set error line to zero
104   0013           LET FTRIP4 = 0                    ; FTRIP4 = 1 when audible horn is to be turned on
105   0014           GETUNIT TRIP                      ; Which sequence unit is program running in
106   0015           IF TRIP != UNITT THEN             ; Abort if running in wrong seq. unit
107                     BEGIN
108   0016              FSTART = 1
109   0017              ILOOP = STAR + 1
110   0018              PTYPE = 0
```

```
111  0019              GOSUB ACKK
112  0020              GOTO TERM
113                    END
114           ;      IF !SELECT() THEN GOTO AGAIN
115           ;
116           ;      B E G I N   I N I T I A L I Z I N G
117           ;
118           ;      ILOOP 0-6 are DINS, 7-16 are AINs
119           ;
120  0021             LET STRNG0 = "INIT"
121  0022  INIT:      LET FABORT = 0                  ; If FABORT = 1 then abort program
122  0023             LET FSTART = 0                  ; Program start flag = 0
123  0024             LAMP ADI,SKDKEY,0               ; Turn off the annunciator key
124  0025             LET LSG0 = " "                  ; Clear the string variables
125  0026             LET LSG1 = " "
126  0027             LET LSG2 = " "
127  0028             LET LSG3 = " "
128  0029             LET LSG4 = " "
129  0030             PDEV:DEV_TMP0 = 1                              ; Set alarm device to non-alarm state
130  0031             IF AND(PDEV:DEV_ALMCT,16) THEN                 ; If alarm device is inhibited, remove the inhibit
131  0032               PDEV:DEV_ALMCT=XOR(PDEV:DEV_ALMCT,16)
132  0033             IF AND(PDEV:DEV_ALMCT,32) THEN                 ; If alarm device is ack, clear it
133  0034                PDEV:DEV_ALMCT = XOR(PDEV:DEV_ALMCT,32)
134           ;
135           ;      First, set all flags and integers:
136           ;
137  0035             ASSIGN PDIN = 'YS-AW102'              ; Assign pointer to first DIN
138  0036             ASSIGN PPASS = FPAS0
139  0037             ASSIGN PREV = FRV0
140  0038             ASSIGN PSTAT = INT0
141  0039             ASSIGN PTYPE = TYPE0
142  0040             ASSIGN PSKID = SKID0
143           ;
144  0041             FOR ILOOP = 0 TO 16
145  0042             PPASS = 0
146  0043             PREV = 0
147  0045             IF PSTAT > 5 THEN PSTAT = 0                    ; Reset current status if undefined
148  0046             PSKID = 0                                      ; Default previous status to no alarms
149  0047             PTYPE = 2                                      ; Default to Yellow alarm
150  0048             IF ILOOP <= 6 THEN                             ; For DINs
151                       BEGIN
152  0049                    IF AND(PDIN:DIN_DFLG,3) THEN            ; Check for alarms if defined (must be red or yellow)
153                             BEGIN
154  0050                       PTYPE = 2                           ; Default to Yellow alarm
155  0052                       IF AND(PDIN:DIN_DFLG,1) THEN LET PREV = 1      ; Reverse acting
156  0054                       IF AND(PDIN:DIN_DFLG,4) THEN LET PTYPE = 4     ; Set to red critical alarm
157                             END
158                       END
159  0055             ELSE PTYPE = 1                                 ; Set to white alarm
160  0057             IF ILOOP < 6 THEN STEP PDIN BY 1       ; Point to next DIN
161  0058             STEP PPASS BY 1
162  0059             STEP PREV BY 1
163  0060             STEP PSTAT BY 1
164  0061             STEP PSKID BY 1
165  0062             STEP PTYPE BY 1
```

```
166   0063          NEXT ILOOP
167         ;
168   0064          LET FPAS0 = 1           ; YS-102AW, turn lamp off when alarm is ACK
169         ;
170   0065          LET TYPE0 = 0           ; YS-AW102 white alarm deleted -mct3+
171   0066          LET TYPE5 = 0           ; ZS-CA1-1C, no alarm
172   0067          LET TYPE6 = 0           ; ZS-CA1-1N, no alarm
173         ;
174   0068          LET FRV0 = 0            ; YS-AW102 alarm on change from 1 to 0
175   0069          LET FRV1 = 1            ; YS-CA1-1 alarm on change from 0 to 1
176   0070          LET FRV2 = 0            ; LDS-SMPL1 alarm on change from 1 to 0
177   0071          LET FRV7 = 1            ; II-AW-102 is a white alarm
178         ;
179   0072          LET AI0 = 12.0          ; II-AW-102 Lo alarm
180         ;
181   0073          FLG13 = 0               ; PID casc (1) flag, for synch FIC-CA1-1 > LIC
182   0074          FLG14 = 0               ; PID auto (1) flag, for synch FIC-CA1-1 > LIC
183   0075          FLG15 = 0               ; PID manual (1) flag, for synch FIC-CA1-1 > LIC
184   0076          AI20 = 0.0              ; For PID O/C display bar, FIC-CA1-1 manual (-mct2+)
185         ;
186         ;       E N D   I N I T I A L I Z I N G
187         ;
188         ;       S E N D   P R O G R A M   S T A R T   M E S S A G E
189         ;
190   0077          ANALOG = 0
191   0078          ILOOP = STAR
192   0079          FPASS = 1
193   0080          FSTART = 1
194   0081          GOSUB ACKK
195   0082          FPASS = 0
196   0083          ON ERROR GOSUB ERRSUB
197   0084          ESTPGM "SHUTDOWN"
198         ;
199         ;       B E G I N   P R O G R A M   L O O P
200         ;
201   0085  AGAIN:  MARK STEP "WTUNTL",0,0
202         ;       IF !SELECT() THEN FTRIP0 = 0     ; If micro not selected then clear flag
203         ;           ELSE FTRIP0 = 1              ; Flag Determines if DOG0A.TSK should run in this micro
204         ;       WAITUNTIL SELECT() ¦ FABORT      ; Wait Until micro is Selected or program aborted
205         ;       IF FABORT THEN GOTO TERM ; Terminate program if abort flag set
206                 ;
207                 ; If POWERFAIL flag set, terminate program and run emergency stop program.
208                 ;
209   0087          IF PWRFAIL() THEN ESTOP
210   0088          MARK STEP "RNNING",0,0
211   0089          WAIT 1
212   0090          IF (FSN0) THEN                  ; If alarm flag set
213   0091             IF STIME0 > 60 THEN          ; If alarm flag active for 1 minute
214                         BEGIN                  ; Reset timer and turn off flag
215   0092                  TIMEROFF STIME0
216   0093                  TIMERCLR STIME0
217   0094                  LET FSN0 = 0
218   0095                  LET ERRN = 0
219   0096                  LET ERRL = 0
220                         END
```

```
221           ;
222           ;       If PAW102ALM $T1 is alarmed and it's ALMCT indicates alarms are acknowledged:
223           ;
224    0097           IF (PDEV:DEV_TMP0 = 2) & AND(PDEV:DEV_ALMCT,32) THEN
225                     BEGIN
226    0098             LET PDEV:DEV_TMP0 = 1 ; Put PAW102ALM in ack state
227    0099             LET FTRIP4 = 1                ; Set for Unit alarm ack, system to graphic
228    0100             LET ACK = 1                   ; Set for Unit alarm ack, graphic to system
229    0101             CRIT
230    0102             LAMP ADI,SKDKEY,1             ; Turn lamp on
231                     END
232           ;
233           ;       BEGIN DINs:
234           ;
235    0103           ASSIGN PDIN = 'YS-AW102'
236    0104           ASSIGN PTYPE = TYPE0
237    0105           ASSIGN PREV = FRV0
238    0106           ASSIGN PSTAT = INT0
239    0107           ASSIGN PPASS = FPAS0
240    0108           LET CURENT = 0                ; Used to set LAMP status
241    0109           LET LAST = 0
242    0110           LET ANALOG = 0
243    0111           LET NXT = 0
244    0112           LMPVAR = 0
245    0113           BITVAL = 0
246           ;
247    0114   LOP:    FOR ILOOP = 0 TO 6
248    0115           IF PTYPE THEN                                          ; If some type of alarm
249                     BEGIN                                        ;be1
250    0117             IF AND(PDIN:DIN_ALMCT,16) THEN LET PSTAT = 3         ; If EPN inhibited
251    0119             IF AND(PDIN:DIN_ALMCT,4) THEN LET PSTAT = 5          ; If EPN is bad input
252    0121             IF !AND(PDIN:DIN_FLG,8) THEN LET PSTAT = 4           ; If EPN is disabled
253    0122             IF PSTAT < 3 THEN                                    ; If not inhibited
254                       BEGIN                                      ;be2
255    0123               IF PSTAT = 1 THEN
256                         BEGIN                                    ;be3
257    0124                 IF PTYPE != 1 THEN
258    0126                 IF AND(PDIN:DIN_ALMCT,32) THEN GOSUB ACK1
259    0128                 IF (FTRIP4) THEN GOSUB ACK1
260                         END                                      ;en3
261    0131               IF PSTAT = 0 THEN IF(PDIN:DIN_VAL = PREV) THEN GOSUB ALM
262    0132               IF PSTAT THEN
263                         BEGIN                                    ;be4
264    0134                 IF PDIN:DIN_VAL != PREV THEN GOSUB CLR
265    0135                 NXT = PSTAT
266    0137                 IF (PPASS) & (PSTAT = 2 ) THEN NXT = 0
267    0138                 LAST = OR(LAST,NXT)
268                         END                                      ;en4
269                       END                                        ;en2
270    0140             ELSE IF !AND(PDIN:DIN_ALMCT,16)&!AND(PDIN:DIN_ALMCT,4)&AND(PDIN:DIN_FLG,8)THEN LET PSTAT = 0
271                     END                                          ;en1
272    0141           STEP PDIN BY 1
273    0142           STEP PTYPE BY 1
274    0143           STEP PREV BY 1
275    0144           STEP PSTAT BY 1
```

```
276   0145          STEP PPASS BY 1
277           ;     IF !SELECT() THEN GOTO AGAIN
278   0146          REMAIN = LAST
279   0147          NEXT ILOOP
280           ;
281           ;     END DINs
282           ;
283           ;     BEGIN ANALOG RED AND YELLOWS:
284           ;
285   0148          ASSIGN PAI = 'II-AW-102'
286   0149          ASSIGN PSKID = SKID7
287   0150          ASSIGN PTYPE = TYPE7
288   0151          FOR JLOOP = 7 TO 16
289   0152          IF (PTYPE) THEN                                  ; If some alarm exists
290                    BEGIN                                   ;be1
291   0153             BITVAL = PAI:AI_ALMCT
292   0154             IF !AND(BITVAL,16) THEN                       ; If not inhibited
293                         BEGIN                              ;be2
294   0155                  IF AND(BITVAL,15) THEN                   ; If alarm exists and is current
295                              BEGIN                         ;be3
296   0157                       IF AND(BITVAL,32) THEN NXT = 2       ; If alarm is ack
297                              ELSE
298                                 BEGIN                      ;be4
299   0158                          IF PSKID !=BITVAL THEN
300                                       BEGIN                ;be5
301   0159                                LET PSKID = BITVAL
302   0161                                IF BITVAL < 3 THEN FALMON = 1
303   0162                                PDEV:DEV_TMP0 = 2
304   0163                                NXT = 1
305                                       END                  ;en5
306   0164                          ELSE NXT = 1
307                                 END                        ;en4
308                              END                           ;en3
309                         ELSE                                  ; Else there is no alarm
310                              BEGIN
311   0165                       NXT = 0
312   0166                       LET PSKID = 0
313                              END
314                         END                                ;en2
315                    ELSE                                       ; Else alarm is inhibited
316                         BEGIN
317   0167                  NXT = 0
318   0168                  LET PSKID = 0
319                         END
320   0169             ANALOG = OR(ANALOG,NXT)
321                    END                                     ;en1
322   0170          STEP PSKID BY 1
323   0171          STEP PTYPE BY 1
324   0173          IF JLOOP < 16 THEN STEP PAI BY 1
325   0174          NEXT JLOOP
326           ;
327           ;     END ANALOG RED AND YELLOWS
328           ;
329           ;     BEGIN ANALOG WHITE ALARMS:
330           ;
```

```
331   0175            WHITE = 0
332   0176            ASSIGN PAI = 'II-AW-102'
333   0177            ASSIGN PALARM = AI0
334   0178            ASSIGN PSTAT = INT7
335   0179            ASSIGN PTYPE = TYPE7
336   0180            ILOOP = 7
337   0181            GOSUB ANLG
338                   ;
339                   ; II-AW-102 Lo, use with String packet 78:
340                   ;
341   0183            IF (INT7 & INT7 < 3) THEN LEAK0 = INT7 + 2
342   0184               ELSE LEAK0 = 0
343           ;
344           ;       END ANALOG WHITE ALARMS:
345           ;
346   0185            MARK STEP "BELOW",0,0
347   0186            FTRIP4 = 0
348   0188            IF REMAIN > 3 THEN PRINT ADI," LAMP VARIABLE IS GREATER THAN 3!"
349   0189            CURENT = OR(REMAIN,WHITE)
350   0190            CURENT = OR(CURENT,ANALOG)
351   0192            IF AND(CURENT,1) THEN LMPVAR = 2
352   0194              ELSE IF AND(CURENT,2) THEN LMPVAR = 1
353   0195                 ELSE LMPVAR = 0
354   0196            LAMP ADI,SKDKEY,LMPVAR
355   0197            MARK STEP "BOTTOM",0,0
356   0198            NOCRIT
357           ;
358           ;       BEGIN HV-CA1-1 LOGIC (for graphic only):
359           ;
360   0199            ASSIGN FLAGBITS = AT0
361   0200            BVAL = 1
362   0201                FOR IN1 = 0 TO 12
363   0202                FLAGBITS = AND(BVAL,HV_TMP0)
364   0203                BVAL=BVAL(1
365   0204                STEP FLAGBITS BY 1
366   0205                NEXT IN1
367   0206            SA1  = 'YS-CA1-1':DIN_VAL
368   0207            AT25 = AND(HV_TMP1,512)
369   0208            AT26 = AND(HV_TMP1,1024)
370   0209            AT27 = AND(HV_TMP1,2048)
371   0210            AT30 = AND(HV_TMP1,16384)
372   0211            AT31 = AND(HV_TMP1,32768)
373   0212            SA2  = AND('RSH-VVB/G':DIN_VAL,'PDSH-FC51':DIN_VAL)
374   0213            AT14 = AND('WFSH-CA11':DIN_VAL,'WFSH-CA12':DIN_VAL)
375   0214            AT15 = AND('RSH-CA1-1':DIN_VAL,'PDSH-CA1':DIN_VAL)
376   0215            AT14 = AND(AT14,SA2)
377   0216            SA3  = AND(AT14,AT15)
378   0217            SA4  = OR('YS-CA1-1':DIN_VAL,SA3)
379   0218            AT14 = AND(HV_TMP1,32)
380   0219            AT15 = AND('HV-CA1-1':DEV_STAT,2)
381   0220            SA5  = AND('HV-CA1-1':DEV_CMD,2)
382   0221            SA6  = AND(SA5,SA4)
383   0222            SB0  = 'ZS-CA1-1C':DIN_VAL
384   0223            SB1  = 'ZS-CA1-1N':DIN_VAL
385   0224            AT20 = AT12
```

```
386   0225            AT21 = AND('VCA1-INLK':DEV_CMD,2)
387   0226            AT23 = AND('VCA1-INLK':DEV_TMP1,1)
388   0227            AT22 = XOR(AT21,AT23)
389   0228            AT28 = AT31
390           ;
391           ;       END HV-CA1-1 LOGIC
392           ;
393           ;       BEGIN P-AW-102 LOGIC (for graphic only):
394           ;
395   0229            ASSIGN FLAGBITS = BT23
396   0230            BVAL = 128
397   0231                FOR IN1 = 23 TO 29
398   0232                FLAGBITS = AND(BVAL,PAW_TMP1)
399   0233                BVAL=BVAL(1
400   0234                STEP FLAGBITS BY 1
401   0235                NEXT IN1
402   0236            BT31 = AND(PAW_TMP1,32768)
403   0237            BT0  = AND('HS-ARM-AW':DEV_TMP0,128)
404   0238            BT1  = OR('RSH-AW-12':DIN_VAL,BT0)
405   0239            AT15 = AND('LDS-AW-PP':DIN_VAL,!'YS-AW-MSD':DIN_VAL)
406   0240            BT2  = AND(AT15,BT1)
407   0241            AT14 = AND('RSH-SP-AW':DIN_VAL,'LDS-SN269':DIN_VAL)
408   0242            AT15 = AND('LDS-SN270':DIN_VAL,'PSH-AWSN2':DIN_VAL)
409   0243            AT14 = AND(AT14,AT15)
410   0244            BT3  = AND(BT2,AT14)
411   0245            AT14 = AND('WFSH-CA11':DIN_VAL,'WFSH-CA12':DIN_VAL)
412   0246            AT15 = AND('RSH-VVB/G':DIN_VAL,'RSH-CA1-1':DIN_VAL)
413   0247            AT16 = AND('PDSH-CA1':DIN_VAL,'RSH-CA1-1':DIN_VAL)
414   0248            AT17 = AND(BT3,'LSH-SC-1':DIN_VAL)
415   0249            AT18 = AND(AT14,AT15)
416   0250            AT19 = AND(AT16,AT17)
417   0251            BT4  = AND(AT18,AT19)
418   0252            BT5  = AND('P-AW-102':DEV_TMP0,2)
419   0253            BT6  = 'YS-AW102':DIN_VAL
420   0254            BT14 = AND(BT31,PAW_STAT)
421   0255            SB3 = AND(PAW_STAT,4)
422   0256            SB4 = AND(PAW_STAT,16)
423   0257            SB5 = AND('HV-CA1-1':DEV_STAT,4) ; RESET
424   0258            SB6 = AND('HV-CA1-1':DEV_STAT,16)        ; CHGSTATE
425   0259            SB7 = AND('HV-CA1-1':DEV_STAT,32)        ; INTERLOCK
426   0260            SC0 = AND(PAW_STAT,32)
427   0261            BT20 = AND(HV_TMP1,16)
428   0262            BT7  = AND(BT20,BT14)
429   0263            BT8  = AND(BT7,BT4)
430   0265            IF AND(PAW102ALM:DEV_FLG,64)THEN PUTMANL PAW102ALM
431   0267            IF AND('P-AW-102':DEV_FLG,64)THEN PUTMANL 'P-AW-102'
432   0269            IF AND('HV-CA1-1':DEV_FLG,64)THEN PUTMANL 'HV-CA1-1'
433   0271            IF AND('VCA1-INLK':DEV_FLG,64)THEN PUTMANL 'VCA1-INLK'
434           ;
435           ;       END P-AW-102 LOGIC
436           ;
437           ;OLD    B E G I N   S Y N C H   P I D   C A S C A D E D   S E T P T
438           ;OLD
439           ;OLD    Fucntion:  Closes slave controller's PID (SPID) cascade,
440           ;OLD               sets cascade controller's PID (CPID) to auto,
```

```
441          ;OLD              sets the CPID output equal to the SPID setpoint
442          ;OLD              only on a manual to auto transition in the SPID.
443          ;OLD              Puts LIC's in manual if FIC was previously auto
444          ;OLD              and cascade open.
445          ;OLD
446          ;OLD     IF AND('FIC-CA1-1':CB_FLG:3,64) THEN            ; If in auto
447          ;OLD        BEGIN                                                ;be1
448          ;OLD          IF (FLG14=0) & (FLG15=0) THEN                ; If initializing
449          ;OLD             BEGIN                                          ;be2
450          ;OLD             IF AND('FIC-CA1-1':PID_SPVS:3,64) THEN BLET FLG13 = 1  ; If casc closed
451          ;OLD             BLET FLG14 = 1
452          ;OLD             BLET FLG15 = 0
453          ;OLD             END                                            ;en2
454          ;OLD          IF AND('FIC-CA1-1':PID_SPVS:3,64) & (FLG13=0) THEN   ; If clos casc + prev auto-open
455          ;OLD             BEGIN                                          ;be3
456          ;OLD               IF !AND('LIC-CA1-1':CB_FLG:1,64) THEN  ; If LIC-CA1-1 in manual
457          ;OLD                  BEGIN                                    ;be3.1
458          ;OLD                  BPUTOUT 'LIC-CA1-1',FIC1SP
459          ;OLD                  BPUTAUTO 'LIC-CA1-1'
460          ;OLD                  END                                      ;en3.1
461          ;OLD               IF !AND('LIC-CA1-2':CB_FLG:1,64) THEN  ; If LIC-CA1-2 in manual
462          ;OLD                  BEGIN                                    ;be3.2
463          ;OLD                  BPUTOUT 'LIC-CA1-2',FIC1SP
464          ;OLD                  BPUTAUTO 'LIC-CA1-2'
465          ;OLD                  END                                      ;en3.2
466          ;OLD             BLET FLG13 = 1
467          ;OLD             END                                            ;en3
468          ;OLD          IF !AND('FIC-CA1-1':PID_SPVS:3,64) & (FLG13=1) THEN ; If open casc + prev auto-clos
469          ;OLD             BEGIN                                          ;be4
470          ;OLD             IF AND('LIC-CA1-1':CB_FLG:1,64) THEN BPUTMANL 'LIC-CA1-1'
471          ;OLD             IF AND('LIC-CA1-2':CB_FLG:1,64) THEN BPUTMANL 'LIC-CA1-2'
472          ;OLD             BLET FLG13 = 0
473          ;OLD             END                                            ;en4
474          ;OLD          IF (FLG14=0) & (FLG15=1) THEN                ; If FIC previously manual
475          ;OLD             BEGIN                                          ;be5
476          ;OLD             CRIT
477          ;OLD             BPUTOUT 'LIC-CA1-1',FIC1SP
478          ;OLD             BPUTOUT 'LIC-CA1-2',FIC1SP
479          ;OLD             BCLOSCASC 'FIC-CA1-1'
480          ;OLD             IF !AND('LIC-CA1-1':CB_FLG:1,64) THEN BPUTAUTO 'LIC-CA1-1'
481          ;OLD             IF !AND('LIC-CA1-2':CB_FLG:1,64) THEN BPUTAUTO 'LIC-CA1-2'
482          ;OLD             BLET FLG13 = 1
483          ;OLD             BLET FLG14 = 1
484          ;OLD             BLET FLG15 = 0
485          ;OLD             NOCRIT
486          ;OLD             END                                            ;en5
487          ;OLD        END                                                 ;en1
488          ;OLD     ELSE                                                ; If FIC in manual
489          ;OLD        BEGIN                                                ;be7
490          ;OLD          IF (FLG14=1) & (FLG15=0) THEN                ; If FIC previously auto
491          ;OLD             BEGIN                                          ;be8
492          ;OLD             IF AND('LIC-CA1-1':CB_FLG:1,64) THEN BPUTMANL 'LIC-CA1-1'
493          ;OLD             IF AND('LIC-CA1-2':CB_FLG:1,64) THEN BPUTMANL 'LIC-CA1-2'
494          ;OLD             BLET FLG13 = 0
495          ;OLD             END                                            ;en8
```

```
496          ;OLD          IF ((FLG14=1)&(FLG15=0)) | ((FLG14=0)&(FLG15=0)) THEN ; If FIC prev auto or init
497          ;OLD             BEGIN                                                ;be9
498          ;OLD             BLET FLG14 = 0
499          ;OLD             BLET FLG15 = 1
500          ;OLD             END                                                  ;en9
501          ;OLD          END                                                     ;en7
502          ;OLD
503          ;OLD     E N D   S Y N C H   P I D   C A S C A D E D   S E T P T
504          ;+mct2-
505          ;        B E G I N   F E E D   V A L V E   A C T I O N
506          ;
507          ;        Function:       Upon loss of CF-OPEN, put the control loop in manual
508          ;                        and set the PID block output to zero.
509          ;
510     0273         IF AND('HV-CA1-1':DEV_STAT,2) THEN DLEAK2 = 1                ; If Valve is open
511     0274         IF !AND('HV-CA1-1':DEV_STAT,2) & AND(DLEAK2,1) THEN          ; If not open and 1st pass
512                     BEGIN
513     0275            PUTMANL 'FIC-CA1-1'
514     0276            PRINT ADI," HV-CA1-1 IS NOT OPEN >>> FIC-CA1-1 PUT IN MANUAL"
515     0277            PUTOUT 'FIC-CA1-1',A120
516     0278            PRINT ADI," HV-CA1-1 IS NOT OPEN >>> FIC-CA1-1 OUTPUT SET TO ZERO"
517     0279            DLEAK2 = 0
518                     END
519          ;
520          ;        E N D   F E E D   V A L V E   A C T I O N
521          ;-mct2+
522          ;OLD     B E G I N   O / C   D I S P L A Y   B A R
523          ;OLD
524          ;OLD     Function:       Upon loss of permissive, put the control loop in manual
525          ;OLD                     and set the PID block output to zero.
526          ;OLD
527          ;OLD     IF AND('PB1-BYPAS':DEV_TMP0,256) THEN DLEAK3 = 1        ; Permissive $T8 must be Hi (1)
528          ;OLD     IF !AND('PB1-BYPAS':DEV_TMP0,256) & AND(DLEAK3,1) THEN  ; Permissive lost and 1st pass
529          ;OLD        BEGIN
530          ;OLD        BPUTMANL 'FIC-CA1-1'
531          ;OLD        PRINT ADI," PB1-BYPAS PERMISSIVE LOST >>> FIC-CA1-1 PUT IN MANUAL"
532          ;OLD        BPUTOUT 'FIC-CA1-1',A120
533          ;OLD        PRINT ADI," PB1-BYPAS PERMISSIVE LOST >>> FIC-CA1-1 OUTPUT SET TO ZERO"
534          ;OLD        DLEAK3 = 0
535          ;OLD        END
536          ;OLD
537          ;OLD           E N D   O / C   D I S P L A Y   B A R
538          ;OLD
539     0280         GOTO AGAIN
540          ;
541          ;        E N D   P R O G R A M   L O O P
542          ;
543          ;        BEGIN ANALOG WHITE SUBROUTINE:
544          ;
545          ; ANLG:  IF !SELECT() THEN GOTO AGAIN
546     0281 ANLG:   IF (PTYPE = 1) THEN                          ; If some alarm
547                    BEGIN                                ;be1
548     0282           BITVAL = PAI:AI_ALMCT
549     0283           IF !(AND(BITVAL,16)) THEN                  ; If not inhibited
550                      BEGIN                              ;be2
```

```
551   0285              IF PSTAT = 3 THEN LET PSTAT = 0
552   0286              IF (PREV) THEN                                ; If rev acting, LOLO, or LO
553                       BEGIN                                     ;be3
554   0287                IF (PSTAT = 0) THEN                         ; If not previously alarmed
555                         BEGIN                                   ;be4
556   0289                  IF (PAI:AI_MEAS <= PALARM) THEN GOSUB ALM
557                         END                                     ;en4
558                       ELSE
559                         BEGIN                                   ;be5
560   0290                  IF (PSTAT = 1) THEN                     ; If previously alarmed
561                           BEGIN                                 ;be6
562   0292                    IF (FTRIP4) THEN GOSUB ACK1           ; If ACK flag set, then ack it
563   0295                    IF PTYPE !=1 THEN IF AND(PAI:AI_ALMCT,32) THEN GOSUB ACK1
564                           END                                   ;en6
565   0297                  IF (PAI:AI_MEAS >= PALARM + PAI:AI_DEAD) THEN GOSUB CLR
566                         END                                     ;en5
567                       END                                       ;en3
568                     ELSE                                        ; If fail-safe, HIHI, or HI
569                       BEGIN                                     ;be10
570   0298                IF (PSTAT = 0) THEN
571                         BEGIN                                   ;be11
572   0300                  IF (PAI:AI_MEAS >= PALARM) THEN GOSUB ALM
573                         END                                     ;en11
574                       ELSE
575                         BEGIN                                   ;be12
576   0301                  IF (PSTAT = 1) THEN
577                           BEGIN                                 ;be13
578   0303                    IF (FTRIP4) THEN GOSUB ACK1
579   0306                    IF PTYPE != 1 THEN IF AND(PAI:AI_ALMCT,32) THEN GOSUB ACK1
580                           END                                   ;en13
581   0308                  IF (PAI:AI_MEAS <= PALARM - PAI:AI_DEAD) THEN GOSUB CLR
582                         END                                     ;en12
583                       END                                       ;en10
584   0309                NXT = PSTAT
585   0311                IF (PPASS) & (PSTAT = 2 ) THEN NXT = 0
586                       END                                       ;en2
587                       ELSE
588                         BEGIN
589   0312                  LET PSTAT = 3
590   0313                  NXT = 0
591                         END
592   0314                WHITE = OR(WHITE,NXT)
593                     END                                         ;en1
594   0315            ELSE PSTAT = 0
595   0316            STEP PALARM BY 1
596   0317            STEP PSTAT BY 1
597   0318            STEP PTYPE BY 1
598   0319            STEP PREV BY 1
599   0320            STEP PPASS BY 1
600   0321            RETURN
601           ;
602           ;       END ANALOG WHITE SUBROUTINE
603           ;
604           ;       BEGIN CLEAR, ALARM, ACKNOWLEDGE, MESSAGE, ERROR, AND TERMINATE SUBROUTINES:
605           ;
```

```
606   0322   CLR:    LET PSTAT = 0                                          ;. Int set to zero
607                  ;
608                  ; If not white then unack:
609                  ;
610   0325           IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
611   0326           LET FCLEAR = 1                                         ; Clear flag
612   0327           GOSUB ACKK                                             ; Print message
613   0328           RETURN
614          ;
615   0329   ALM:    LET PSTAT = 1                                          ; Int set to one
616                  ;
617                  ; If not white then unack:
618                  ;
619   0332           IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
620                  ;
621                  ; Turn audible alarm on if white or yellow:
622                  ;
623   0334           IF AND(PTYPE,3) THEN FALMON = 1                        ; -mct1+
624   0335           CRIT
625   0336           LET PDEV:DEV_TMP0 = 2
626   0337           GOSUB ACKK                                             ; Print message
627   0338           RETURN
628          ;
629   0339   ACK1:   LET PSTAT = 2                                          ; Ack the alarm
630   0340           LET FACK = 1                                           ; Set ACK flag for C program
631          ;
632          ;       ALARM TYPER STRINGS:
633          ;
634   0342   ACKK:   IF ILOOP = 0 THEN STRNG0 = "FEED PUMP YS-AW102 SHUTDOWN "
635   0344           IF ILOOP = 1 THEN STRNG0 = "FEED VALVE INTERLOCK BYPASS ON"
636   0346           IF ILOOP = 2 THEN STRNG0 = "FEED SAMPLER ENCLOSURE LEAK   "
637   0348          IF ILOOP = 3 THEN STRNG0 = "TK-102-AW PRESSURE HI "
638   0350           IF ILOOP = 4 THEN STRNG0 = "FEED PUMP INTERLOCK BYPASS ON "
639   0352           IF ILOOP = 7 THEN STRNG0 = "FEED PUMP II-AW-102 AMPS LO "
640   0354           IF ILOOP = 24 THEN STRNG0 = "START FEEDBATCH PGM "
641   0356           IF ILOOP = 25 THEN STRNG0 = "FEEDBATCH WRONG UNIT"
642   0358           IF ILOOP = 26 THEN STRNG0 = "TERMINATE FEEDBATCH "
643          ;
644          ; If acknowledging alarm, then:
645          ;
646   0359   AMSG:   IF PTYPE = 1 THEN
647                     BEGIN
648                     ;
649                     ; If white alarm, print:
650                     ;
651   0361              IF (FACK = 0) & (FCLEAR = 0) & (FSTART = 0) THEN PRINT ADI,"WHITE ALARM %p",STRNG0
652   0362              IF (FACK) THEN
653                       BEGIN
654                       ;
655                       ; If white, print it is ack and reset the flag:
656                       ;
657   0363                PRINT ADI,"%p ALARM ACKNOWLEDGE",STRNG0
658   0364                LET FACK = 0
659                       END
660                    END
```

```
661                     ;
662                     ; If CLEAR flag is set, print it is clear and reset the flag:
663                     ;
664     0365            IF (FCLEAR) THEN
665                        BEGIN
666     0366               PRINT ADI,"%p ALARM CLEAR",STRNGO
667     0367               LET FCLEAR = 0
668                        END
669                     ;
670                     ; If STARTUP, then print startup message:
671                     ;
672     0368            IF (FSTART) THEN
673                        BEGIN
674     0369               LET FSTART = 0
675     0370               PRINT ADI,"%p",STRNGO
676                        END
677     0371            RETURN
678             ;
679             ; If ERROR - print message, start timer, and set flag:
680             ;
681     0372    ERRSUB:  IF ERRN != ERRNUM() THEN
682                        BEGIN
683     0373               TIMERON STIMEO
684     0374               TIMERCLR STIMEO
685     0375               LET FSNO = 1
686     0376               LET ERRN = ERRNUM()
687     0377               LET ERRL = ERRLIN()
688     0378               PRINT ADI,"ERROR # %i AT LINE %i",errn,errl
689                        END
690     0379               ERSCLR
691     0380            GOTO AGAIN
692             ;
693             ; If FABORT = 1, then abort program:
694             ;
695             ; TERM:  LET FABORT = 0
696     0381    TERM:    LET STRNGO = "FEEDBATCH TERMINATED    "
697     0382             LET ILOOP = STAR + 2
698     0383             LET FSTART = 1
699     0384             LET STRNGO = "TERM"
700     0385             GOSUB ACKK
701                      END

      Total Errors Detected      0     Total Warnings      0
           Sequence Program Size = 7584 Bytes, 474 Pages
                19 Forward Reference(s)
```

CA1BATCH

Sequence Instruction Compiler Listing 15:34:11 24-Feb-95 Page 1

```
;       C A 1 B A T C H . S E Q                                 VERSION 2
;
;       Function:       Sequence program for Lamp function,
;                       message display on Graphic number 010,
;                       and printer messages.
;
;       Written By      : R. N. Kersey
;       Date            : 05-31-92
;       Facility        : 242-A Evaporator Bldg.
;       Machine         : VAX 4104-A
;       Op. System      : VAX/VMS 5.5-2AH
;       D/3 Version     : 6.3-2 BASILINE
;       Compiler        : SIC
;       Build command   : @[320,20]CA1BATCH
;
;       UNIT = 6, SKDKEY = 34
;
;       Assoc. UPF:     [300,1]_PRCESS.___      PRCESS.LST
;       Assoc. RPF:     [300,1]_242A.___        242A.LST
;       SDB:            [300,1]PCMOA.SDB        PCMOASDB.CMD
;
;       POINTER   SDB VAR.      FUNCTION
;       -------   ---------     ------------------------------
;       PSTAT     INTn          Current EPN alarm state is an integer tied to
;                               graphic flashing string of value:
;       .         (n=0-100)             0 = previously not alarmed
;       .                               1 = previously alarmed, not ack
;       .                               2 = previously alarmed, is ack
;       .                               3 = alarm is inhibited
;       .                               4 = input is off scan
;       .                               5 = input is bad
;       .         LEAKn         Special INT variable for multiple white alarms on an EPN
;       PDIN      'EPN'         ---
;       PTYPE     TYPEn         alarm type 0 = none, 1 = white, 2 = yellow,
;                                  3 = yellow and yellow, 4 = red,
;                                  6 = yellow and red
;       PREV      FRVn          Reverse acting (not fail safe) flag (1=set)
;       PSKID     SKIDn         Used for analog alarms
;       PPASS     FPASn         Turn lamp off when alarm is ACKnowledged
;       PALARM    FALMn         Analog alarm flag
;       ---       FNING         Unused error flag
;
; MODIFICATION LOG:
;
;       16-SEP-92       Delete white alarms on DDI-CA1-1        M. Teats
;
;       30-DEC-92       Fixed white alarm print message.        M. Teats
;
;       07-MAR-93       Added comments and baselined for        M. Teats
;                       D/3 version 3.3.  CCN# 93JUN01-01.
;
;       15-DEC-94       Revised header to indicate the new VMS          J. Berger
;                       system. Rebase line document. Removed
;                       obsolite Select() statements.
;
```

```
                ;  END OF MOD LOG
                ;
                UNIT    CA1
                ;
                        COMMON PB2
                ;
0000            NOP
0001            STAR = 17                       ; Start message
0002            LET SKDKEY = 34         ; SKID Key
0003            LET UNITT = 6                   ; SEQ Unit number
0004            ASSIGN PDEV = 'CA1-ALARM'       ; Alarm device
0005            ASSIGN PTYPE = TYPE17
0006            LET ADI = 0                     ; Alarm Destination Index
0007            LET FSNO = 0                    ; No alarm yet
0008            TIMEROFF STIME0         ; Alarm timer off
0009            TIMERCLR STIME0         ; Alarm timer clear
0010            STRNG0 = " "                    ; Blank 30 character string
0011            LET ERRN = 0                    ; Set error number to zero
0012            LET ERRL = 0                    ; Set error line to zero
0013            LET FTRIP4 = 0                  ; FTRIP4 = 1 when audible horn is to be turned on
0014            GETUNIT TRIP                    ; Which sequence unit is program running in
0015            IF TRIP != UNITT THEN           ; Abort if running in wrong seq. unit
                   BEGIN
0016               FSTART = 1
0017               ILOOP = STAR + 1
0018               PTYPE = 0
0019               GOSUB ACKK
0020               GOTO TERM
                   END
                ;       IF !SELECT() THEN GOTO AGAIN
                ;
                ;       B E G I N   I N I T I A L I Z I N G
                ;
0021            STRNG0 = "INIT"
0022    INIT:   LET FABORT = 0                  ; If FABORT = 1 then abort program
0023            LET FSTART = 0                  ; Program start flag = 0
0024            LAMP ADI,SKDKEY,0               ; Turn off annunciator key
0025            LET LSG0 = " "                  ; Clear the string variables
0026            LET LSG1 = " "
0027            LET LSG2 = " "
0028            LET LSG3 = " "
0029            LET LSG4 = " "
0030            LET PDEV:DEV_TMP0 = 1                           ; Set alarm device to non-alarm state
0031            IF AND(PDEV:DEV_ALMCT,16) THEN                  ; If alarm device is inhibited, remove the inhibit
                  PDEV:DEV_ALMCT=XOR(PDEV:DEV_ALMCT,16)
0033            IF AND(PDEV:DEV_ALMCT,32) THEN                  ; If alarm device is ack, clear it
0034               PDEV:DEV_ALMCT = XOR(PDEV:DEV_ALMCT,32)
                ;
                ;       First, set all flags and integers:
                ;
0035            ASSIGN PDIN = 'PSH-CA111'                       ; Assign pointer to first DIN
0036            ASSIGN PSTAT = INT0
0037            ASSIGN PTYPE = TYPE0
0038            ASSIGN PREV = FRV0
0039            ASSIGN PSKID = SKID0
```

```
111  0040          ASSIGN PPASS = FPAS0
112          ;
113  0041          FOR ILOOP = 0 TO 39
114  0042          LET PPASS = 0
115  0043          LET PREV = 0
116  0045          IF PSTAT > 5 THEN PSTAT = 0                    ; Reset current status if undefined
117  0046          LET PSKID = 0                                  ; Default previous status to no alarms
118  0047          LET PTYPE = 0                                  ; Default to no alarm
119  0048          IF ILOOP < 4 THEN                              ; For DINs
120                   BEGIN
121  0049             LET PTYPE = 0                               ; Default to no alarm
122  0050             IF AND(PDIN:DIN_DFLG,3) THEN                ; Check for alarms if defined (must be red or yellow)
123                        BEGIN
124  0051                  LET PTYPE = 2                                ; Default to yellow
125  0053                  IF AND(PDIN:DIN_DFLG,1) THEN LET PREV = 1    ; Reverse acting
126  0055                  IF AND(PDIN:DIN_DFLG,4) THEN LET PTYPE = 4   ; Set to red critical alarm
127                        END
128  0056             STEP PDIN BY 1                              ; Point to next DIN
129                   END
130  0058            IF ILOOP > 34 THEN PTYPE = 1                 ; Set to white alarm
131  0059            STEP PPASS BY 1
132  0060            STEP PREV BY 1
133  0061            STEP PSTAT BY 1
134  0062            STEP PSKID BY 1
135  0063            STEP PTYPE BY 1
136  0064          NEXT ILOOP
137          ;
138  0065          LET TYPE4 = 2
139  0066          LET TYPE6 = 4
140  0067          LET TYPE7 = 4
141  0068          LET TYPE10 = 6
142  0069          LET TYPE12 = 6
143  0070          LET TYPE14 = 6
144  0071          LET TYPE16 = 6
145  0072          LET TYPE22 = 4
146  0073          LET TYPE27 = 2
147  0074          LET TYPE29 = 6
148  0075          LET TYPE32 = 6
149          ;       BLET TYPE35 = 0          ; Delete White alarms on DDI-CA1-1
150          ;       BLET TYPE36 = 0          ; Delete White alarms on DDI-CA1-1
151  0076          LET TYPE37 = 1
152  0077          LET TYPE38 = 1
153  0078          LET TYPE39 = 1
154  0079          LET FRV0 = 0
155  0080          LET FRV1 = 0
156  0081          LET FRV2 = 0
157  0082          LET FRV3 = 0
158          ;       BLET FRV35 = 1           ; Delete White alarms on DDI-CA1-1
159          ;       BLET AI0 = -0.08         ; Delete White alarms on DDI-CA1-1
160          ;       BLET AI1 = 0.08          ; Delete White alarms on DDI-CA1-1
161  0083          LET AI2 = 15.0
162  0084          LET AI3 = 15.0
163  0085          LET AI4 = 5.0
164          ;
165          ;       E N D   I N I T I A L I Z I N G
```

```
166             ;
167             ;       S E N D   P R O G R A M   S T A R T   M E S S A G E
168             ;
169     0086            ANALOG = 0
170     0087            ILOOP = STAR
171     0088            FSTART = 1
172     0089            PTYPE = 0
173     0090            GOSUB ACKK
174     0091            ON ERROR GOSUB ERRSUB
175             ;
176             ;       B E G I N   P R O G R A M   L O O P
177             ;
178     0092    AGAIN:  MARK STEP "WTUNTL",0,0
179     0093            STRNG0 = "WAIT"
180             ;       WAITUNTIL SELECT() ¦ FABORT      ; Wait until micro is selected or program aborted
181             ;       IF (FABORT) THEN GOTO TERM       ; Terminate program if FABORT flag is set
182     0094            MARK STEP "RNNING",0,0
183     0095            WAIT 1
184     0096            IF (FSN0) THEN                   ; If alarm flag set
185     0097               IF STIME0 > 60 THEN           ; If alarm flag active for 1 minute
186                           BEGIN                     ; Reset timer and turn off flag
187     0098                  TIMEROFF STIME0
188     0099                  TIMERCLR STIME0
189     0100                  LET FSN0 = 0
190     0101                  LET ERRN = 0
191     0102                  LET ERRL = 0
192                           END
193             ;
194             ;       If CA1-ALARM $T1 is alarmed and it's ALMCT indicates alarms are acknowledged:
195             ;
196     0103            IF (PDEV:DEV_TMP0 = 2 ) & AND(PDEV:DEV_ALMCT,32) THEN
197                        BEGIN
198     0104               LET PDEV:DEV_TMP0 = 1 ; Put CA1-ALARM in ack state
199     0105               LET FTRIP4 = 1                ; Set for unit alarm ack, system to graphic
200     0106               LET ACK = 1                   ; Set for unit alarm ack, graphic to system
201     0107               CRIT
202     0108               LAMP ADI,SKDKEY,1             ; Turn lamp on
203                        END
204             ;
205             ;       BEGIN DINs:
206             ;
207     0109            ASSIGN PDIN = 'PSH-CA111'
208     0110            ASSIGN PTYPE = TYPE0
209     0111            ASSIGN PREV = FRV0
210     0112            ASSIGN PSTAT = INT0
211     0113            ASSIGN PPASS = FPAS0
212     0114            LET CURENT = 0                   ; Used to set LAMP status
213     0115            LET LAST = 0
214     0116            LET ANALOG = 0
215     0117            LET NXT = 0
216     0118            LMPVAR = 0
217     0119            BITVAL = 0
218             ;
219     0120    LOP:    FOR ILOOP = 0 TO 3
220             ;       IF !SELECT() THEN GOTO AGAIN
```

```
221  0121          IF (PTYPE > 0) THEN                                    ; If some type of alarm
222                   BEGIN                                              ;be1
223  0123             IF AND(PDIN:DIN_ALMCT,16) THEN LET PSTAT = 3        ; If EPN inhibited
224  0125             IF AND(PDIN:DIN_ALMCT,4) THEN LET PSTAT = 5         ; If EPN is bad input
225  0127             IF !AND(PDIN:DIN_FLG,8) THEN LET PSTAT = 4          ; If EPN is disabled
226  0128             IF (PSTAT < 3) THEN                                 ; If not inhibited
227                        BEGIN                                         ;be2
228  0129                  IF (PSTAT = 1) THEN
229                           BEGIN                                       ;be3
230  0130                     IF (PTYPE != 1) THEN
231  0132                        IF AND(PDIN:DIN_ALMCT,32) THEN GOSUB ACK1
232  0134                     IF (FTRIP4) THEN GOSUB ACK1
233                           END                                         ;en3
234  0137                  IF PSTAT = 0 THEN IF(PDIN:DIN_VAL = PREV) THEN GOSUB ALM
235  0138                  IF (PSTAT) THEN
236                           BEGIN                                       ;be4
237  0140                     IF (PDIN:DIN_VAL != PREV) THEN GOSUB CLR
238  0141                     NXT = PSTAT
239  0143                     IF (PPASS) & (PSTAT = 2 ) THEN NXT = 0
240  0144                     LAST = OR(LAST,NXT)
241                           END                                         ;en4
242                     END                                               ;en2
243  0146           ELSE IF !AND(PDIN:DIN_ALMCT,16)&!AND(PDIN:DIN_ALMCT,4)&AND(PDIN:DIN_FLG,8)THEN LET PSTAT = 0
244                 END                                                 ;en1
245  0147          STEP PDIN BY 1
246  0148          STEP PTYPE BY 1
247  0149          STEP PREV BY 1
248  0150          STEP PSTAT BY 1
249  0151          STEP PPASS BY 1
250  0152          NEXT ILOOP
251          ;
252          ;     END DINs
253          ;
254          ;     BEGIN ANALOG RED AND YELLOWS:
255          ;
256  0153          LET STRNGO = "RUN"
257  0154          ASSIGN PAI = 'PI-CA1-11'
258  0155          ASSIGN PSKID = SKID4
259  0156          ASSIGN PTYPE = TYPE4
260  0157          FOR JLOOP = 4 TO 34
261          ;     IF !SELECT() THEN GOTO AGAIN
262  0158          IF (PTYPE) THEN                                       ; If some alarm exists
263                   BEGIN                                  ;be1
264  0159             BITVAL = PAI:AI_ALMCT
265  0160             IF !AND(BITVAL,16) THEN                            ; If not inhibited
266                      BEGIN                               ;be2
267  0161                IF AND(BITVAL,15) THEN                          ; If alarm exists and is current
268                          BEGIN                           ;be3
269  0163                   IF AND(BITVAL,32)THEN NXT = 2                ; If alarm is ack
270                          ELSE
271                             BEGIN                        ;be4
272  0164                       IF PSKID !=BITVAL THEN
273                                BEGIN                    ;be5
274  0166                          IF BITVAL < 3 THEN FALMON = 1
275  0167                          LET PSKID = BITVAL
```

```
276  0168                          LET PDEV:DEV_TMP0 = 2
277  0169                          NXT = 1
278                                END                              ;en5
279  0170                       ELSE NXT = 1
280                             END                              ;en4
281                          END                                 ;en3
282                       ELSE                                          ; Else there is no alarm
283                          BEGIN
284  0171                    NXT = 0
285  0172                    LET PSKID = 0
286                          END
287                       END                                    ;en2
288                    ELSE                                          ; Else alarm is inhibited
289                       BEGIN
290  0173                 NXT = 0
291  0174                 LET PSKID = 0
292                       END
293  0175                 ANALOG = OR(ANALOG,NXT)
294                       END                                    ;en1
295  0176              STEP PSKID BY 1
296  0177              STEP PTYPE BY 1
297  0178              STEP PAI BY 1
298  0179              NEXT JLOOP
299            ;
300            ;        END ANALOG RED AND YELLOWS
301            ;
302            ;        BEGIN ANALOG WHITE ALARMS:
303            ;
304  0180      WHT:    WHITE = 0
305  0181              ASSIGN PALARM = AI2
306  0182              ASSIGN PREV = FRV37
307  0183              ASSIGN PSTAT = INT37
308  0184              ASSIGN PTYPE = TYPE37
309  0185              ASSIGN PPASS = FPAS37
310            ;        ILOOP = 35                        ; Delete White alarms on DDI-CA1-1
311            ;        ASSIGN PAI = 'DDI-CA1-1'
312            ;        GOSUB ANLG                                ; DDI-CA1-2 LOLO        35
313            ;        ILOOP = 36
314            ;        GOSUB ANLG                                ; DDI-CA1-2 HIHI        36
315  0186              ILOOP = 37
316  0187              ASSIGN PAI = 'PDI-FH1-1'
317  0188              GOSUB ANLG                                ; PDI-FH1-1 HI          37
318  0189              ILOOP = 38
319  0190              ASSIGN PAI = 'PDI-FH2-1'
320  0191              GOSUB ANLG                                ; PDI-FH2-1 HI          38
321  0192              ILOOP = 39
322  0193              ASSIGN PAI = 'PDI-FH3-1'
323  0194              GOSUB ANLG                                ; PDI-FH3-1 HI          39
324            ;
325            ;        Deleted White alarms on DDI-CA1-1:
326            ;
327            ;        IF INT35 & INT35 < 3 THEN BLET LEAK0 = INT35
328            ;          ELSE IF INT36 & INT36 < 3 THEN BLET LEAK0 = INT36 + 6
329            ;            ELSE IF INT35 = 3 THEN BLET LEAK0 = 9
330            ;              ELSE BLET LEAK0 = 0
```

```
331             ;
332             ;       END ANALOG WHITE ALARMS
333             ;
334     0195            LET FTRIP4 = 0
335     0196            MARK STEP "BELOW",0,0
336     0198            IF CURENT > 3 THEN PRINT ADI," LAMP VARIABLE IS GREATER THAN 3!"
337     0199            CURENT = OR(LAST,WHITE)
338     0200            CURENT = OR(CURENT,ANALOG)
339     0202            IF AND(CURENT,1)THEN LMPVAR = 2
340     0204               ELSE IF AND(CURENT,2)THEN LMPVAR = 1
341     0205                       ELSE LMPVAR = 0
342     0206            LAMP ADI,SKDKEY,LMPVAR
343     0207            MARK STEP "BOTTOM",0,0
344     0208            NOCRIT
345             ;
346             ;       BEGIN GRAPHIC SPECIALS:
347             ;
348     0209    WF1:    VL1 = 'LI-CA1-1G':AI_MEAS
349     0210            VL2 = 'LI-CA1-2G':AI_MEAS
350     0211            ASSIGN PSPAR0 = VL1
351     0213            IF AND('SELECT-WF':DEV_STAT,4)THEN STEP PSPAR0 BY 1
352     0214            IF PSPAR0 > 19860 THEN
353                        BEGIN
354     0216               IF PSPAR0 < 20472 THEN WF1 = ((PSPAR0-19860)/612) * 7
355     0218                  ELSE IF PSPAR0 < 21780 THEN WF1=((PSPAR0-20472)/1308)*13 + 7
356     0220                   ELSE IF PSPAR0 < 23088 THEN WF1=((PSPAR0-21780)/2308)*10+ 20
357     0222                    ELSE IF PSPAR0 <= 30000 THEN WF1=((PSPAR0-23088)/6912)*70 + 30
358                        END
359     0223            ELSE WF1 = 0
360             ;
361     0224    WF3:    VL3 = 'LI-CA1-3G':AI_MEAS
362     0225            IF VL3 > 760 THEN
363                        BEGIN
364     0227,              IF VL3 < 8013 THEN WF3 = ((VL3-760)/7253) * 19
365     0229                  ELSE IF VL3 < 11377 THEN WF3 = ((VL3- 8013)/3364)*29 + 19
366     0231                   ELSE IF VL3 < 19815 THEN WF3 = ((VL3-11377)/8438)*18 + 48
367     0233                    ELSE IF VL3 < 20472 THEN WF3 = ((VL3-19860)/612)*2 + 66
368     0235                     ELSE IF VL3 < 21780 THEN WF3 = ((VL3-20472)/1308)*3 + 68
369     0237                      ELSE IF VL3 < 23088 THEN WF3 = ((VL3-21780)/1308)*3 + 71
370     0239                       ELSE IF VL3 <= 30560 THEN WF3 = ((VL3-23088)/7472)*26 + 74
371                        END
372             ;
373             ;       END GRAPHIC SPECIALS
374             ;
375             ;       Devices To Lock In Manual:
376             ;
377     0241            IF AND('CA1-ALARM':DEV_FLG,64)THEN PUTMANL 'CA1-ALARM'
378     0243            IF AND('SELECT-WF':DEV_FLG,64)THEN PUTMANL 'SELECT-WF'
379     0245            IF AND('SELECT-DI':DEV_FLG,64)THEN PUTMANL 'SELECT-DI'
380     0247            IF AND('HV-PDSPRY':DEV_FLG,64)THEN PUTMANL 'HV-PDSPRY'
381             ;
382     0248            GOTO AGAIN
383             ;
384             ;       E N D   P R O G R A M   L O O P
385             ;
```

```
386           ;       BEGIN ANALOG WHITE ALARM SUBROUTINE:
387           ;
388           ; ANLG:  IF !SELECT() THEN GOTO AGAIN
389    0249   ANLG:    IF (PTYPE = 1) THEN                          ; If some alarm
390                       BEGIN                                ;be1
391    0250               BITVAL = PAI:AI_ALMCT
392    0251               IF !(AND(BITVAL,16)) THEN                ; If not inhibited
393                         BEGIN                              ;be2
394    0253                 IF PSTAT = 3 THEN LET PSTAT = 0
395    0254                 IF (PREV) THEN                     ; If rev acting, LOLO, or LO
396                            BEGIN                           ;be3
397    0255                    IF (PSTAT = 0) THEN             ; If not previously alarmed
398                               BEGIN                        ;be4
399    0257                       IF (PAI:AI_MEAS <= PALARM) THEN GOSUB ALM
400                               END                          ;en4
401                            ELSE
402                               BEGIN                        ;be5
403    0258                       IF (PSTAT = 1) THEN          ; If previously alarmed
404                                  BEGIN                     ;be6
405    0260                          IF (FTRIP4) THEN GOSUB ACK1   ; If ACK flag set, then ack it
406    0263                          IF PTYPE !=1 THEN IF AND(PAI:AI_ALMCT,32) THEN GOSUB ACK1
407                                  END                       ;en6
408    0265                       IF (PAI:AI_MEAS >= PALARM + PAI:AI_DEAD) THEN GOSUB CLR
409                               END                          ;en5
410                            END                             ;en3
411                         ELSE                               ; If fail-safe, HIHI, or HI
412                            BEGIN                           ;be10
413    0266                    IF (PSTAT = 0) THEN
414                               BEGIN                        ;be11
415    0268                       IF (PAI:AI_MEAS >= PALARM) THEN GOSUB ALM
416                               END                          ;en11
417                            ELSE
418                               BEGIN                        ;be12
419    0269                       IF (PSTAT = 1) THEN
420                                  BEGIN                     ;be13
421    0271                          IF (FTRIP4) THEN GOSUB ACK1
422    0274                          IF PTYPE != 1 THEN IF AND(PAI:AI_ALMCT,32) THEN GOSUB ACK1
423                                  END                       ;en13
424    0276                       IF (PAI:AI_MEAS <= PALARM - PAI:AI_DEAD) THEN GOSUB CLR
425                               END                          ;en12
426                            END                             ;en10
427    0277                 NXT = PSTAT
428    0279                 IF (PPASS) & (PSTAT = 2) THEN NXT = 0
429                         END                                ;en2
430                         ELSE
431                           BEGIN
432    0280                   LET PSTAT = 3
433    0281                   NXT = 0
434                           END
435    0282               WHITE = OR(WHITE,NXT)
436                       END                                  ;en1
437    0283               ELSE PSTAT = 0
438    0284               STEP PALARM BY 1
439    0285               STEP PSTAT BY 1
440    0286               STEP PTYPE BY 1
```

```
441   0287          STEP PREV BY 1
442   0288          STEP PPASS BY 1
443   0289          RETURN
444           ;
445           ;       END ANALOG WHITE ALARM SUBROUTINE
446           ;
447           ;       BEGIN CLEAR, ALARM, ACKNOWLEDGE, MESSAGE, ERROR, AND TERMINATE SUBROUTINES:
448           ;
449   0290    CLR:    LET PSTAT = 0                                          ; Int set to zero
450                   ;
451                   ; If not white, then unack:
452                   ;
453   0293            IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
454   0294            LET FCLEAR = 1                                         ; Clear flag
455   0295            GOSUB ACKK                                             ; Print message
456   0296            RETURN
457           ;
458   0297    ALM:    LET PSTAT = 1                                          ; Int set to one
459                   ;
460                   ; If not white, then unack:
461                   ;
462   0300            IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
463   0301            LET PDEV:DEV_TMP0 = 2
464                   ;
465                   ; Turn audible alarm on if white or yellow:
466                   ;
467   0303            IF AND(PTYPE,3) THEN FALMON = 1
468   0304            CRIT
469   0305            LAMP ADI,SKDKEY,2                                      ; Blink the LAMP
470   0306            GOSUB ACKK                                             ; Print message
471   0307            RETURN
472           ;
473   0308    ACK1:   LET PSTAT = 2                                          ; Ack the alarm
474   0309            LET FACK = 1                                           ; Set ACK flag for C program
475           ;
476           ;       ALARM TYPER STRINGS:
477           ;
478   0311    ACKK:   IF ILOOP = 0 THEN STRNG0 = "CA1 VESSEL PRESSURE HI"
479   0313            IF ILOOP = 1 THEN STRNG0 = "CA1-1 WEIGHT FACTOR HI"
480   0315            IF ILOOP = 2 THEN STRNG0 = "CA1-2 WEIGHT FACTOR HI"
481   0317            IF ILOOP = 3 THEN STRNG0 = "LOWER DE-ENTRNMNT PAD DP HI"
482           ;       IF ILOOP = 35 THEN STRNG0 = "(CA1-1)-(CA1-2) SPG DIFF LOLO"      ; Delete White on DDI-CA1-1
483           ;       IF ILOOP = 36 THEN STRNG0 = "(CA1-1)-(CA1-2) SPG DIFF HIHI"      ; Delete White on DDI-CA1-1
484   0319            IF ILOOP = 37 THEN STRNG0 = "F-H-1 RW FILTER DELTA P HIGH"
485   0321            IF ILOOP = 38 THEN STRNG0 = "F-H-2 RW FILTER DELTA P HIGH"
486   0323            IF ILOOP = 39 THEN STRNG0 = "F-H-3 RW FILTER DELTA P HIGH"
487   0325            IF ILOOP = 17 THEN STRNG0 = "START CA1BATCH PGM "
488   0327            IF ILOOP = 18 THEN STRNG0 = "CA1BATCH WRONG UNIT"
489   0329            IF ILOOP = 19 THEN STRNG0 = "TERMINATE CA1BATCH "
490           ;
491           ; If acking the alarm, then:
492           ;
493   0330    AMSG:   IF PTYPE = 1 THEN
494                     BEGIN
495                     ;
```

```
496                     ; If white alarm, print:
497                     ;
498    0332             IF (FACK = 0) & (FCLEAR = 0) & (FSTART = 0) THEN PRINT ADI,"WHITE ALARM %p",STRNG0
499    0333             IF (FACK) THEN
500                       BEGIN
501                       ;
502                       ; If white, print it is ack and reset the flag:
503                       ;
504    0334               PRINT ADI,"%p ALARM ACKNOWLEDGE",STRNG0
505    0335               LET FACK = 0
506                       END
507                     END
508                   ;
509                   ; If Clear, print it is clear and reset the flag:
510                   ;
511    0336           IF (FCLEAR) THEN
512                     BEGIN
513    0337             PRINT ADI,"%p ALARM CLEAR",STRNG0
514    0338             LET FCLEAR = 0
515                     END
516                   ;
517                   ; If Startup, then print startup message:
518                   ;
519    0339           IF (FSTART) THEN
520                     BEGIN
521    0340             LET FSTART = 0
522    0341             PRINT ADI,"%p",STRNG0
523                     END
524    0342           RETURN
525           ;
526           ; If Error - print message, start timer, and set flag:
527           ;
528    0343   ERRSUB:  IF ERRN != ERRNUM() THEN
529                     BEGIN
530    0344             TIMERON STIME0
531    0345             TIMERCLR STIME0
532    0346             LET FSN0 = 1
533    0347             LET ERRN = ERRNUM()
534    0348             LET ERRL = ERRLIN()
535    0349             PRINT ADI,"ERROR # %i AT LINE %i",errn,errl
536                     END
537    0350             ERSCLR
538    0351           GOTO AGAIN
539           ;
540           ; If FABORT = 1, then abort program:
541           ;
542           ; TERM:  LET FABORT = 0
543    0352   TERM:    LET STRNG0 = "CA1BATCH TERMINATED    "
544    0353           LET ILOOP = STAR + 2
545    0354           LET FSTART = 1
546    0355           LET STRNG0 = "TERM"
547    0356           GOSUB ACKK
548                   END

      Total Errors Detected      0     Total Warnings      0
          Sequence Program Size = 6960 Bytes, 435 Pages
```

21 Forward Reference(s)

PB1BATCH

Sequence Instruction Compiler Listing 15:35:33 24-Feb-95 Page 1

```
;       P B 1 B A T C H . S E Q                                  VERSION 3
;
;       Function:       Sequence program for Lamp function,
;                       message display on Graphic number 012,
;                       and printer messages.
;
;       Written By      : R. N. Kersey
;       Date            : 05-31-92
;       Facility        : 242-A Evaporator Bldg.
;       Machine         : VAX 4104-A
;       Op. System      : VAX/VMS 5.5-2AH
;       D/3 Version     : 6.3-2 BASELINE
;       Compiler        : SIC
;       Build command   : @[320,20]PB1BATCH
;
;       UNIT = 7, SKDKEY = 30
;
;       Assoc. UPF:     [300,1]_PRCESS.___      PRCESS.LST
;       Assoc. RPF:     [300,1]_242A.___        242A.LST
;       SDB:            [300,1]PCM0A.SDB        PCM0ASDB.CMD
;
;       POINTER   SDB VAR.      FUNCTION
;       -------   ---------     ------------------------------------------
;       PSTAT     INTn          Current EPN alarm state is an integer tied to
;                               graphic flashing string of value:
;       .         (n=0-100)             0 = previously not alarmed
;       .                               1 = previously alarmed, not ack
;       .                               2 = previously alarmed, is ack
;       .                               3 = alarm is inhibited
;       .                               4 = input is off scan
;       .                               5 = input is bad
;       .         LEAKn         Special INT variable for multiple white alarms on an EPN
;       PDIN      'EPN'         ---
;       PTYPE     TYPEn         alarm type 0 = none, 1 = white, 2 = yellow,
;                                  3 = yellow and yellow, 4 = red,
;                                  6 = yellow and red
;       PREV      FRVn          Reverse acting (not fail safe) flag (1=set)
;       PSKID     SKIDn         Used for analog alarms
;       PPASS     FPASn         Turn lamp off when alarm is ACKnowledged
;       PALARM    FALMn         Analog alarm flag
;       ---       FNING         Unused error flag
;
; MODIFICATION LOG:
;
;       30-DEC-92       Fixed white alarm print message.        M. Teats
;
;       07-JUN-93       Added comments and baselined for        M. Teats
;                       D/3 version 3.3.  CCN# 93JUN01-01.
;       28-JUL-94       Added white alarm for PDI-CA1-3                 J. Berger
;                       per EE-94-00019.
;
;       15-DEC-94       Revised header to indicate the new VMS          J. Berger
;                       system. Rebaselined docmument. Removed
;                       obsolite Select() statement.
;
```

```
56          ;  END MOD LOG
57          ;
58          UNIT    PB1
59          ;
60                  COMMON PB2
61    0000          NOP
62    0001          STAR = 24                       ; Start message
63    0002          LET SKDKEY = 30          ; SKID Key
64    0003          LET UNITT = 7                   ; SEQ Unit number
65    0004          ASSIGN PDEV = 'PB1-ALARM'       ; Alarm device
66    0005          ASSIGN PTYPE = TYPE24
67    0006          LET ADI = 0                     ; Alarm destination index
68    0007          LET FSN0 = 0                    ; No alarm yet
69    0008          TIMEROFF STIME0          ; Alarm timer off
70    0009          TIMERCLR STIME0          ; Alarm timer clear
71    0010          STRNG0 = " "                    ; Blank 30 character string
72    0011          LET ERRN = 0                    ; Set error number to zero
73    0012          LET ERRL = 0                    ; Set error line to zero
74    0013          LET FTRIP4 = 0                  ; FTRIP4 = 1 when audible horn is to be turned on
75    0014          GETUNIT TRIP                    ; Which sequence unit is program running in
76    0015          IF TRIP != UNITT THEN           ; Abort if running in wrong seq. unit
77                     BEGIN
78    0016             FSTART = 1
79    0017             ILOOP = STAR + 1
80    0018             PTYPE = 0
81    0019             GOSUB ACKK
82    0020             GOTO TERM
83                     END
84          ;       IF !SELECT() THEN GOTO AGAIN
85          ;
86          ;       B E G I N   I N I T I A L I Z I N G
87          ;
88    0021          LET STRNG0 = "INIT"
89    0022    INIT: LET FABORT = 0                  ; If FABORT = 1 then abort program
90    0023          LET FSTART = 0                  ; Program start flag = 0
91    0024          LAMP ADI,SKDKEY,0               ; Turn off the annunciator key
92    0025          LET LSG0 = " "                  ; Clear the string variables
93    0026          LET LSG1 = " "
94    0027          LET LSG2 = " "
95    0028          LET LSG3 = " "
96    0029          LET LSG4 = " "
97    0030          LET PDEV:DEV_TMP0 = 1                          ; Set alarm device to non-alarm state
98    0031          IF AND(PDEV:DEV_ALMCT,16) THEN                 ; If the alarm device is inhibited, remove the inhibit
99    0032            PDEV:DEV_ALMCT=XOR(PDEV:DEV_ALMCT,16)
100   0033          IF AND(PDEV:DEV_ALMCT,32) THEN                 ; If the alarm device is ack, clear it
101   0034             PDEV:DEV_ALMCT = XOR(PDEV:DEV_ALMCT,32)
102         ;
103         ;       First, set all flags and integers:
104         ;
105   0035          ASSIGN PDIN = 'YS-PC-105'
106   0036          ASSIGN PPASS = FPAS0
107   0037          ASSIGN PREV = FRV0
108   0038          ASSIGN PSTAT = INT0
109   0039          ASSIGN PSKID = SKID0
110   0040          ASSIGN PTYPE = TYPE0
```

```
111             ;
112     0041            FOR ILOOP = 0 TO 25
113     0042            LET PPASS = 0
114     0043            LET PREV = 0
115     0045            IF PSTAT > 5 THEN LET PSTAT = 0          ; Reset current status if undefined
116     0046            LET PSKID = 0                                    ; Default previous status to no alarms
117     0047            LET PTYPE = 0                                    ; Default to no alarm
118     0048            IF ILOOP < 10 THEN                               ; For DINs
119                        BEGIN
120     0049               LET PTYPE = 0
121     0050               IF AND(PDIN:DIN_DFLG,3) THEN                  ; Check for alarms if defined (must be red or yellow)
122                           BEGIN
123     0051                  LET PTYPE = 2                              ; Default to yellow
124     0053                  IF AND(PDIN:DIN_DFLG,1) THEN LET PREV = 1  ; Reverse acting
125     0055                  IF AND(PDIN:DIN_DFLG,4) THEN LET PTYPE = 4 ; Set to red critical alarm
126                           END
127                        END
128     0056            ELSE LET PTYPE = 4                               ; For AIs, default to red
129     0058            IF ILOOP < 9 THEN STEP PDIN BY 1         ; Point to next DIN
130     0059            STEP PPASS BY 1
131     0060            STEP PREV BY 1
132     0061            STEP PSTAT BY 1
133     0062            STEP PSKID BY 1
134     0063            STEP PTYPE BY 1
135     0064            NEXT ILOOP
136             ;
137     0065            LET FRV2 = 1
138     0066            LET FRV4 = 1
139     0067            LET FRV5 = 0
140     0068.           LET FRV30 = 0
141             ;
142     0069            LET TYPE10 = 0
143     0070            LET TYPE11 = 0
144     0071            LET TYPE18 = 0
145     0072            LET TYPE22 = 0
146     0073            LET TYPE24 = 0
147     0074            LET TYPE28 = 2
148     0075            LET TYPE30 = 1
149     0076            LET INT28 = 0
150     0077            LET AI30 = 15
151             ;
152             ;       E N D   I N I T I A L I Z I N G
153             ;
154             ;       S E N D.  P R O G R A M   S T A R T   M E S S A G E
155             ;
156     0078            ILOOP = STAR
157     0079            FSTART = 1
158     0080            PTYPE = 0
159     0081            CURENT = 0
160     0082            LAST = 0
161     0083            ANALOG = 0
162     0084            WHITE = 0
163     0085            GOSUB ACKK
164     0086            ON ERROR GOSUB ERRSUB
165             ;
```

```
166          ;       B E G I N   P R O G R A M   L O O P
167          ;
168    0087  AGAIN:  MARK STEP "WTUNTL",0,0
169          ;       WAITUNTIL SELECT() | FABORT       ; Wait until micro is selected or program aborted
170          ;       IF (FABORT) THEN GOTO TERM        ; Terminate program if abort flag is set
171    0088          MARK STEP "RNNING",0,0
172    0089          WAIT 1
173    0090          IF (FSN0) THEN                    ; If alarm flag set
174    0091             IF STIME0 > 60 THEN            ; If alarm flag active for 1 minute
175                         BEGIN                      ; Reset timer and turn off flag
176    0092                 TIMEROFF STIME0
177    0093                 TIMERCLR STIME0
178    0094                 LET FSN0 = 0
179    0095                 LET ERRN = 0
180    0096                 LET ERRL = 0
181                         END
182          ;
183          ;       If PB1-ALARM $T1 is alarmed and it's ALMCT indicates alarms are acknowledged:
184          ;
185    0097          IF (PDEV:DEV_TMP0 = 2) & AND(PDEV:DEV_ALMCT,32) THEN
186                   BEGIN
187    0098           LET PDEV:DEV_TMP0 = 1 ; Put PB1-ALARM in ack state
188    0099           LET FTRIP4 = 1                   ; Set for Unit alarm ack, system to graphic
189    0100           LET ACK = 1                      ; Set for Unit alarm ack, graphic to system
190    0101           CRIT
191    0102           LAMP ADI,SKDKEY,1                ; Turn LAMP on
192                   END
193          ;
194          ;       BEGIN DINs:
195          ;
196    0103          ASSIGN PDIN = 'YS-PC-105'                ; Assign pointer to first DIN
197    0104          ASSIGN PTYPE = TYPE0
198    0105          ASSIGN PREV = FRV0
199    0106          ASSIGN PSTAT = INT0
200    0107          ASSIGN PPASS = FPAS0
201    0108          LAST = 0
202    0109          NXT = 0
203    0110          LMPVAR = 0
204    0111          BITVAL = 0
205          ;
206    0112  LOP:    FOR ILOOP = 0 TO 9
207          ;       IF !SELECT() THEN GOTO AGAIN
208    0113          IF ( PTYPE > 0 ) THEN                                    ; If some type of alarm
209                   BEGIN                                                ;be1
210    0115           IF AND(PDIN:DIN_ALMCT,16) THEN LET PSTAT = 3        ; If EPN inhibited
211    0117           IF AND(PDIN:DIN_ALMCT,4) THEN LET PSTAT = 5         ; If EPN is bad input
212    0119           IF !AND(PDIN:DIN_FLG,8) THEN LET PSTAT = 4          ; If EPN is disabled
213    0120           IF ( PSTAT < 3 ) THEN                               ; If not inhibited
214                       BEGIN                                            ;be2
215    0121                IF (PSTAT = 1) THEN
216                           BEGIN                                        ;be3
217    0122                   IF (PTYPE != 1) THEN
218    0124                      IF AND(PDIN:DIN_ALMCT,32) THEN GOSUB ACK1
219    0126                   IF (FTRIP4) THEN GOSUB ACK1
220                           END                                          ;en3
```

```
221  0129                  IF PSTAT = 0 THEN IF(PDIN:DIN_VAL = PREV) THEN GOSUB ALM
222  0130                  IF (PSTAT) THEN
223                              BEGIN                                    ;be4
224  0132                        IF (PDIN:DIN_VAL != PREV) THEN GOSUB CLR
225  0133                        NXT = PSTAT
226  0135                        IF (PPASS) & (PSTAT = 2) THEN NXT = 0
227  0136                        LAST = OR(LAST,NXT)
228                              END                                      ;en4
229                         END                                           ;en2
230  0138                ELSE IF !AND(PDIN:DIN_ALMCT,16)&!AND(PDIN:DIN_ALMCT,4)&AND(PDIN:DIN_FLG,8)THEN LET PSTAT = 0
231                      END                                             ;en1
232  0140           IF ILOOP < 9 THEN STEP PDIN BY 1
233  0141           STEP PTYPE BY 1
234  0142           STEP PREV BY 1
235  0143           STEP PSTAT BY 1
236  0144           STEP PPASS BY 1
237  0145           NEXT ILOOP
238           ;
239           ;     END DINs:
240           ;
241           ;     BEGIN ANALOG RED AND YELLOWS:
242           ;
243  0146           ANALOG = 0
244  0147           ASSIGN PAI = 'KY-PB1-1'
245  0148           ASSIGN PSKID = SKID0
246  0149           FOR JLOOP = 0 TO 14
247           ;     IF !SELECT() THEN GOTO AGAIN
248  0150           IF (PTYPE) THEN                                       ; If some alarm exists
249                    BEGIN                                            ;be1
250  0151              BITVAL = PAI:AI_ALMCT
251  0152              IF !AND(BITVAL,16) THEN                           ; If not inhibited
252                       BEGIN                                         ;be2
253  0153                 IF AND(BITVAL,15) THEN                        ; If alarm exists and is current
254                          BEGIN                                     ;be3
255  0155                    IF AND(BITVAL,32) THEN NXT = 2             ; If alarm is ack
256                          ELSE                                       ; If alarm is unack
257                             BEGIN                                  ;be4
258  0156                       IF PSKID !=BITVAL THEN
259                                BEGIN                               ;be5
260  0158                             IF BITVAL < 3 THEN FALMON = 1
261  0159                             LET PSKID = BITVAL
262  0160                             LET PDEV:DEV_TMP0 = 2
263  0161                             NXT = 1
264                                   END                              ;en5
265  0162                       ELSE NXT = 1
266                             END                                    ;en4
267                          END                                       ;en3
268                       ELSE                                         ; Else there is no alarm
269                          BEGIN
270  0163                    NXT = 0
271  0164                    LET PSKID = 0
272                          END
273                       END                                          ;en2
274                    ELSE                                            ; Else alarm is inhibited
275                       BEGIN
```

```
276  0165                NXT = 0
277  0166                LET PSKID = 0
278                      END
279  0167              ANALOG = OR(ANALOG,NXT)
280                    END                                          ;en1
281  0168           STEP PSKID BY 1
282  0169           STEP PTYPE BY 1
283  0170           STEP PAI BY 1
284  0171           STEP PSTAT BY 1
285  0172           NEXT JLOOP
286         ;
287         ;       END ANALOG RED AND YELLOWS
288         ;
289         ;       BEGIN DEVICE ALARMS:
290         ;
291  0173   JGV:    ILOOP = 28                                               ; JGV-SUMP
292  0174           ASSIGN PSTAT  = INT28
293  0175           ASSIGN PTYPE = TYPE28
294         ;       IF !SELECT() THEN GOTO AGAIN
295  0176           IF (TYPE28) THEN                                         ; If some type of alarm
296                    BEGIN                                          ;be1
297  0178              IF AND('JGV-SUMP':DEV_ALMCT,16) THEN PSTAT = 3 ; If EPN inhibited
298  0179              IF (PSTAT < 3) THEN                                   ; If not inhibited
299                         BEGIN                                     ;be2
300  0180                   IF (PSTAT = 1) THEN
301                            BEGIN                                  ;be3
302  0181                      IF (PTYPE != 1) THEN
303  0183                         IF AND('JGV-SUMP':DEV_ALMCT,32) THEN GOSUB ACK1
304  0185                      IF (FTRIP4) THEN GOSUB ACK1
305                            END                                    ;en3
306                        ;
307                        ; If alarm (ON 8+ MIN) is active and is previously not alarmed:
308                        ;
309  0187                  IF PSTAT = 0 THEN IF AND('JGV-SUMP':DEV_STAT,8) THEN
310                            BEGIN                                  ;be4
311  0188                      LET PSTAT = 1
312  0191                      IF PTYPE != 1 THEN IF AND('JGV-SUMP':DEV_ALMCT,32)THEN 'JGV-SUMP':DEV_ALMCT = XOR('JGV-SUMP':DEV_ALMCT,32)
313  0192                      LET PDEV:DEV_TMP0 = 2
314  0194                      IF AND(PTYPE,3) THEN FALMON = 1
315  0195                      CRIT
316  0196                      LAMP ADI,SKDKEY,2
317                            END                                    ;en4
318  0197                   IF (PSTAT) THEN                                   ; If previously alarmed
319                            BEGIN                                  ;be5
320  0198                      IF !AND('JGV-SUMP':DEV_STAT,8) THEN            ; If not alarming now, then clear
321                                 BEGIN                             ;be6
322  0199                           PSTAT = 0
323  0200                           GOSUB ACKK
324                                 END                               ;en6
325  0201                      NXT = PSTAT
326  0203                      IF (PPASS) & (PSTAT = 2 ) THEN NXT = 0
327  0204                      LAST = OR(LAST,NXT)
328                            END                                    ;en5
329                         END                                       ;en2
330  0206                ELSE IF !AND('JGV-SUMP':DEV_ALMCT,16) THEN PSTAT = 0
```

```
331                       END                                          ;en1
332           ;
333           ;       END DEVICE ALARMS
334           ;
335           ;       BEGIN ANALOG WHITE ALARMS:
336           ;
337           ;
338     0207  WHT:     WHITE = 0
339     0208          ASSIGN PALARM = AI30
340     0209          ASSIGN PREV = FRV30
341     0210          ASSIGN PSTAT = INT30
342     0211          ASSIGN PTYPE = TYPE30
343     0212          ASSIGN PPASS = FPAS30
344     0213          ILOOP = 30
345     0214          ASSIGN PAI = 'PDI-CA1-3'
346           ;       GOSUB ANLG                          ; PDICA1-3      30
347           ;
348           ;       END ANALOG WHITE ALARMS
349           ;
350           ;       E N D   P R O G R A M   L O O P
351           ;
352           ;       BEGING ANALOG WHITE ALARMS SUBROUTINE:
353           ;
354           ; ANLG:  IF !SELECT() THEN GOTO AGAIN
355     0215  ANLG:    IF (PTYPE = 1) THEN                  ; IF SOME ALARM
356                       BEGIN                       ;BE1
357     0216              BITVAL = PAI:AI_ALMCT
358     0217              IF !(AND(BITVAL,16)) THEN         ; IF NOT INHIBITED
359                            BEGIN                  ;BE2
360     0219                   IF PSTAT = 3 THEN LET PSTAT = 0
361     0220                   IF (PREV) THEN             ; IF REV ACTING, LOLO, OR LO
362                               BEGIN               ;BE3
363     0221                      IF (PSTAT = 0) THEN     ; IF NOT PREVIOUSLY ALARMED
364                                    BEGIN          ;BE4
365     0223                      IF (PAI:AI_MEAS <= PALARM) THEN GOSUB ALM
366                                    END
367                               ELSE
368                                    BEGIN          ;BE5
369     0224                           IF (PSTAT = 1) THEN  ; IF PREVIOUSLY ALARMED
370                                       BEGIN       ;BE6
371     0226                              IF (FTRIP4) THEN GOSUB ACK1   ;IF ACK FLAG SET, THEN ACK IT
372     0229                              IF PTYPE !=1 THEN IF AND(PAI:AI_ALMCT,32) THEN GOSUB ACK1
373                                       END         ;EN5
374     0231                           IF (PAI:AI_MEAS >= PALARM + PAI:AI_DEAD) THEN GOSUB CLR
375                                    END            ;EN3
376                                    END
377                               ELSE                    ; IF FAIL-SAFE, HIHI, OR HI
378                                    BEGIN          ;BE10
379     0232                           IF (PSTAT = 0) THEN
380                                       BEGIN       ;BE11
381     0234                              IF (PAI:AI_MEAS >= PALARM) THEN GOSUB ALM
382                                       END         ;EN11
383                                ELSE
384                                    BEGIN          ;BE12
385     0235                           IF (PSTAT = 1) THEN
```

```
386                                   BEGIN          ;BE13
387    0237                           IF (FTRIP4) THEN GOSUB ACK1
388    0240                           IF PTYPE != 1 THEN IF AND(PAI:AI_ALMCT,32) THEN GOSUB ACK1
389                                   END            ;EN13
390    0242                        IF (PAI:AI_MEAS <= PALARM - PAI:AI_DEAD) THEN GOSUB CLR
391                                 END              ;EN12
392                                   END            ;EN10
393    0243                         NXT = PSTAT
394    0245                         IF (PPASS) & (PSTAT = 2) THEN NXT = 0
395                                 END              ;EN2
396                                 ELSE
397                                   BEGIN
398    0246                           LET PSTAT = 3
399    0247                           NXT = 0
400                                   END
401    0248                 WHITE = OR(WHITE,NXT)
402                         END                      ;EN1
403    0249                 ELSE PSTAT = 0
404    0250                 STEP PALARM BY 1
405    0251                 STEP PSTAT BY 1
406    0252                 STEP PTYPE BY 1
407    0253                 STEP PREV BY 1
408    0254                 STEP PPASS BY 1
409            ;            RETURN
410            ;
411            ;       END ANALOG WHITE ALARM SUBROUTINE
412            ;
413    0255            FTRIP4 = 0
414    0257            IF LAST > 3 THEN PRINT ADI," LAMP VARIABLE IS GREATER THAN 3!"
415    0258            CURENT = OR(LAST,ANALOG)
416    0259            CURENT = OR(CURENT,WHITE)
417    0261            IF AND(CURENT,1) THEN LMPVAR = 2
418    0263               ELSE IF AND(CURENT,2) THEN LMPVAR = 1
419    0264                       ELSE LMPVAR = 0
420    0265            LAMP ADI,SKDKEY,LMPVAR
421    0266            MARK STEP "BOTTOM",0,0
422    0267            NOCRIT
423            ;
424            ;       BEGIN LOGIC FOR GRAPHICS:
425            ;
426            ;       SA0 = AND('PB-1':DEV_STAT,1)     ; CF-OFF
427            ;       SA1 = AND('PB-1':DEV_STAT,2)     ; CF-ON
428    0268            SA2 = AND('PB-1':DEV_STAT,4)     ; SD-RESET
429            ;       SA3 = AND('PB-1':DEV_STAT,8)     ; CF-JOGNG
430    0269            SA4 = AND('PB-1':DEV_STAT,16)    ; SHUTDOWN
431    0270            SA5 = AND('PB-1':DEV_STAT,32)    ; INTERLOCK
432            ;       SA6 = AND('PB-1':DEV_STAT,64)    ; STOPPING
433            ;       SA7 = AND('PB-1':DEV_STAT,128)   ; STARTING
434            ;
435    0272            IF AND('PB1-ALARM':DEV_FLG,64) THEN PUTMANL 'PB1-ALARM'
436    0274            IF AND('PB-1':DEV_FLG,64) THEN PUTMANL 'PB-1'
437    0276            IF AND('PB1WFINLK':DEV_FLG,64) THEN PUTMANL 'PB1WFINLK'
438    0278            IF AND('PB1-BYPAS':DEV_FLG,64) THEN PUTMANL 'PB1-BYPAS'
439    0280            IF AND('BOT-DUMP':DEV_FLG,64) THEN PUTMANL 'BOT-DUMP'
440    0282            IF AND('HV-SUMP-1':DEV_FLG,64) THEN PUTMANL 'HV-SUMP-1'
```

```
441   0284            IF AND('JGV-SUMP':DEV_FLG,64) THEN PUTMANL 'JGV-SUMP'
442         ;
443   0285            AT0 = AND('PB1WFINLK':DEV_TMP0,1)                ; LI-CA1-3
444   0286            AT1 = AND('PB1WFINLK':DEV_TMP0,2)                ; LI-CA1-1
445   0287            AT2 = AND('PB1WFINLK':DEV_TMP0,4)                ; LI-CA1-2
446   0288            AT3 = AND('PB1WFINLK':DEV_TMP0,8)                ; PB1WFINLK INTERLOCK
447   0289            AT4 = AND('PB1WFINLK':DEV_TMP0,16)               ; BYPAS OR NO INTERLOCK
448         ;         AT12= AND(AT0,AT4)                               ; LI-CA1-3 CONTROLS PB1WFINLK
449   0290            AT5 = AND('PB1-BYPAS':DEV_TMP0,32)               ; timer running
450   0291            AT6 = AND('PB1-BYPAS':DEV_TMP0,64)               ; time out
451   0292            AT8 = AND('PB1-BYPAS':DEV_TMP0,256)              ; PERMISSIVE ON
452   0293            AT23 = AND('PB-1':DEV_TMP1,128)
453   0294            AT24 = AND('PB-1':DEV_TMP1,256)
454   0295            AT25 = AND('PB-1':DEV_TMP1,512)
455   0296            AT31= AND('PB-1':DEV_TMP1,32768)          ; PB-1 INTERLOCKS
456         ;
457   0297            BT26 = AND('P-C-105':DEV_TMP1,1024)
458   0298            BT27 = AND('P-C-105':DEV_TMP1,2048)
459   0299            BT28 = AND('P-C-105':DEV_TMP1,4096)
460   0300            BT29 = AND('P-C-105':DEV_TMP1,8192)
461   0301            BT30 = AND('P-C-105':DEV_TMP1,16384)
462         ;
463   0302            CT0 = 'YS-PB1-1':DIN_VAL
464   0303            CT1 = AND('PB-1':DEV_TMP0,2)
465         ;
466   0304            GT8 = AND(PB2GLOBAL:DEV_TMP0,256)
467   0305            LEAK0 = GT8
468         ;
469   0306            SB4 = AND('P-C-105':DEV_STAT,16)
470   0307            SB5 = AND('P-C-105A':DEV_STAT,16)
471         ;
472         ;       LEAK0 CONTROLS THE COLOR OF LINE BETWEEN LI-CA1-1/2 AND LI-CA1-3:
473         ;               0 = GREEN, 1 = RED, 2 - 7 = BLACK.  IF AT4 THEN LINE = GREEN
474         ;               ELSE IF !AT12 THEN LINE = RED, ELSE LINE = BLACK.
475         ;
476   0309            IF AT12 THEN LEAK0 = 0          ; WFs are OK or BYPASS ON and WF-3 not LO (GREEN)
477   0311               ELSE IF !AT0 THEN LEAK0 = 2  ; WF-3 LO                                  (BLACK)
478   0312                  ELSE LEAK0 = 1            ; INTERLOCK and WF-3 not LO               (RED)
479         ;
480         ;       END LOGIC FOR GRAPHICS
481         ;
482   0313            GOTO AGAIN
483         ;
484         ;       E N D.  P R O G R A M   L O O P
485         ;
486         ;       BEGIN CLEAR, ALARM, ACKNOWLEDGE, MESSAGE, ERROR, AND TERMINATE SUBROUTINES:
487         ;
488   0314  CLR:    LET PSTAT = 0                                    ; Int set to zero
489                 ;
490                 ; If not white, then unack:
491                 ;
492   0317          IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
493   0318          LET FCLEAR = 1                                   ; Clear flag
494   0319          GOSUB ACKK                                       ; Print message
495   0320          RETURN
```

```
496          ;
497    0321  ALM:    LET PSTAT = 1                                          ; Int set to one
498                  ;
499                  ; If not white, then unack:
500                  ;
501    0324          IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
502                  ;
503                  ; Turn audible alarm on if white or yellow:
504                  ;
505    0325          PDEV:DEV_TMPO = 2
506    0327          IF AND(PTYPE,3) THEN FALMON = 1
507    0328          CRIT
508    0329          LAMP ADI,SKDKEY,2
509    0330          GOSUB ACKK                                             ; Print message
510    0331          RETURN
511          ;
512    0332  ACK1:   LET PSTAT = 2                                          ; Ack the alarm
513    0333          LET FACK = 1                                           ; Set ACK flag for C program
514          ;
515          ;       ALARM TYPER STRINGS:
516          ;
517    0335  ACKK:   IF ILOOP = 0 THEN STRNGO = "P-C-105 SEAL WATER PUMP ON"
518    0337          IF ILOOP = 1 THEN STRNGO = "P-C-105A SEAL WATER PUMP ON"
519    0339          IF ILOOP = 2 THEN STRNGO = "CA1-7 POT DUMP VALVE OPEN"
520    0341         IF ILOOP = 3 THEN STRNGO = "P-B-1 PUMP MOTOR CONFIRMED"
521    0343          IF ILOOP = 4 THEN STRNGO = "CA1-7 POT DUMP VALVE CLOSED"
522    0345          IF ILOOP = 5 THEN STRNGO = "SLURRY SAMPLER ENCLOSURE LEAK"
523    0347          IF ILOOP = 6 THEN STRNGO = "HV-CA1-8 RAW WATER VALVE OPEN"
524    0349          IF ILOOP = 7 THEN STRNGO = "HV-CA1-8 RAW WATER VALVE CLOS"
525    0351          IF ILOOP = 8 THEN STRNGO = "CA1-9 POT DUMP VALVE CLOSED"
526    0353          IF ILOOP = 9 THEN STRNGO = "CA1-9 POT DUMP VALVE OPEN"
527    0355          IF ILOOP = 28 THEN STRNGO = "JGV-SUMP ON 8 + MINUTES "
528    0357          IF ILOOP = 24 THEN STRNGO = "START PB1BATCH PGM "
529    0359          IF ILOOP = 25 THEN STRNGO = "PB1BATCH WRONG UNIT"
530    0361          IF ILOOP = 26 THEN STRNGO = "TERMINATE PB1BATCH "
531    0363          IF ILOOP = 30 THEN STRNGO = "F-CA1-L/R DELTA PRESS HI "
532          ;
533    0364  AMSG:   IF PTYPE = 1 THEN
534                     BEGIN
535                     ;
536                     ; If white alarm, print:
537                     ;
538    0366             IF (FACK = 0) & (FCLEAR = 0) & (FSTART = 0) THEN PRINT ADI,"WHITE ALARM %p",STRNGO
539    0367             IF (FACK) THEN
540                       BEGIN
541                       ;
542                       ; If white, print it is ack are reset the flag:
543                       ;
544    0368               PRINT ADI,"%p ALARM ACKNOWLEDGE",STRNGO
545    0369               LET FACK = 0
546                       END
547                     END
548          ;
549          ;  If CLEAR, print it is clear and reset the flag:
550          ;
```

```
551   0370            IF (FCLEAR) THEN
552                      BEGIN
553   0371               PRINT ADI,"%p ALARM CLEAR",STRNG0
554   0372               LET FCLEAR = 0
555                      END
556           ;
557           ;  If STARTUP, then print startup message:
558           ;
559   0373            IF (FSTART) THEN
560                      BEGIN
561   0374               LET FSTART = 0
562   0375               PRINT ADI,"%p",STRNG0
563                      END
564   0376            RETURN
565           ;
566           ;  If ERROR - print message, start timer, and set flag:
567           ;
568   0377    ERRSUB:  IF ERRN != ERRNUM() THEN
569                      BEGIN
570   0378               TIMERON STIME0
571   0379               TIMERCLR'STIME0
572   0380               LET FSN0 = 1
573   0381               LET ERRN = ERRNUM()
574   0382               LET ERRL = ERRLIN()
575   0383               PRINT ADI,"ERROR # %i AT LINE %i",errn,errl
576                      END
577   0384               ERSCLR
578   0385            GOTO AGAIN
579           ;
580           ;  If FABORT = 1, then abort program:
581           ;
582           ; TERM:  LET FABORT = 0
583   0386    TERM:    LET STRNG0 = "PB1BATCH TERMINATED    "
584   0387            LET ILOOP = STAR + 2
585   0388            LET FSTART = 1
586   0389            GOSUB ACKK
587                    END

        Total Errors Detected      0     Total Warnings      0
             Sequence Program Size = 7648 Bytes, 478 Pages
                  21 Forward Reference(s)
```

PB2BATCH

```
;       P B 2 B A T C H . S E Q                                 VERSION 2
;
;       Function:       Sequence program for Lamp function,
;                       message display on Graphic number 015,
;                       and printer messages.
;
;       Written By      : R. N. Kersey
;       Date            : 05-31-92
;       Facility        : 242-A Evaporator Bldg.
;       Machine         : VAX 4104-A
;       Op. System      : VAX/VMS 5.5-2AH
;       D/3 Version     : 6.3-2 BASELINE
;       Compiler        : SIC
;       Build command   : @[320,20]PB2BATCH
;
;       UNIT = 8, SKDKEY = 35
;
;       Assoc. UPF:     [300,1]_PRCESS.___      PRCESS.LST
;       Assoc. RPF:     [300,1]_242A.___        242A.LST
;       SDB:            [300,1]PCM0A.SDB        PCM0ASDB.CMD
;
;       POINTER   SDB VAR.      FUNCTION
;       -------   ---------     ------------------------------------------
;       PSTAT     INTn          Current EPN alarm state is an integer tied to
;                               graphic flashing string of value:
;       .         (n=0-100)             0 = previously not alarmed
;       .                               1 = previously alarmed, not ack
;       .                               2 = previously alarmed, is ack
;       .                               3 = alarm is inhibited
;       .                               4 = input is off scan
;       .                               5 = input is bad
;       .         LEAKn         Special INT variable for multiple white alarms on an EPN
;       PDIN      'EPN'         ---
;       PTYPE     TYPEn         alarm type 0 = none, 1 = white, 2 = yellow,
;                                  3 = yellow and yellow, 4 = red,
;                                  6 = yellow and red
;       PREV      FRVn          Reverse acting (not fail safe) flag (1=set)
;       PSKID     SKIDn         Used for analog alarms
;       PPASS     FPASn         Turn lamp off when alarm is ACKnowledged
;       PALARM    FALMn         Analog alarm flag
;       ---       FNING         Unused error flag
;
;  MODIFICATION LOG:
;
;       20-AUG-92       Set frv5=0.                                     M. Teats
;
;       08-SEP-92       Comment-out auto-initializing of DIC-CA1-1      M. Teats
;                       (search ";NU") and deleted CA1 Unit in
;                       COMMON for synch PID.
;
;       30-DEC-92       Fixed white alarm print message.        M. Teats
;
;       07-MAY-93       Added bldg raw water alarm, FQA-RW-1.           M. Teats
;
;       07-JUN-93       Added comments and baselined for        M. Teats
```

```
56          ;                       D/3 version 3.3.  CCN# 93JUN01-01.
57          ;
58          ;       15-DEC-94       Revised header to indicate the new VMS          J. Berger
59          ;                       system. Rebase line document. Remove
60          ;                       obsolite Select() statement.
61          ;
62          ;   END OF MOD LOG
63          ;
64          UNIT    PB2
65          ;
66                  COMMON WHOLE0,WHOLE1
67                  PB2_TMP0 == 'PB-2':DEV_TMP0
68                  PB2_TMP1 == 'PB-2':DEV_TMP1
69                  SLURRY == 'PB2SLFILK':DEV_TMP0
70                  WFACTOR == 'PB2WFINLK':DEV_TMP0
71                  GBITS   == 'PB2GLOBAL':DEV_TMP0
72                  FLAGBITS == PSPAR0
73                  HV_TMP0 == 'HV-CA1-2':DEV_TMP0
74                  HV_CMD == 'HV-CA1-2':DEV_CMD
75          ;
76  0000            NOP
77  0001            LET STAR = 24                   ; Start message
78  0002            LET SKDKEY = 35         ; SKID Key
79  0003            LET UNITT = 8                   ; SEQ Unit number
80  0004            ASSIGN PDEV = 'PB2-ALARM'       ; Alarm device
81  0005            ASSIGN PTYPE = TYPE24
82  0006            PTYPE = 0                       ; No start message yet
83  0007            LET ADI = 0                     ; Alarm destination index
84  0008            LET FSN0 = 0                    ; No alarm yet
85  0009            TIMEROFF STIME0         ; Alarm timer off
86  0010            TIMERCLR STIME0         ; Alarm timer clear
87  0011            STRNG0 = " "                    ; Blank 30 character string
88  0012            LET ERRN = 0                    ; Set error number to zero
89  0013            LET ERRL = 0                    ; Set error line to zero
90  0014            FTRIP4 = 0                      ; FTRIP = 1 when audible horn is to be turned on
91  0015            GETUNIT TRIP                    ; Which sequence unit is program running in
92  0016            IF TRIP != UNITT THEN           ; Abort if running in wrong seq. unit
93                     BEGIN
94  0017               FSTART = 1
95  0018               ILOOP = STAR + 1
96  0019               PTYPE = 0
97  0020               GOSUB ACKK
98  0021               GOTO TERM
99                     END
100         ;       IF !SELECT() THEN GOTO AGAIN
101         ;
102         ;       B E G I N   I N I T I A L I Z I N G
103         ;
104 0022    INIT:   LET FABORT = 0                  ; If FABORT = 1 then abort program
105 0023            LET FSTART = 0                  ; Program start flag = 0
106 0024            LAMP ADI,SKDKEY,0               ; Turn off the annunciator key
107 0025            LET LSG0 = " "                  ; Clear the string variables
108 0026            LET LSG1 = " "
109 0027            LET LSG2 = " "
110 0028            LET LSG3 = " "
```

```
111  0029          LET LSG4 = " "
112  0030          PDEV:DEV_TMP0 = 1                          ; Set alarm device to non-alarm state
113  0031          IF AND(PDEV:DEV_ALMCT,16) THEN             ; If alarm device is inhibited, remove the inhibit
114  0032            PDEV:DEV_ALMCT=XOR(PDEV:DEV_ALMCT,16)
115  0033          IF AND(PDEV:DEV_ALMCT,32) THEN             ; If alarm device is ack, clear it
116  0034             PDEV:DEV_ALMCT = XOR(PDEV:DEV_ALMCT,32)
117           ;
118           ;      First, set all flags and integers:
119           ;
120           ;      DIN = 0 - 8, AI = 9 - 21, WHITE = 22
121           ;
122  0035          ASSIGN PDIN = 'YS-PB2-1'                   ; Assign pointer to first DIN
123  0036          ASSIGN PSTAT = INT0
124  0037          ASSIGN PTYPE = TYPE0
125  0038          ASSIGN PREV = FRV0
126  0039          ASSIGN PSKID = SKID0
127  0040          ASSIGN PPASS = FPAS0
128           ;
129  0041          FOR ILOOP = 0 TO 21
130  0042          LET PPASS = 0
131  0043          LET PREV = 0
132  0045          IF PSTAT > 5 THEN PSTAT = 0                ; Reset current status if undefined
133  0046          LET PSKID = 0                              ; Default previous status to no alarms
134  0047          LET PTYPE = 0                              ; Default to no alarms
135  0048          IF ILOOP < 9 THEN                          ; For DINs
136                   BEGIN
137  0049             LET PTYPE = 0                           ; Default is no alarm
138  0050             IF AND(PDIN:DIN_DFLG,3) THEN            ; Check for alarms if defined (must be red or yellow)
139                        BEGIN
140  0051                  LET PTYPE = 2                      ; Default to yellow alarm
141  0053                  IF AND(PDIN:DIN_DFLG,1) THEN LET PREV = 1          ; Reverse acting
142  0055                  IF AND(PDIN:DIN_DFLG,4) THEN LET PTYPE = 4   ; Set to red critical alarm
143                        END
144                   END
145  0057          IF ILOOP < 8 THEN STEP PDIN BY 1           ; Point to next DIN
146  0058          STEP PPASS BY 1
147  0059          STEP PREV BY 1
148  0060          STEP PSTAT BY 1
149  0061          STEP PSKID BY 1
150  0062          STEP PTYPE BY 1
151  0063          NEXT ILOOP
152           ;
153  0064          LET TYPE10 = 2
154  0065          LET TYPE12 = 2
155  0066          LET TYPE13 = 4
156  0067          LET TYPE15 = 2
157  0068          LET TYPE16 = 2
158  0069          LET TYPE19 = 2
159  0070          LET TYPE22 = 1
160           ;
161  0071          LET FRV0 = 0
162  0072          LET FRV1 = 0
163  0073          LET FRV2 = 0
164  0074          LET FRV3 = 0
165  0075          LET FRV4 = 0
```

```
166   0076            LET FRV5 = 0
167           ;
168   0077            LET AI0 = 95.0
169           ;
170   0078            FLG14 = 0               ; PID auto (1) flag, for synch PID
171   0079            FLG15 = 0               ; PID manual (1) flag, for synch PID
172   0080            AI20 = 0.0              ; For PID O/C display bar
173           ;
174           ;       E N D   I N I T I A L I Z I N G
175           ;
176           ;       S E N D   P R O G R A M   S T A R T   M E S S A G E
177           ;
178   0081            ANALOG = 0
179   0082            ILOOP = STAR
180   0083            FSTART = 1
181   0084            ASSIGN PTYPE = TYPE34
182   0085            GOSUB ACKK
183   0086            ON ERROR GOSUB ERRSUB
184           ;
185           ;       B E G I N   P R O G R A M   L O O P
186           ;
187   0087    AGAIN:  MARK STEP "WTUNTL",0,0
188           ;       WAITUNTIL SELECT() ¦ FABORT     ; Wait until micro is selected or program aborted
189           ;       IF (FABORT) THEN GOTO TERM      ; Terminate program if abort flag set
190   0088            MARK STEP "RNNING",0,0
191   0089            WAIT 1
192   0090            IF (FSN0) THEN                  ; If alarm flag is set
193   0091               IF STIME0 > 60 THEN          ; If alarm flag active for 1 minute
194                        BEGIN                     ; Reset timer and turn off flag
195   0092                 TIMEROFF STIME0
196   0093                 TIMERCLR STIME0
197   0094                 LET FSN0 = 0
198   0095                 LET ERRN = 0
199   0096                 LET ERRL = 0
200                        END
201           ;
202           ;       If PB2-ALARM $T1 is alarmed and it's ALMCT indicates alarms are acknowledged:
203           ;
204   0097            IF (PDEV:DEV_TMP0 = 2) & AND(PDEV:DEV_ALMCT,32) THEN
205                      BEGIN
206   0098               PDEV:DEV_TMP0 = 1            ; Put PB2-ALARM in ack state
207   0099               LET FTRIP4 = 1               ; Set for unit alarm ack, system to graphic
208   0100               LET ACK = 1                  ; Set for unit alarm ack, graphic to system
209   0101               CRIT
210   0102               LAMP ADI,SKDKEY,1            ; Turn lamp on
211                      END
212           ;
213           ;       BEGIN DINs:
214           ;
215   0103            ASSIGN PDIN = 'YS-PB2-1'
216   0104            ASSIGN PTYPE = TYPE0
217   0105            ASSIGN PREV = FRV0
218   0106            ASSIGN PSTAT = INT0
219   0107            ASSIGN PPASS = FPAS0
220   0108            LET CURENT = 0                  ; Used to set lamp status
```

```
221   0109          LET LAST = 0
222   0110          LET ANALOG = 0
223   0111          LET NXT = 0
224   0112          LMPVAR = 0
225   0113          BITVAL = 0
226         ;
227   0114  LOP:    FOR ILOOP = 0 TO 8
228         ;       IF !SELECT() THEN GOTO AGAIN
229   0115          IF ( PTYPE > 0 ) THEN                                   ; If some type of alarm
230                   BEGIN                                         ;be1
231   0117            IF AND(PDIN:DIN_ALMCT,16) THEN LET PSTAT = 3  ; If EPN inhibited
232   0119            IF AND(PDIN:DIN_ALMCT,4) THEN LET PSTAT = 5           ; If EPN is bad input
233   0121            IF !AND(PDIN:DIN_FLG,8) THEN LET PSTAT = 4            ; If EPN is disabled
234   0122            IF ( PSTAT < 3 ) THEN                         ; If not inhibited
235                       BEGIN                                     ;be2
236   0123                IF (PSTAT = 1) THEN
237                         BEGIN                                   ;be3
238   0124                  IF (PTYPE != 1) THEN
239   0126                    IF AND(PDIN:DIN_ALMCT,32) THEN GOSUB ACK1
240   0128                  IF (FTRIP4) THEN GOSUB ACK1
241                         END                                     ;en3
242   0131                IF PSTAT = 0 THEN IF(PDIN:DIN_VAL = PREV) THEN GOSUB ALM
243   0132                IF (PSTAT) THEN
244                           BEGIN                                 ;be4
245   0134                    IF (PDIN:DIN_VAL != PREV) THEN GOSUB CLR
246   0135                    NXT = PSTAT
247   0137                    IF (PPASS) & (PSTAT = 2) THEN NXT = 0
248   0138                    LAST = OR(LAST,NXT)
249                           END                                   ;en4
250                       END                                       ;en2
251   0140              ELSE IF !AND(PDIN:DIN_ALMCT,16)&!AND(PDIN:DIN_ALMCT,4)&AND(PDIN:DIN_FLG,8)THEN LET PSTAT = 0
252                     END                                         ;en1
253   0141          STEP PDIN BY 1
254   0142          STEP PTYPE BY 1
255   0143          STEP PREV BY 1
256   0144          STEP PSTAT BY 1
257   0145          STEP PPASS BY 1
258   0146          NEXT ILOOP
259         ;
260         ;       END DINs
261         ;
262         ;       BEGIN ANALOG RED AND YELLOWS:
263         ;
264   0147          ASSIGN PAI = 'HI-WVR-1'
265   0148          ASSIGN PSKID = SKID9
266   0149          FOR JLOOP = 9 TO 21
267         ;       IF !SELECT() THEN GOTO AGAIN
268   0150          IF (PTYPE) THEN                                         ; If some alarm exists
269                   BEGIN                                       ;be1
270   0151            BITVAL = PAI:AI_ALMCT
271   0152            IF !AND(BITVAL,16) THEN                               ; If not inhibited
272                     BEGIN                                     ;be2
273   0153              IF AND(BITVAL,15) THEN                              ; If alarm exists and is current
274                       BEGIN                                   ;be3
275   0155                IF AND(BITVAL,32) THEN NXT = 2                    ; If alarm is ack
```

```
276                          ELSE                                    ; If alarm is unack
277                             BEGIN                          ;be4
278   0156                      IF PSKID !=BITVAL THEN
279                                BEGIN                       ;be5
280   0157                         PSKID = BITVAL
281   0159                         IF BITVAL < 3 THEN FALMON = 1
282   0160                         LET PDEV:DEV_TMPO = 2
283   0161                         NXT = 1
284                                END                         ;en5
285   0162                      ELSE NXT = 1
286                             END                            ;en4
287                          END                               ;en3
288                       ELSE                                       ; Else there is no alarm
289                          BEGIN
290   0163                   NXT = 0
291   0164                   LET PSKID = 0
292                          END
293                       END                                  ;en2
294                    ELSE                                          ; Else alarm is inhibited
295                       BEGIN
296   0165                NXT = 0
297   0166                LET PSKID = 0
298                       END
299   0167              ANALOG = OR(ANALOG,NXT)
300                       END                                  ;en1
301   0168           STEP PREV BY 1
302   0169           STEP PSTAT BY 1
303   0170           STEP PSKID BY 1
304   0171           STEP PTYPE BY 1
305   0172           STEP PAI BY 1
306   0173           NEXT JLOOP
307            ;
308            ;      END ANALOG RED AND YELLOWS
309            ;
310            ;      BEGIN ANALOG WHITE ALARMS:
311            ;
312   0174     WHT:  WHITE = 0
313   0175           ASSIGN PAI = 'SIC-PB2-1'
314   0176           ASSIGN PALARM = AI0
315   0177           ASSIGN PREV = FRV22
316   0178           ASSIGN PSTAT = INT22
317   0179           ASSIGN PTYPE = TYPE22
318   0180           ASSIGN PPASS = FPAS22
319   0181           ILOOP = 22
320   0182           GOSUB ANLG
321            ;
322            ;      END ANALOG WHITE ALARMS:
323            ;
324   0183           MARK STEP "BELOW",0,0
325   0185           IF LAST > 3 THEN PRINT ADI," LAMP VARIABLE IS GREATER THAN 3!"
326   0186           CURENT = OR(LAST,WHITE)
327   0187           CURENT = OR(CURENT,ANALOG)
328   0189           IF AND(CURENT,1)THEN LMPVAR = 2
329   0191              ELSE IF AND(CURENT,2)THEN LMPVAR = 1
330   0192                      ELSE LMPVAR = 0
```

```
331  0193          LAMP ADI,SKDKEY,LMPVAR
332  0194          MARK STEP "BOTTOM",0,0
333  0195          NOCRIT
334  0196          LET FTRIP4 = 0
335          ;
336          ;     BEGIN LOGIC FOR GRAPHICS:
337          ;
338  0197          at21 = and('PSH-CA1-3':DIN_VAL,'TSH-PB2-1':DIN_VAL)
339  0198          at22 = and('ys-pb2-il':din_val,'psh-pb2-1':din_val)
340  0199          at23 = and(pb2_tmp1,128)
341  0200          at24 = and(pb2_tmp1,256)
342  0201          at25 = and(pb2_tmp1,512)
343  0202          at26 = and(at21,at22)
344  0203          at27 = and(at23,at24)
345  0204          at28 = and(at25,at26)
346  0205          at29 = and(at27,at28)
347          ;
348  0206          AT31 = AND(PB2_TMP1,32768)
349  0207          AT0  = AND(WFACTOR,1)
350  0208          AT1  = AND(wfactor,2)
351  0209          AT2  = AND(wfactor,4)
352  0210          AT3  = AND(wfactor,8)
353  0211          GT3  = AND(GBITS,8)
354  0212          GT4  = AND(GBITS,16)
355  0213          GT9  = AND(GBITS,512)
356  0214          GT11 = AND(GBITS,2048)
357  0215          BT0 = AND('LDS-SL502':DIN_VAL,'LDS-SL503':DIN_VAL)
358  0216          BT1 = AND('LDS-SL504':DIN_VAL,'LDS-SL505':DIN_VAL)
359  0217          BT2 = AND('LDS-DR700':DIN_VAL,'LDS-SL500':DIN_VAL)
360  0218          BT3 = AND(BT2,'LDS-SL501':DIN_VAL)
361  0219          BT4 = AND(BT0,BT1)
362  0220          AT9 = AND(BT3,BT4)
363  0221          BT5 = AND(!'YS-AN-MSD':DIN_VAL,'LDS-COBAN':DIN_VAL)
364  0222          BT6 = AND('LDS-ANSL1':DIN_VAL,'LDS-DR368':DIN_VAL)
365  0223          BT7 = AND('LDS-SP-AN':DIN_VAL,'LDS-AP-A':DIN_VAL)
366  0224          BT8 = AND(!'YS-AP-MSD':DIN_VAL,'LDS-AP-VP':DIN_VAL)
367  0225          BT9 = AND(BT5,BT6)
368  0226          BT10= AND(BT7,BT8)
369  0227          AT10= AND(BT9,BT10)
370  0228          BT11= AND('PSH-CA1-3':DIN_VAL,'TSH-PB2-1':DIN_VAL)
371  0229          BT12= AND('PSH-PB2-1':DIN_VAL,'PSH-SL113':DIN_VAL)
372  0230          BT13= AND('PSH-SL114':DIN_VAL,'PSH-SL101':DIN_VAL)
373  0231          BT14= AND(BT13,'PSH-SL100':DIN_VAL)
374  0232          BT15= AND(BT14,BT12)
375  0233          AT11= AND(BT14,BT11)
376  0234          BT16= AND('LDS-AP-SL':DIN_VAL,!'YS-AW-MSD':DIN_VAL)
377  0235          BT17= AND('LDS-AW-SL':DIN_VAL,'RSH-SP-AW':DIN_VAL)
378  0236          BT18= AND('LDS-AW-DR':DIN_VAL,'LDSCOBAAX':DIN_VAL)
379  0237          BT19= OR('RSH-AXZY1':DIN_VAL,'YS-AAXARM':DIN_VAL)
380  0238          BT20= AND(BT19,'LDSPPCOB1':DIN_VAL)
381  0239          BT21= AND(BT16,BT17)
382  0240          BT22= AND(BT18,BT20)
383  0241          AT12= AND(BT21,BT22)
384  0242          BT23= AND(AT9,AT10)
385  0243          BT24= AND(AT11,AT12)
```

```
386  0244          AT13= AND(BT23,BT24)
387  0245          BT25= AND(PB2_TMP0,1)
388  0246          BT26= AND(BT25,AT31)
389  0247          SA1 = AND(AT20,BT26)
390  0248          SA3 = AND(SA1,AT13)
391  0249          SA2 = AND('PB-2':DEV_STAT,4)      ; PB-2 RESET
392  0250          SA4 = AND('PB-2':DEV_STAT,16)     ; PB-2 SHUTDOWN
393  0251          SB4 = AND(PB2_TMP0,2)             ; PB-2 TEMPORARY BIT $T1
394  0252          SB5 = 'YS-PB2-1':DIN_VAL
395  0253          SB6 = AND('PB-2':DEV_STAT,32)     ; PB-2 INTERLOCK
396  0254          CT0 = AND(1,HV_TMP0)              ; FIC-CA1-4 LOW FLOW  $T0  HV-CA1-2
397  0255          CT2 = AND(4,HV_CMD)               ; EVAPORATOR FLUSH,   $C2  HV-CA1-2
398  0256          CT3 = AND(8,HV_TMP0)              ; EVAP FLUSH TIMEOUT, $T3  HV-CA1-2
399  0257          CT4 = AND(4,HV_TMP0)              ; EVAP BTIMER ON       $T2  HV-CA1-2
400  0258          CT5 = AND(2048,HV_TMP0)           ; FARM BTIMER ON       $T11 HV-CA1-2
401  0259          CT6 = 'ZS-CA1-21':DIN_VAL         ; HV-CA1-2  POSN-1
402  0260          CT7 = 'ZS-CA1-22':DIN_VAL         ; HV-CA1-2  POSN-2
403  0261          CT8 = 'ZS-CA12A1':DIN_VAL         ; HV-CA1-2A POSN-1
404  0262          CT9 = 'ZS-CA12A2':DIN_VAL         ; HV-CA1-2A POSN-2
405  0263          CT10= XOR(CT5,CT12)
406  0264          CT11= AND(8,HV_CMD)               ; FARM FLUSH          $C3  HV-CA1-2
407  0265          CT12= AND(4096,HV_TMP0)           ; FARM FLUSH TIMEOUT  $T12 HV-CA1-2
408  0266          CT13= AND(CT4,CT3)
409  0268          IF AND('PB2-ALARM':DEV_FLG,64)THEN PUTMANL 'PB2-ALARM'
410  0270          IF AND('PB-2':DEV_FLG,64)THEN PUTMANL 'PB-2'
411  0272          IF AND('HV-CA1-5':DEV_FLG,64)THEN PUTMANL 'HV-CA1-5'
412  0274          IF AND('HV-CA1-6':DEV_FLG,64)THEN PUTMANL 'HV-CA1-6'
413  0276          IF AND('JGV-STMFL':DEV_FLG,64)THEN PUTMANL 'JGV-STMFL'
414  0278          IF AND('CONTROL':DEV_FLG,64) THEN PUTMANL 'CONTROL'
415  0280          IF AND('PB2SLFILK':DEV_FLG,64) THEN PUTMANL 'PB2SLFILK'
416  0282          IF AND('PB2WFINLK':DEV_FLG,64) THEN PUTMANL 'PB2WFINLK'
417           ;
418           ;     END LOGIC FOR GRAPHICS
419           ;
420           ;     B E G I N   S Y N C H   P I D   C A S C A D E D   S E T P T :
421           ;
422           ;     Function:  Closes slave controller's PID (SPID) cascade,
423           ;                sets cascade controller's PID (CPID) to auto.
424           ;                (NOTE:  the continuous database sets the CPID
425           ;                output equal to the SPID setpoint only on a
426           ;                manual to auto transition in the SPID.)
427           ;
428  0283          IF AND('SIC-PB2-1':CB_FLG:1,64) THEN          ; If SIC in auto
429                   BEGIN                                          ;be1
430  0284               IF (FLG14=0) & (FLG15=1) THEN             ; If SIC previously manual
431                        BEGIN                                     ;be2
432  0285                  CLOSCASC 'SIC-PB2-1'
433  0287                  IF !AND('FIC-CA1-4':CB_FLG:4,64) THEN PUTAUTO 'FIC-CA1-4'
434  0288                  LET FLG14 = 1
435  0289                  LET FLG15 = 0
436                        END                                       ;en2
437                   END                                            ;en1
438               ELSE                                          ; If SIC in manual
439                   BEGIN                                          ;be3
440  0290               IF ((FLG14=1)&(FLG15=0)) ¦ ((FLG14=0)&(FLG15=0)) THEN   ; If prev auto or init
```

```
441                         BEGIN                                          ;be4
442    0291                 LET FLG14 = 0
443    0292                 LET FLG15 = 1
444                         END                                            ;en4
445                   END                                                  ;en3
446          ;
447          ;       E N D   S Y N C H   P I D   C A S C A D E D   S E T P T
448          ;
449          ;       B E G I N   O / C   D I S P L A Y   B A R :
450          ;
451          ;       Function:  Upon loss of permissive, put the control loop in manual
452          ;                  and set the PID block output to zero.
453          ;
454    0294          IF ('SIC-PB2-1':CB_LOUT:3 >= 2) THEN DLEAK3 = 1          ; Permissive must be Hi (1)
455    0295          IF ('SIC-PB2-1':CB_LOUT:3 = 0) & AND(DLEAK3,1) THEN      ; Permissive lost and 1st pass
456             BEGIN
457    0296        PUTMANL 'SIC-PB2-1'
458    0297        PRINT ADI," PERMISSIVE LOST >>> SIC-PB2-1 PUT IN MANUAL"
459    0298        PUTOUT 'SIC-PB2-1',AI20
460    0299        PRINT ADI," PERMISSIVE LOST >>> SIC-PB2-1 OUTPUT SET TO ZERO"
461    0300        DLEAK3 = 0
462             END
463          ;
464          ;       E N D   O / C   D I S P L A Y   B A R
465          ;
466    0301          GOTO AGAIN
467          ;
468          ;       E N D   P R O G R A M   L O O P
469          ;
470          ;       BEGIN ANALOG WHITE SUBROUTINE:
471          ;
472          ; ANLG:  IF !SELECT() THEN GOTO AGAIN
473    0302  ANLG:    IF (PTYPE = 1) THEN                                    ; If some alarm
474                      BEGIN                                       ;be1
475    0303              BITVAL = PAI:AI_ALMCT
476    0304              IF !(AND(BITVAL,16)) THEN                           ; If not inhibited
477                        BEGIN                                     ;be2
478    0306                IF PSTAT = 3 THEN LET PSTAT = 0
479    0307                IF (PREV) THEN                                  ; If rev acting, LOLO, or LO
480                          BEGIN                                   ;be3
481    0308                  IF (PSTAT = 0) THEN                           ; If not previously alarmed
482                            BEGIN                                 ;be4
483    0310                    IF PAI:AI_MEAS <= PALARM THEN GOSUB ALM
484                            END                                   ;en4
485                          ELSE
486                            BEGIN                                 ;be5
487    0311                    IF (PSTAT = 1) THEN                         ; If previously alarmed
488                              BEGIN                               ;be6
489    0313                      IF (FTRIP4) THEN GOSUB ACK1               ; If ACK flag set, then ack it
490    0316                      IF PTYPE !=1 THEN IF AND(PAI:AI_ALMCT,32) THEN GOSUB ACK1
491                              END                                 ;en6
492    0318                    IF (PAI:AI_MEAS >= PALARM + PAI:AI_DEAD) THEN GOSUB CLR
493                            END                                   ;en5
494                          END                                     ;en3
495                        ELSE                                            ; If fail-safe, HIHI, or HI
```

```
496                          BEGIN                                ;be10
497    0319                  IF (PSTAT = 0) THEN
498                             BEGIN                             ;be11
499    0321                     IF (PAI:AI_MEAS >= PALARM) THEN GOSUB ALM
500                             END                               ;en11
501                          ELSE
502                             BEGIN                             ;be12
503    0322                     IF (PSTAT = 1) THEN
504                                BEGIN                          ;be13
505    0324                        IF (FTRIP4) THEN GOSUB ACK1
506    0327                        IF PTYPE != 1 THEN IF AND(PAI:AI_ALMCT,32) THEN GOSUB ACK1
507                                END                            ;en13
508    0329                     IF (PAI:AI_MEAS <= PALARM - PAI:AI_DEAD) THEN GOSUB CLR
509                             END                               ;en12
510                          END                                  ;en10
511    0330               NXT = PSTAT
512    0332               IF (PPASS) & (PSTAT = 2) THEN NXT = 0
513                       END                                   ;en2
514                       ELSE
515                         BEGIN
516    0333                 LET PSTAT = 3
517    0334                 NXT = 0
518                         END
519    0335             WHITE = OR(WHITE,NXT)
520                     END                                     ;en1
521    0336             ELSE PSTAT = 0
522    0337             STEP PALARM BY 1
523    0338             STEP PSTAT BY 1
524    0339             STEP PTYPE BY 1
525    0340             STEP PREV BY 1
526    0341             STEP PPASS BY 1
527    0342             ILOOP = ILOOP + 1
528    0343             RETURN
529             ;
530             ;       END ANALOG WHITE SUBROUTINE
531             ;
532             ;       BEGIN CLEAR, ALARM, ACKNOWLEDGE, MESSAGE, ERROR, AND TERMINATE SUBROUTINES:
533             ;
534    0344     CLR:    PSTAT = 0                                          ; Int set to zero
535                     ;
536                     ; If not white, then unack:
537                     ;
538    0347             IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
539    0348             LET FCLEAR = 1                                     ; Clear flag
540    0349             GOSUB ACKK                                         ; Print message
541    0350             RETURN
542             ;
543    0351     ALM:    LET PSTAT = 1                                      ; Int set to one
544                     ;
545                     ; If not white, then unack:
546                     ;
547    0354             IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
548                     ;
549                     ; Turn audible alarm on if white or yellow:
550                     ;
```

```
551   0355            CRIT
552   0357            IF AND(PTYPE,3) THEN FALMON = 1
553   0358            LET PDEV:DEV_TMPO = 2
554   0359            LAMP ADI,SKDKEY,2
555   0360            GOSUB ACKK                                          ; Print message
556   0361            RETURN
557           ;
558   0362    ACK1:   LET PSTAT = 2                                       ; Ack the alarm
559   0363            LET FACK = 1                                        ; Set ACK flag for C program
560   0364            LAMP ADI,SKDKEY,1                                   ; Lamp on
561           ;
562           ;       ALARM TYPER STRINGS:
563           ;
564   0366    ACKK:   IF ILOOP = 1 THEN STRNGO = "PUMP PB-2 HARDWARE INTERLOCK"
565   0368           IF ILOOP = 2 THEN STRNGO = "PUMP PB-2 TEMPERATURE HIGH  "
566   0370            IF ILOOP = 3 THEN STRNGO = "PUMP PB-2 DISCHARGE PRESS HI"
567   0372            IF ILOOP = 4 THEN STRNGO = "PUMP PB-2 RUPTURE DISK BROKN"
568   0374            IF ILOOP = 5 THEN STRNGO = "PUMP PB-2 ENCODER LOSS      "
569   0376            IF ILOOP = 6 THEN STRNGO = "PB-2 VARABLE FREQ DRIVE LOSS"
570   0378            IF ILOOP = 7 THEN STRNGO = "PUMP PB-2 ZERO SPEED        "
571   0380            IF ILOOP = 8 THEN STRNGO = "BUILDING FLUSH WATER ON     "
572   0382            IF ILOOP = 22 THEN STRNGO = "PUMP PB-2 CURRENT HIGH     "
573   0384            IF ILOOP = 24 THEN STRNGO = "START PB2BATCH PGM "
574   0386            IF ILOOP = 25 THEN STRNGO = "PB2BATCH WRONG UNIT"
575   0388            IF ILOOP = 26 THEN STRNGO = "TERMINATE PB2BATCH "
576           ;
577           ; If acknowledging alarm, then:
578           ;
579   0389    AMSG:   IF PTYPE = 1 THEN
580                     BEGIN
581                     ;
582                     ; If white alarm, print:
583                     ;
584   0391              IF (FACK = 0) & (FCLEAR = 0) & (FSTART = 0) THEN PRINT ADI,"WHITE ALARM %p",STRNGO
585   0392              IF (FACK) THEN
586                       BEGIN
587                       ;
588                       ; If white, print it is ack and reset the flag:
589                       ;
590   0393                PRINT ADI,"%p ALARM ACKNOWLEDGE",STRNGO
591   0394                LET FACK = 0
592                       END
593                   END
594           ;
595           ;  If CLEAR, print it is clear and reset flag:
596           ;
597   0395            IF (FCLEAR) THEN
598                     BEGIN
599   0396              PRINT ADI,"%p ALARM CLEAR",STRNGO
600   0397              LET FCLEAR = 0
601                     END
602           ;
603           ;  If STARTUP, print startup message:
604           ;
605   0398            IF (FSTART) THEN
```

```
606                         BEGIN
607    0399                 LET FSTART = 0
608    0400                 PRINT ADI,"%p",STRNGO
609                         END
610    0401               RETURN
611             ;
612             ;  If ERROR - print message, start timer, and set flag:
613             ;
614    0402     ERRSUB:    IF ERRN != ERRNUM() THEN
615                         BEGIN
616    0403                 TIMERON STIME0
617    0404                 TIMERCLR STIME0
618    0405                 LET FSN0 = 1
619    0406                 LET ERRN = ERRNUM()
620    0407                 LET ERRL = ERRLIN()
621    0408                 PRINT ADI,"ERROR # %i AT LINE %i",errn,errl
622                         END
623    0409               ERSCLR
624    0410               GOTO AGAIN
625             ;
626             ;  If FABORT = 1, then abort program:
627             ;
628             ; TERM:  LET FABORT = 0
629    0411     TERM:     LET STRNGO = "PB2BATCH TERMINATED    "
630    0412               LET ILOOP = STAR + 2
631    0413               LET FSTART = 1
632    0414               GOSUB ACKK
633                       END

       Total Errors Detected      0     Total Warnings      0
            Sequence Program Size = 8400 Bytes, 525 Pages
                 19 Forward Reference(s)
```

EA1BATCH

```
;       E A 1 B A T C H . S E Q                                  VERSION 2
;
;       Function:       Sequence program for Lamp function,
;                       message display on Graphic number 013,
;                       and printer messages.
;
;       Written By      : R. N. Kersey
;       Date            : 05-31-92
;       Facility        : 242-A Evaporator Bldg.
;       Machine         : VAX 4104-A
;       Op. System      : VAX/VMS 5.5-2AH
;       D/3 Version     : 6.3-2 BASELINE
;       Compiler        : SIC
;       Build command   : @[320,20]EA1BATCH
;
;       UNIT = 9, SKDKEY = 33
;
;       Assoc. UPF:     [300,1]_PRCESS.___      PRCESS.LST
;       Assoc. RPF:     [300,1]_242A.___        242A.LST
;       SDB:            [300,1]PCM0A.SDB        PCM0ASDB.CMD
;
;       POINTER   SDB VAR.      FUNCTION
;       -------   ---------     ----------------------------------------
;       PSTAT     INTn          Current EPN alarm state is an integer tied to
;                               graphic flashing string of value:
;       .         (n=0-100)             0 = previously not alarmed
;       .                               1 = previously alarmed, not ack
;       .                               2 = previously alarmed, is ack
;       .                               3 = alarm is inhibited
;       .                               4 = input is off scan
;       .                               5 = input is bad
;       .         LEAKn         Special INT variable for multiple white alarms on an EPN
;       PDIN      'EPN'         ---
;       PTYPE     TYPEn         alarm type 0 = none, 1 = white, 2 = yellow,
;                                  3 = yellow and yellow, 4 = red,
;                                  6 = yellow and red
;       PREV      FRVn          Reverse acting (not fail safe) flag (1=set)
;       PSKID     SKIDn         Used for analog alarms
;       PPASS     FPASn         Turn lamp off when alarm is ACKnowledged
;       PALARM    FALMn         Analog alarm flag
;       ---       FNING         Unused error flag
;
; MODIFICATION LOG:
;
;       30-DEC-92       Fixed white alarm print message.        M. Teats
;
;       07-MAY-93       Delete KY-EA1-1, move CI-EC1-U1 to              M. Teats
;                               Unit EC1.
;
;       07-JUN-93       Added comments and baselined for        M. Teats
;                       D/3 version 3.3.  CCN# 93JUN01-01.
;
;       15-DEC-94       Revised header to indicate the new VMS          J. Berger
;                       system. Rebase line document. Removed
;                       obsolite Select() statements.
```

```
56          ;
57          ;  END MOD LOG
58          ;
59          UNIT    EA1
60          ;
61                  COMMON WHOLE0
62                  FLAGBITS == PSPAR0
63                  FVEA1 == 'FV-EA1-1':DEV_TMP0
64                  INLK1 == 'EA1-INLK1':DEV_TMP0
65                  INLK2 == 'EA1-INLK2':DEV_TMP0
66                  G_bits == PB2GLOBAL:DEV_TMP0
67                  STAT_BITS == 'FV-EA1-1':DEV_STAT
68          ;
69   0000           NOP
70   0001           LET STAR = 34                    ; Start message
71   0002           LET SKDKEY = 33          ; SKID Key
72   0003           LET UNITT = 9                    ; SEQ Unit number
73   0004           ASSIGN PDEV = 'EA1-ALARM'        ; Alarm device
74   0005           ASSIGN PTYPE = TYPE34
75   0006           LET ADI = 0                      ; Alarm destination index
76   0007           LET FSN0 = 0                     ; No alarms yet
77   0008           TIMEROFF STIME0          ; Alarm timer off
78   0009           TIMERCLR STIME0          ; Alarm timer clear
79   0010           LET STRNG0 = " "                 ; Blank 30 character string
80   0011           LET ERRN = 0                     ; Set error number to zero
81   0012           LET ERRL = 0                     ; Set error line to zero
82   0013           GETUNIT TRIP                     ; Which sequence unit is program running in
83   0014           IF TRIP != UNITT THEN            ; Abort if running in wrong seq. unit
84                     BEGIN
85   0015              FSTART = 1
86   0016              ILOOP = STAR + 1
87   0017              PTYPE = 0
88   0018              GOSUB ACKK
89   0019              GOTO TERM
90                     END
91          ;       IF !SELECT() THEN GOTO AGAIN
92          ;
93          ;       B E G I N   I N I T I A L I Z I N G
94          ;
95   0020   INIT:   LET FABORT = 0                   ; If FABORT = 1 then abort program
96   0021           LET FSTART = 0                   ; Program start flag = 0
97   0022           LAMP ADI,SKDKEY,0                ; Turn off annunciator key
98   0023           LET LSG0 = " "                   ; Clear the string variables
99   0024           LET LSG1 = " "
100  0025           LET LSG2 = " "
101  0026           LET LSG3 = " "
102  0027           LET LSG4 = " "
103  0028           PDEV:DEV_TMP0 = 1                          ; Set alarm device to non-alarm state
104  0029           IF AND(PDEV:DEV_ALMCT,16) THEN             ; If alarm device is inhibited, remove the inhibit
105  0030             PDEV:DEV_ALMCT=XOR(PDEV:DEV_ALMCT,16)
106  0031           IF AND(PDEV:DEV_ALMCT,32) THEN             ; If alarm device is ack, clear it
107  0032              PDEV:DEV_ALMCT = XOR(PDEV:DEV_ALMCT,32)
108         ;
109         ;       First, set all flags and integers:
110         ;
```

```
111  0033          ASSIGN PDIN = 'PSL-EA1-1'                    ; Assign pointer to first DIN
112  0034          ASSIGN PSTAT = INT0
113  0035          ASSIGN PTYPE = TYPE0
114  0036          ASSIGN PREV = FRV0
115  0037          ASSIGN PSKID = SKID0
116  0038          ASSIGN PPASS = FPAS0
117           ;
118  0039          FOR ILOOP = 0 TO 29
119  0040          LET PPASS = 0
120  0041          LET PREV = 0
121  0043          IF PSTAT > 5 THEN PSTAT = 0                  ; Reset current status if undefined
122  0044          LET PSKID = 0                                ; Default previous status to no alarms
123  0045          LET PTYPE = 0                                ; Default to no alarms
124  0046          IF ILOOP < 3 THEN                            ; For DINs
125                   BEGIN
126  0047             LET PTYPE = 0                             ; Default to no alarm
127  0048             IF AND(PDIN:DIN_DFLG,3) THEN              ; Check for alarms if defined (must be red or yellow)
128                        BEGIN
129  0049                  LET PTYPE = 2                        ; Default to yellow alarm
130  0051                  IF AND(PDIN:DIN_DFLG,1) THEN LET PREV = 1
                                                               ; Reverse acting
131  0053                  IF AND(PDIN:DIN_DFLG,4) THEN LET PTYPE = 4   ; Set to red critical alarm
132                        END
133                   END
134  0054          ELSE LET PTYPE = 0                           ; Set to no alarm
135  0056          IF ILOOP < 2 THEN STEP PDIN BY 1          ; Point to next DIN
136  0058          IF ILOOP > 23 THEN LET PTYPE = 1          ; Set to white alarm
137  0059          STEP PPASS BY 1
138  0060          STEP PREV BY 1
139  0061          STEP PSTAT BY 1
140  0062          STEP PSKID BY 1
141  0063          STEP PTYPE BY 1
142  0064          NEXT ILOOP
143           ;
144  0065          LET TYPE0 = 1
145  0066          LET TYPE3 = 4
146  0067          LET TYPE4 = 4
147  0068          LET TYPE5 = 4
148  0069          LET TYPE6 = 4
149  0070          LET TYPE7 = 4
150  0071          LET TYPE11 = 2
151  0072          LET TYPE20 = 2
152           ;
153  0073          LET FRV0 = 0
154  0074          LET FRV1 = 0
155  0075          LET FRV2 = 1
156  0076          LET FRV24 = 1
157  0077          LET FRV25 = 1
158  0078          LET FRV26 = 1
159  0079          LET FRV27 = 1
160           ;
161  0080          LET AI0 = 75.0
162  0081          LET AI1 = 80.0
163  0082          LET AI2 = 3000.0
164  0083          LET AI3 = 17.7
165  0084          LET AI4 = 29.7
```

```
166  0085          LET AI5 = 190.0
167          ;
168  0086          LET AI19 = 0.0                ; For FIC-EA1-1 output (O/C Display Bar)
169          ;
170          ;     E N D   I N I T I A L I Z I N G
171          ;
172          ;     S E N D   P R O G R A M   S T A R T   M E S S A G E
173          ;
174  0087          ANALOG = 0
175  0088          ILOOP = STAR
176  0089          FSTART = 1
177  0090          PTYPE = 0
178  0091          GOSUB ACKK
179  0092          ON ERROR GOSUB ERRSUB
180          ;
181          ;     B E G I N   P R O G R A M   L O O P
182          ;
183  0093  AGAIN: MARK STEP "WTUNTL",0,0
184          ;     WAITUNTIL SELECT() ¦ FABORT   ; Wait until micro is selected or program aborted
185          ;     IF (FABORT) THEN GOTO TERM    ; Terminate program if abort flag set
186  0094          MARK STEP "RNINNG",0,0
187  0095          WAIT 1
188  0096          IF (FSNO) THEN                ; If alarm flag is set
189  0097             IF STIMEO > 60 THEN        ; If alarm flag active for 1 minute
190                                             ; Reset timer and turn off flag
191  0098                BEGIN
192  0099                TIMEROFF STIMEO
193  0100                TIMERCLR STIMEO
194  0101                LET FSNO = 0
195  0102                LET ERRN = 0
196                      LET ERRL = 0
197                      END
198          ;     If EA1-ALARM $T1 is alarmed and it's ALMCT indicates alarms are acknowledged:
199          ;
200  0103          IF (PDEV:DEV_TMPO = 2) & AND(PDEV:DEV_ALMCT,32) THEN
201                   BEGIN
202  0104             LET PDEV:DEV_TMPO = 1 ; Put EA1-ALARM in ack state
203  0105             LET FTRIP4 = 1             ; Set for unit alarm ack, system to graphic
204  0106             LET ACK = 1                ; Set for unit alarm ack, graphic to system
205  0107             LET FACK = 1
206  0108             CRIT
207  0109             LAMP AD1,SKDKEY,1          ; Turn lamp on
208                   END
209          ;
210          ;     BEGIN DINs:
211          ;
212  0110          ASSIGN PDIN = 'PSL-EA1-1'
213  0111          ASSIGN PTYPE = TYPEO
214  0112          ASSIGN PREV = FRVO
215  0113          ASSIGN PSTAT = INTO
216  0114          ASSIGN PPASS = FPASO
217  0115          LET CURENT = 0                ; Used to set lamp status
218  0116          LET LAST = 0
219  0117          LET ANALOG = 0
220  0118          LET NXT = 0
```

```
221  0119          LMPVAR = 0
222  0120          BITVAL = 0
223           ;
224  0121     LOP: FOR ILOOP = 0 TO 2
225           ;    IF !SELECT() THEN GOTO AGAIN
226  0122          IF ( PTYPE > 0 ) THEN                                  ; If some type of alarm
227                   BEGIN                                             ;be1
228  0124             IF AND(PDIN:DIN_ALMCT,16) THEN LET PSTAT = 3       ; If EPN inhibited
229  0126             IF AND(PDIN:DIN_ALMCT,4) THEN LET PSTAT = 5        ; If EPN is bad input
230  0128             IF !AND(PDIN:DIN_FLG,8) THEN LET PSTAT = 4         ; If EPN is disabled
231  0129             IF ( PSTAT < 3 ) THEN                             ; If not inhibited
232                        BEGIN                                        ;be2
233  0130                  IF (PSTAT = 1) THEN
234                           BEGIN                                     ;be3
235  0131                     IF (PTYPE != 1) THEN
236  0133                        IF AND(PDIN:DIN_ALMCT,32) THEN GOSUB ACK1
237  0135                     IF (FTRIP4) THEN GOSUB ACK1
238                           END                                       ;en3
239  0138                  IF PSTAT = 0 THEN IF(PDIN:DIN_VAL = PREV) THEN GOSUB ALM
240  0139                  IF (PSTAT) THEN
241                              BEGIN                                  ;be4
242  0141                        IF (PDIN:DIN_VAL != PREV) THEN GOSUB CLR
243  0142                        NXT = PSTAT
244  0144                        IF (PPASS) & (PSTAT = 2) THEN NXT = 0
245  0145                        LAST = OR(LAST,NXT)
246                              END                                    ;en4
247                        END                                          ;en2
248  0147              ELSE IF !AND(PDIN:DIN_ALMCT,16)&!AND(PDIN:DIN_ALMCT,4)&AND(PDIN:DIN_FLG,8)THEN LET PSTAT = 0
249                    END                                             ;en1
250  0148          STEP PDIN BY 1
251  0149          STEP PTYPE BY 1
252  0150          STEP PREV BY 1
253  0151          STEP PSTAT BY 1
254  0152          STEP PPASS BY 1
255  0153          NEXT ILOOP
256           ;
257           ;    END DINs
258           ;
259           ;    BEGIN ANALOG RED AND YELLOWS:
260           ;
261  0154          ASSIGN PAI = 'TI-EA1-7'
262  0155          ASSIGN PSKID = SKID3
263  0156          FOR JLOOP = 3 TO 23
264           ;    IF !SELECT() THEN GOTO AGAIN
265  0157          IF (PTYPE) THEN                                       ; If some alarm exists
266                   BEGIN                                  ;be1
267  0158             BITVAL = PAI:AI_ALMCT
268  0159             IF !AND(BITVAL,16) THEN                           ; If not inhibited
269                      BEGIN                              ;be2
270  0160                IF AND(BITVAL,15) THEN                         ; If alarm exists and is current
271                         BEGIN                           ;be3
272  0162                   IF AND(BITVAL,32) THEN NXT = 2              ; If alarm is ack
273                         ELSE                                        ; If alarm is unack
274                            BEGIN                        ;be4
275  0163                      IF (PSKID !=BITVAL) THEN
```

```
276                                 BEGIN                       ;be5
277    0164                         LET PSKID = BITVAL
278    0166                         IF BITVAL < 3 THEN FALMON = 1
279    0167                         PDEV:DEV_TMP0 = 2
280    0168                         NXT = 1
281                                 END                         ;en5
282    0169                      ELSE NXT = 1
283                               END                           ;en4
284                            END                              ;en3
285                         ELSE                                        ; Else there is no alarm
286                            BEGIN
287    0170                    NXT = 0
288    0171                    LET PSKID = 0
289                            END
290                         END                                 ;en2
291                      ELSE                                           ; Else alarm is inhibited
292                         BEGIN
293    0172                 NXT = 0
294    0173                 LET PSKID = 0
295                         END
296    0174                 ANALOG = OR(ANALOG,NXT)
297                         END                                 ;en1
298    0175              STEP PREV BY 1
299    0176              STEP PSTAT BY 1
300    0177              STEP PSKID BY 1
301    0178              STEP PTYPE BY 1
302    0179              STEP PAI BY 1
303    0180              NEXT JLOOP
304            ;
305            ;        END ANALOG RED AND YELLOWS
306            ;
307            ;        BEGIN ANALOG WHITE ALARMS:
308            ;
309    0181    WHT:     WHITE = 0
310    0182             ASSIGN PALARM = AI0
311    0183             ASSIGN PREV = FRV24
312    0184             ASSIGN PSTAT = INT24
313    0185             ASSIGN PTYPE = TYPE24
314            ;
315    0186             ASSIGN PAI = 'PI-EA1-13'
316    0187             ILOOP = 24
317    0188             GOSUB ANLG                                  ; LOLO
318    0189             GOSUB ANLG                                  ; LO
319    0190             ASSIGN PAI = 'FIC-EA1-1'
320    0191             GOSUB ANLG                                  ; LO
321    0192             ASSIGN PAI = 'PI-EA1-1'
322    0193             GOSUB ANLG                                  ; LO
323    0194             GOSUB ANLG                                  ; HI
324    0195             ASSIGN PAI = 'TI-EA1-5'
325    0196             GOSUB ANLG                                  ; HI
326            ;
327    0198             IF(INT24 = 3 ¦ INT25 = 3)THEN LET LEAK0 = 9
328    0200               ELSE IF(INT24)THEN LET LEAK0 = INT24
329    0202                 ELSE IF(INT25)THEN LET LEAK0 = INT25 + 2
330    0203                   ELSE LET LEAK0 = 0
```

```
331  0205          IF(INT27 = 3 ¦ INT28 = 3)THEN LET LEAK1 = 9
332  0207            ELSE IF(INT27)THEN LET LEAK1 = INT27 + 2
333  0209              ELSE IF(INT28)THEN LET LEAK1 = INT28 + 4
334  0210                ELSE LET LEAK1 = 0
335  0211          FTRIP4 = 0
336           ;
337           ;    END ANALOG WHITE ALARMS
338           ;
339  0212          MARK STEP "BELOW",0,0
340  0214          IF LAST > 3 THEN PRINT ADI," LAMP VARIABLE IS GREATER THAN 3!"
341  0215          CURENT = OR(LAST,WHITE)
342  0216          CURENT = OR(CURENT,ANALOG)
343  0218          IF AND(CURENT,1) THEN LMPVAR = 2
344  0220             ELSE IF AND(CURENT,2) THEN LMPVAR = 1
345  0221                     ELSE LMPVAR = 0
346  0222          LAMP ADI,SKDKEY,LMPVAR
347  0223          MARK STEP "BOTTOM",0,0
348           ;
349           ;    BEGIN LOGIC FOR GRAPHICS:
350           ;
351  0224          NOCRIT
352  0225          ASSIGN FLAGBITS = AT2
353  0226          BVAL = 4
354  0227              FOR IN1 = 2 TO 13                    ; $T2 TO $T13
355  0228              FLAGBITS = AND(FVEA1,BVAL)
356  0229              BVAL = BVAL  ( 1
357  0230              STEP FLAGBITS BY 1
358  0231              NEXT IN1
359  0232          ASSIGN FLAGBITS = SA0
360  0233          BVAL = 1
361  0234             FOR IN1 = 0 TO 4
362  0235             FLAGBITS = AND(INLK2,BVAL)            ; EA1-INLK2
363  0236             BVAL = BVAL ( 1
364  0237             STEP FLAGBITS BY 1
365  0238             NEXT IN1
366  0239          SA5 = AND(INLK1,1)
367  0240          SA6 = AND(INLK1,2)
368  0241          SA7 = AND(INLK1,4)
369  0242          GT7 = AND(G_bits,128)
370  0243          GT12 = AND(G_bits,4096)
371  0244          GT14 = AND(G_bits,16384)
372           ;    AT10 = AND(FVEA1,2048)
373  0245          AT18 = AND(STAT_BITS,4)                  ; STEAM PERMISSIVE
374  0246          AT19 = AND(STAT_BITS,32)          ; FV-EA1-1 INTERLOCK
375  0247          AT20 = AND('RSH-CA1-1':DIN_VAL,'PSH-CA111':DIN_VAL)
376  0248          AT21 = AND('RSH-RC1-1':DIN_VAL,'RSH-EA1-1':DIN_VAL)
377  0249          AT22 = AND('PDSH-FC51':DIN_VAL,'RSH-VVB/G':DIN_VAL)
378  0250          AT23 = AND('TSH-DSH-3':DIN_VAL,AT20)
379  0251          AT24 = AND(AT20,AT21)
380  0252          AT25 = AND(AT22,AT23)
381  0253          AT26 = AND(AT24,AT25)
382  0254          AT27 = AND(AT26,AT18)
383  0255          SB0 = AND('EA1-INLK1':DEV_STAT,32)
384  0256          SB1 = AND('EA1-INLK2':DEV_STAT,32)
385  0257          SB2 = AND('TV-DSH1/2':DEV_STAT,32)
```

```
386  0258            BTO = AND(AT8,AT9)
387  0260            IF AND('EA1-ALARM':DEV_FLG,64) THEN PUTMANL 'EA1-ALARM'
388  0262            IF AND('FV-EA1-1':DEV_FLG,64) THEN PUTMANL 'FV-EA1-1'
389  0264            IF AND('EA1-INLK1':DEV_FLG,64) THEN PUTMANL 'EA1-INLK1'
390  0266            IF AND('EA1-INLK2':DEV_FLG,64) THEN PUTMANL 'EA1-INLK2'
391  0268            IF AND('TV-DSH1/2':DEV_FLG,64) THEN PUTMANL 'TV-DSH1/2'
392  0270            IF AND('SELECT-TI':DEV_FLG,64) THEN PUTMANL 'SELECT-TI'
393          ;
394          ;       END LOGIC FOR GRAPHICS
395          ;
396          ;       B E G I N   O / C   D I S P L A Y   B A R :
397          ;
398          ;       Function:  Upon loss of permissive, put the control loop in manual
399          ;                  and set the PID block output to zero.
400          ;
401  0272            IF AND('FV-EA1-1':DEV_TMP1,4) THEN DLEAK3 = 1                    ; Permissive ($T18) must be Hi (1)
402  0273            IF !AND('FV-EA1-1':DEV_TMP1,4) & AND(DLEAK3,1) THEN              ; Permissive ($T18) lost and 1st pass
                       BEGIN
404  0274              PUTMANL 'FIC-EA1-1'
405  0275              PRINT ADI," PERMISSIVE LOST >>> FIC-EA1-1 PUT IN MANUAL"
406  0276              PUTOUT 'FIC-EA1-1',AI19
407  0277              PRINT ADI," PERMISSIVE LOST >>> FIC-EA1-1 OUTPUT SET TO ZERO"
408  0278              DLEAK3 = 0
                       END
410          ;
411          ;            E N D   O / C   D I S P L A Y   B A R
412          ;
413  0279            NOCRIT
414  0280            GOTO AGAIN
415          ;
416          ;       E N D   P R O G R A M   L O O P
417          ;
418          ;       BEGIN ANALOG WHITE SUBROUTINE:
419          ;
420          ; ANLG:  IF !SELECT() THEN GOTO AGAIN
421  0281    ANLG:   IF (PTYPE = 1) THEN                              ; If some alarm
422                     BEGIN                                 ;be1
423  0282               BITVAL = PAI:AI_ALMCT
424  0283               IF !(AND(BITVAL,16)) THEN                     ; If not inhibited
425                       BEGIN                               ;be2
426  0285                 IF PSTAT = 3 THEN LET PSTAT = 0
427  0286                 IF (PREV) THEN                              ; If rev acting, LOLO, or LO
428                          BEGIN                            ;be3
429  0287                    IF (PSTAT = 0) THEN                      ; If not previously alarmed
430                             BEGIN                         ;be4
431  0289                       IF (PAI:AI_MEAS <= PALARM) THEN GOSUB ALM
432                             END                           ;en4
433                          ELSE
434                             BEGIN                         ;be5
435  0290                       IF (PSTAT = 1) THEN                   ; If previously alarmed
436                                BEGIN                      ;be6
437  0292                          IF (FTRIP4) THEN GOSUB ACK1        ; If ACK flag set, then ack it
438  0295                          IF PTYPE !=1 THEN IF AND(PAI:AI_ALMCT,32)THEN GOSUB ACK1
439                                END                        ;en6
440  0297                       IF (PAI:AI_MEAS >= PALARM + PAI:AI_DEAD) THEN GOSUB CLR
```

```
                              END                                 ;en5
                           END                                    ;en3
                        ELSE                                                ; If fail-safe, HIHI, or HI
                           BEGIN                                  ;be10
0298                       IF (PSTAT = 0) THEN
                              BEGIN                               ;be11
0300                          IF (PAI:AI_MEAS >= PALARM) THEN GOSUB ALM
                              END                                 ;en11
                           ELSE
                              BEGIN                               ;be12
0301                          IF (PSTAT = 1) THEN
                                 BEGIN                            ;be13
0303                             IF (FTRIP4) THEN GOSUB ACK1
0306                             IF PTYPE != 1 THEN IF AND(PAI:AI_ALMCT,32) THEN GOSUB ACK1
                                 END                              ;en13
0308                          IF (PAI:AI_MEAS <= PALARM - PAI:AI_DEAD) THEN GOSUB CLR
                              END                                 ;en12
                           END                                    ;en10
0309                    NXT = PSTAT
0311                    IF (PPASS) & (PSTAT = 2) THEN NXT = 0
                        END                                       ;en2
                     ELSE
                        BEGIN
0312                    LET PSTAT = 3
0313                    NXT = 0
                        END
0314                 WHITE = OR(WHITE,NXT)
                     END                                          ;en1
0315                 ELSE LET PSTAT = 0
0316                 ILOOP = ILOOP + 1
0317                 STEP PALARM BY 1
0318                 STEP PSTAT BY 1
0319                 STEP PTYPE BY 1
0320                 STEP PREV BY 1
0321                 STEP PPASS BY 1
0322                 RETURN
              ;
              ;      END ANALOG WHITE SUBROUTINE
              ;
              ;      BEGIN CLEAR, ALARM, ACKNOWLEDGE, MESSAGE, ERROR, AND TERMINATE SUBROUTINES:
              ;
0323    CLR:  LET PSTAT = 0                                                 ; Int set to zero
              ;
              ; If not white, then unack:
              ;
0326          IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
0328          IF ILOOP = 24 THEN INT25 = 0
0329          LET FCLEAR = 1                                                ; Clear flag
0330          GOSUB ACKK                                                    ; Print message
0331          RETURN
              ;
0332    ALM:  LET PSTAT = 1                                                 ; Int set to one
              ;
              ; If not white, then unack:
              ;
```

```
496          ;        IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
497                   ;
498                   ; Turn audible alarm on if white or yellow:
499                   ;
500     0333          CRIT
501     0335          IF AND(PTYPE,3) THEN FALMON = 1
502     0336          PDEV:DEV_TMP0 = 2
503     0337          GOSUB ACKK                                          ; Print message
504     0338          RETURN
505          ;
506     0339     ACK1:    LET PSTAT = 2                                  ; Ack the alarm
507     0340              LET FACK = 1                                   ; Set ACK flag for C program
508          ;
509          ;        ALARM TYPER STRINGS:
510          ;
511     0342     ACKK:    IF ILOOP = 0 THEN STRNG0 = "E-A-1 10 LB STEAM PRESSURE LO"
512     0344              IF ILOOP = 1 THEN STRNG0 = "DESUPER HEAT OUTLET TEMP HI  "
513     0346              IF ILOOP = 24 THEN STRNG0 = "PI-EA1-13 STM HDR PRESS LOLO"
514     0348              IF ILOOP = 25 THEN STRNG0 = "PI-EA1-13 STM HDR PRESS LO  "
515     0350              IF ILOOP = 26 THEN STRNG0 = "FIC-EA1-1 REBOIL STM FLO LOW"
516     0352              IF ILOOP = 27 THEN STRNG0 = "PI-EA1-1 REBOIL STM PRESS LO"
517     0354              IF ILOOP = 28 THEN STRNG0 = "PI-EA1-1 REBOIL STM PRESS HI"
518     0356              IF ILOOP = 29 THEN STRNG0 = "TI-EA1-5 STM CNDNS8 TEMP HI "
519     0358              IF ILOOP = 34 THEN STRNG0 = "START EA1BATCH PGM "
520     0360              IF ILOOP = 35 THEN STRNG0 = "EA1BATCH WRONG UNIT"
521     0362              IF ILOOP = 36 THEN STRNG0 = "TERMINATE EA1BATCH "
522          ;
523          ;  If acknowledging the alarm, then:
524          ;
525     0363     AMSG:    IF PTYPE = 1 THEN
526                          BEGIN
527                          ;
528                          ; If white alarm, print:
529                          ;
530     0365                 IF (FACK = 0) & (FCLEAR = 0) & (FSTART = 0) THEN PRINT ADI,"WHITE ALARM %p",STRNG0
531     0366                 IF (FACK) THEN
532                            BEGIN
533                            ;
534                            ; If white, print it is ack and reset the flag:
535                            ;
536     0367                   PRINT ADI,"%p ALARM ACKNOWLEDGE",STRNG0
537     0368                   LET FACK = 0
538                            END
539                          END
540          ;
541          ;  If CLEAR, print it is clear and reset the flag:
542          ;
543     0369               IF (FCLEAR) THEN
544                          BEGIN
545     0370                 PRINT ADI,"%p ALARM CLEAR",STRNG0
546     0371                 LET FCLEAR = 0
547                          END
548          ;
549          ;  If STARTUP, print startup message:
550          ;
```

```
551   0372            IF (FSTART) THEN
552                      BEGIN
553   0373               LET FSTART = 0
554   0374               PRINT ADI,"%p",STRNGO
555                      END
556   0375            RETURN
557          ;
558          ;  If ERROR - print message, start timer, and set flag:
559          ;
560   0376   ERRSUB:  IF ERRN != ERRNUM() THEN
561                      BEGIN
562   0377               TIMERON STIMEO
563   0378               TIMERCLR STIMEO
564   0379               LET FSNO = 1
565   0380               LET ERRN = ERRNUM()
566   0381               LET ERRL = ERRLIN()
567   0382               PRINT ADI,"ERROR # %i AT LINE %i",errn,errl
568                      END
569   0383               ERSCLR
570   0384            GOTO AGAIN
571          ;
572          ;  If FABORT = 1, then abort program:
573          ;
574          ; TERM:  LET FABORT = 0
575   0385   TERM:    LET STRNGO = "EA1BATCH TERMINATED    "
576   0386            LET ILOOP = STAR + 2
577   0387            LET FSTART = 1
578   0388            GOSUB ACKK
579                   END

      Total Errors Detected      0     Total Warnings     0
          Sequence Program Size = 7456 Bytes, 466 Pages
               24 Forward Reference(s)
```

EC1BATCH

```
1           ;       E C 1 B A T C H . S E Q                                 VERSION 2
2           ;
3           ;       Function:       Sequence program for Lamp function,
4           ;                       message display on Graphic number 016,
5           ;                       and printer messages.
6           ;
7           ;       Written By      : R. N. Kersey
8           ;       Date            : 05-31-92
9           ;       Facility        : 242-A Evaporator Bldg.
10          ;       Machine         : VAX 4104-A
11          ;       Op. System      : VAX/VMS 5.5-2AH
12          ;       D/3 Version     : 6.3-2 BASELINE
13          ;       Compiler        : SIC
14          ;       Build command   : @[320,20]EC1BATCH
15          ;
16          ;       UNIT = 10, SKDKEY = 38
17          ;
18          ;       Assoc. UPF:     [300,1]_PRCESS.___      PRCESS.LST
19          ;       Assoc. RPF:     [300,1]_242A.___        242A.LST
20          ;       SDB:            [300,1]PCMOA.SDB        PCMOASDB.CMD
21          ;
22          ;       POINTER   SDB VAR.      FUNCTION
23          ;       -------   ---------     ----------------------------------------
24          ;       PSTAT     INTn          Current EPN alarm state is an integer tied to
25          ;                               graphic flashing string of value:
26          ;       .         (n=0-100)             0 = previously not alarmed
27          ;       .                               1 = previously alarmed, not ack
28          ;       .                               2 = previously alarmed, is ack
29          ;       .                               3 = alarm is inhibited
30          ;       .                               4 = input is off scan
31          ;       .                               5 = input is bad
32          ;       .         LEAKn         Special INT variable for multiple white alarms on an EPN
33          ;       PDIN      'EPN'         ---
34          ;       PTYPE     TYPEn         alarm type 0 = none, 1 = white, 2 = yellow,
35          ;                                  3 = yellow and yellow, 4 = red,
36          ;                                  6 = yellow and red
37          ;       PREV      FRVn          Reverse acting (not fail safe) flag (1=set)
38          ;       PSKID     SKIDn         Used for analog alarms
39          ;       PPASS     FPASn         Turn lamp off when alarm is ACKnowledged
40          ;       PALARM    FALMn         Analog alarm flag
41          ;       ---       FNING         Unused error flag
42          ;
43          ;  MODIFICATION LOG:
44          ;
45          ;        09-DEC-92      Changed FIC-EC3-1 LO alarm from 140 to 50        M. Teats
46          ;                       and added comments for FIC-EC3-1.
47          ;
48          ;        30-DEC-92      Fixed white alarm print message.         M. Teats
49          ;
50          ;        07-JUN-93      Added comments.  CCN# 93JUN01-01.                M. Teats
51          ;
52          ;-mct1+  21-JUN-93      Changed TYPE25 = 0 to TYPE25 = 1.  Baselined      M. Teats
53          ;                       for D/3 version 3.3.  CCN# 93JUN21-01.
54          ;
55          ;        15-DEC-94      Revised header to indicate the new VMS           J. Berger
```

```
56          ;                       system. Rebaselined document. Removed
57          ;                       obsolict Select() statement.
58          ;
59          ;  END MOD LOG
60          ;
61          UNIT     EC1
62          ;
63                   COMMON WHOLE0
64                   HVEC1 == 'HV-EC1-1':DEV_TMP1
65                   HVEC2 == 'HVEC2/3-1':DEV_TMP1
66                   FLAGBITS == PSPAR0
67                   G_bits == PB2GLOBAL:DEV_TMP0
68          ;
69   0000            NOP
70   0001            STAR = 34                          ; Start message
71   0002            LET SKDKEY = 38          ; SKID Key
72   0003            LET UNITT = 10                     ; SEQ Unit number
73   0004            ASSIGN PDEV = 'EC1-ALARM'          ; Alarm device
74   0005            ASSIGN PTYPE = TYPE34
75   0006            LET ADI = 0                        ; Alarm destination index
76   0007            LET FSN0 = 0                       ; No alarm yet
77   0008            TIMEROFF STIME0          ; Alarm timer off
78   0009            TIMERCLR STIME0          ; Alarm timer clear
79   0010            LET STRNG0 = " "                   ; Blank 30 character string
80   0011            LET ERRN = 0                       ; Set error number to zero
81   0012            LET ERRL = 0                       ; Set error line to zero
82   0013            LET FTRIP4 = 0                     ; FTRIP4 = 1 when audible horn is to be turned on
83   0014            GETUNIT TRIP                       ; Which sequence unit is program running in
84   0015            IF TRIP != UNITT THEN              ; Abort if running in wrong seq. unit
85                      BEGIN
86   0016               FSTART = 1
87   0017               ILOOP = STAR + 1
88   0018               PTYPE = 0
89   0019               GOSUB ACKK
90   0020               GOTO TERM
91                      END
92          ;        IF !SELECT() THEN GOTO AGAIN
93          ;
94          ;        B E G I N   I N I T I A L I Z I N G
95          ;
96   0021   INIT:    LET FABORT = 0                     ; If FABORT = 1 then abort program
97   0022            LET FSTART = 0                     ; Program start flag = 0
98   0023            LAMP ADI,SKDKEY,0                  ; Turn off the annunciator key
99   0024            LET LSG0 = " "                     ; Clear the string variables
100  0025            LET LSG1 = " "
101  0026            LET LSG2 = " "
102  0027            LET LSG3 = " "
103  0028            LET LSG4 = " "
104  0029            PDEV:DEV_TMP0 = 1                                ; Set alarm device to non-alarm state
105  0030            IF AND(PDEV:DEV_ALMCT,16) THEN                   ; If alarm device is inhibited, remove the inhibit
106  0031              PDEV:DEV_ALMCT=XOR(PDEV:DEV_ALMCT,16)
107  0032            IF AND(PDEV:DEV_ALMCT,32) THEN                   ; If alarm device is ack, clear it
108  0033               PDEV:DEV_ALMCT = XOR(PDEV:DEV_ALMCT,32)
109         ;
110         ;        First, set all flags and integers:
```

```
111             ;
112     0034            ASSIGN PDIN = 'RSH-CA1-1'                       ; Assign pointer to first DIN
113     0035            ASSIGN PSTAT = INT0
114     0036            ASSIGN PTYPE = TYPE0
115     0037            ASSIGN PREV = FRV0
116     0038            ASSIGN PSKID = SKID0
117     0039            ASSIGN PPASS = FPAS0
118             ;
119     0040            FOR ILOOP = 0 TO 25
120     0041            PPASS = 0
121     0042            PREV = 0
122     0044            IF PSTAT > 5 THEN PSTAT = 0                     ; Reset currnet status if undefined
123     0045            PSKID = 0                                       ; Default previous status to no alarms
124     0046            PTYPE = 0                                       ; Default to no alarms
125     0047            IF ILOOP = 0 THEN                               ; For the only DIN
126                        BEGIN
127     0048               IF AND(PDIN:DIN_DFLG,3) THEN                 ; Check for alarms if defined (must be red or yellow)
128                           BEGIN
129     0049                  LET PTYPE = 2                             ; Default to yellow alarm
130     0051                  IF AND(PDIN:DIN_DFLG,1) THEN LET PREV = 1 ; Reverse acting
131     0053                  IF AND(PDIN:DIN_DFLG,4) THEN LET PTYPE = 4 ; Set to red critical alarm
132                           END
133                        END
134     0055            IF ILOOP > 22 THEN PTYPE = 1                    ; Set to white alarm
135     0056            STEP PPASS BY 1
136     0057            STEP PREV BY 1
137     0058            STEP PSTAT BY 1
138     0059            STEP PSKID BY 1
139     0060            STEP PTYPE BY 1
140     0061            NEXT ILOOP
141             ;
142     0062            TYPE19 = 4
143     0063            TYPE23 = 1              ; PI-EC1-5
144     0064            TYPE24 = 1              ; FIC-EC3-1
145     0065            TYPE25 = 1              ; TIC-EC1-2   +mct1-
146             ;
147     0066            FRV0 = 0
148     0067            FRV1 = 0
149     0068            FRV24 = 1               ; FIC-EC3-1 LO
150             ;
151     0069            AI0 = 100.0             ; PI-EC1-5  HI
152     0070            AI1 = 50.0              ; FIC-EC3-1 LO
153     0071            AI2 = 110.0             ; TIC-EC1-2 HI
154             ;
155             ;       E N D   I N I T I A L I Z I N G
156             ;
157             ;       S E N D   P R O G R A M   S T A R T   M E S S A G E
158             ;
159     0072            ANALOG = 0
160     0073            ILOOP = STAR
161     0074            FSTART = 1
162     0075            PTYPE = 0
163     0076            GOSUB ACKK
164     0077            ON ERROR GOSUB ERRSUB
165             ;
```

```
166          ;       B E G I N   P R O G R A M   L O O P
167          ;
168    0078  AGAIN:  MARK STEP "WTUNTL",0,0
169          ;       WAITUNTIL SELECT() | FABORT      ; Wait until micro is selected or program aborted
170          ;       IF (FABORT) THEN GOTO TERM       ; Terminate program if abort flag set
171    0079          MARK STEP "RNNING",0,0
172    0080.         WAIT 1
173    0081          IF (FSNO) THEN                   ; If alarm flag is set
174    0082             IF STIMEO > 60 THEN           ; If alarm flag active for 1 minute
175                         BEGIN                     ; Reset timer and turn off flag
176    0083                 TIMEROFF STIMEO
177    0084                 TIMERCLR STIMEO
178    0085                 LET FSNO = 0
179    0086                 LET ERRN = 0
180    0087                 LET ERRL = 0
181                         END
182          ;
183          ;       If EC1-ALARM $T1 is alarmed and it's ALMCT indicates alarms are acknowledged:
184          ;
185    0088          IF (PDEV:DEV_TMPO = 2) & AND(PDEV:DEV_ALMCT,32) THEN
186                   BEGIN
187    0089           PDEV:DEV_TMPO = 1               ; Put EC1-ALARM in ack state
188    0090           LET FTRIP4 = 1                  ; Set for unit alarm ack, system to graphic
189    0091           LET ACK = 1                     ; Set for unit alarm ack, graphic to system
190    0092           CRIT
191    0093           LAMP ADI,SKDKEY,1               ; Turn lamp on
192                   END
193          ;
194          ;       BEGIN DINs:
195          ;
196    0094          ASSIGN PDIN = 'RSH-CA1-1'
197    0095          ASSIGN PTYPE = TYPEO
198    0096          ASSIGN PREV = FRVO
199    0097          ASSIGN PSTAT = INTO
200    0098          ASSIGN PPASS = FPASO
201    0099          LET CURENT = 0                   ; Used to set lamp status
202    0100          LET LAST = 0
203    0101          LET ANALOG = 0
204    0102          LET NXT = 0
205    0103          LMPVAR = 0
206    0104          BITVAL = 0
207          ;
208    0105  LOP:    IF ( PTYPE > 0 ) THEN                                    ; If some type of alarm
209                      BEGIN                                              ;be1
210    0107              IF AND(PDIN:DIN_ALMCT,16) THEN LET PSTAT = 3       ; If EPN inhibited
211    0109              IF AND(PDIN:DIN_ALMCT,4) THEN LET PSTAT = 5                ; If EPN is bad input
212    0111              IF !AND(PDIN:DIN_FLG,8) THEN LET PSTAT = 4                 ; If EPN is disabled
213    0112              IF ( PSTAT < 3 ) THEN                              ; If not inhibited
214                           BEGIN                                         ;be2
215    0113                   IF (PSTAT = 1) THEN
216                              BEGIN                                      ;be3
217    0114                      IF (PTYPE != 1) THEN
218    0116                         IF AND(PDIN:DIN_ALMCT,32) THEN GOSUB ACK1
219    0118                      IF (FTRIP4) THEN GOSUB ACK1
220                              END                                        ;en3
```

```
221  0121                  IF PSTAT = 0 THEN IF(PDIN:DIN_VAL = PREV) THEN GOSUB ALM
222  0122                  IF (PSTAT) THEN
223                              BEGIN                                    ;be4
224  0124                        IF (PDIN:DIN_VAL != PREV) THEN GOSUB CLR
225  0125                        NXT = PSTAT
226  0127                        IF (PPASS) & (PSTAT = 2 ) THEN NXT = 0
227  0128                        LAST = OR(LAST,NXT)
228                              END                                      ;en4
229                         END                                           ;en2
230  0130               ELSE IF !AND(PDIN:DIN_ALMCT,16)&!AND(PDIN:DIN_ALMCT,4)&AND(PDIN:DIN_FLG,8)THEN LET PSTAT = 0
231                      END                                             ;en1
232  0131            STEP PDIN BY 1
233  0132            STEP PTYPE BY 1
234  0133            STEP PREV BY 1
235  0134            STEP PSTAT BY 1
236  0135            STEP PPASS BY 1
237          ;
238          ;       END DINs
239          ;
240          ;       BEGIN ANALOG RED AND YELLOWS:
241          ;
242  0136            ASSIGN PAI = 'TI-CA1-9'
243  0137            ASSIGN PSKID = SKID1
244  0138            FOR JLOOP = 1 TO 22
245          ;       IF !SELECT() THEN GOTO AGAIN
246  0139            IF (PTYPE) THEN                                   ; If some alarm exists
247                    BEGIN                                  ;be1
248  0140              BITVAL = PAI:AI_ALMCT
249  0141              IF !AND(BITVAL,16) THEN                         ; If not inhibited
250                        BEGIN                              ;be2
251  0142                  IF AND(BITVAL,15) THEN                      ; If alarm exists and is current
252                              BEGIN                        ;be3
253  0144                        IF AND(BITVAL,32) THEN NXT = 2        ; If alarm is ack
254                              ELSE                                  ; If alarm is unack
255                                   BEGIN                   ;be4
256  0145                             IF PSKID !=BITVAL THEN
257                                        BEGIN              ;be5
258  0147                                  IF BITVAL < 3 THEN FALMON = 1
259  0148                                  LET PSKID = BITVAL
260  0149                                  PDEV:DEV_TMP0 = 2
261  0150                                  NXT = 1
262                                        END                ;en5
263  0151                             ELSE NXT = 1
264                                   END                     ;en4
265                              END                          ;en3
266                         ELSE                                      ; Else there is no alarm
267                              BEGIN
268  0152                        NXT = 0
269  0153                        LET PSKID = 0
270                              END
271                         END                               ;en2
272                    ELSE                                           ; Else alarm is inhibited
273                         BEGIN
274  0154                   NXT = 0
275  0155                   LET PSKID = 0
```

```
276                        END
277    0156              ANALOG = OR(ANALOG,NXT)
278                      END                                    ;en1
279    0157            STEP PREV BY 1
280    0158            STEP PSTAT BY 1
281    0159            STEP PSKID BY 1
282    0160            STEP PTYPE BY 1
283    0161            STEP PAI BY 1
284    0162            NEXT JLOOP
285            ;
286            ;       END ANALOG RED AND YELLOWS
287            ;
288            ;       BEGIN ANALOG WHITE ALARMS:
289            ;
290    0163            WHITE = 0
291    0164            ASSIGN PALARM = AI0
292    0165            ASSIGN PREV = FRV23
293    0166            ASSIGN PSTAT = INT23
294    0167            ASSIGN PTYPE = TYPE23
295            ;
296    0168            ASSIGN PAI = 'PI-EC1-5'
297    0169            ILOOP = 23
298    0170            GOSUB ANLG
299    0171            ASSIGN PAI = 'FIC-EC3-1'
300    0172            GOSUB ANLG
301    0173            ASSIGN PAI = 'TIC-EC1-2'
302    0174            GOSUB ANLG
303            ;
304            ;       END ANALOG WHITE ALARMS
305            ;
306    0175            MARK STEP "BELOW",0,0
307    0176            FTRIP4 = 0
308    0178            IF LAST > 3 THEN PRINT ADI," LAMP VARIABLE IS GREATER THAN 3!"
309    0179            CURENT = OR(LAST,WHITE)
310    0180            CURENT = OR(CURENT,ANALOG)
311    0182            IF AND(CURENT,1) THEN LMPVAR = 2
312    0184               ELSE IF AND(CURENT,2)THEN LMPVAR = 1
313    0185                       ELSE LMPVAR = 0
314    0186            LAMP ADI,SKDKEY,LMPVAR
315    0187            MARK STEP "BOTTOM",0,0
316    0188            NOCRIT
317            ;
318            ;       BEGIN LOGIC FOR GRAPHICS:
319            ;
320    0189            AT25 = AND(HVEC1,512)
321    0190            AT26 = AND(HVEC1,1024)
322    0191            AT27 = AND(HVEC1,2048)
323    0192            AT28 = AND(HVEC1,4096)
324    0193            AT29 = AND(HVEC1,8192)
325    0194            AT31 = AND(HVEC1,32768)
326            ;
327    0195            SA0 = 'ZS-EC1-1N':DIN_VAL
328    0196            SA1 = 'ZS-EC1-1C':DIN_VAL
329            ;
330    0197            SA4 = and('HV-EC1-1':DEV_STAT,4) ; RESET
```

```
331   0198          SA5 = AND('HV-EC1-1':DEV_STAT,16)          ; CHGSTATE
332         ;
333   0199          BT27 = AND(HVEC2,2048)                     ; START TIMER
334   0200          BT28 = AND(HVEC2,4096)                     ; TIMER OVER
335   0201          BT29 = AND(HVEC2,8192)                     ; EX-C-1 OFF
336   0202          BT31 = AND(HVEC2,32768)                    ; INTERLOCK
337         ;
338   0203          SA2 = 'ZSEC2/31N':DIN_VAL
339   0204          SA3 = 'ZSEC2/31C':DIN_VAL
340         ;
341   0205          SB0 = AND('HVEC2/3-1':DEV_STAT,4)
342   0206          SB1 = AND('HVEC2/3-1':DEV_STAT,16)
343   0207          SB2 = AND('HVEC2/3-1':DEV_STAT,32)
344         ;
345   0208          AT3 =  AND('RSH-VVB/G':DIN_VAL,'TSH-DSH-3':DIN_VAL)
346   0209          AT4 =  AND('WFSH-CA11':DIN_VAL,'WFSH-CA12':DIN_VAL)
347   0210          AT5 =  AND('RSH-CA1-1':DIN_VAL,'PDSH-CA1':DIN_VAL)
348   0211          AT6 =  AND(AT3,AT4)
349   0212          AT7 =  AND(AT5,AT6)
350         ;
351   0213          IF !BT27 THEN
352   0214             IF !FLG0 THEN
353                         BEGIN
354   0215                  TIMERCLR OSTMR
355   0216                  TIMERON OSTMR
356   0217                  FLG0 = 1
357                         END
358         ;
359   0218          IF FLG0 THEN
360                   BEGIN
361   0219            IF OSTMR >= 255 THEN
362                     BEGIN
363   0220              TIMEROFF OSTMR
364   0221              TIMERCLR OSTMR
365                     END
366                   END
367   0223          IF BT27 THEN FLG0 = 0
368   0225          IF AND('EC1-ALARM':DEV_FLG,64) THEN PUTMANL 'EC1-ALARM'
369   0227          IF AND('HV-EC1-1':DEV_FLG,64) THEN PUTMANL 'HV-EC1-1'
370   0229          IF AND('HVEC2/3-1':DEV_FLG,64) THEN PUTMANL 'HVEC2/3-1'
371         ;
372         ;       END LOGIC FOR GRAPHICS
373         ;
374   0230          GOTO AGAIN
375         ;
376         ;       E N D   P R O G R A M   L O O P
377         ;
378         ;       BEGIN ANALOG WHITE SUBROUTINE:
379         ;
380         ; ANLG: IF !SELECT() THEN GOTO AGAIN
381   0231  ANLG:   IF (PTYPE = 1) THEN                              ; If some alarm
382                   BEGIN                                   ;be1
383   0232            BITVAL = PAI:AI_ALMCT
384   0233            IF !(AND(BITVAL,16)) THEN                     ; If not inhibited
385                     BEGIN                                 ;be2
```

```
386   0235              IF PSTAT = 3 THEN LET PSTAT = 0
387   0236              IF (PREV) THEN                          ; If rev acting, LOLO, or LO
388                        BEGIN                          ;be3
389   0237                 IF (PSTAT = 0) THEN                  ; If not previously alarmed
390                           BEGIN                       ;be4
391   0239                    IF (PAI:AI_MEAS <= PALARM) THEN GOSUB ALM
392                           END                         ;en4
393                        ELSE
394                           BEGIN                       ;be5
395   0240                    IF (PSTAT = 1) THEN                ; If previously alarmed
396                              BEGIN                    ;be6
397   0242                       IF (FTRIP4) THEN GOSUB ACK1     ; If ACK flag set, then ack it
398   0245                       IF PTYPE !=1 THEN IF AND(PAI:AI_ALMCT,32) THEN GOSUB ACK1
399                              END                      ;en6
400   0247                    IF (PAI:AI_MEAS >= PALARM + PAI:AI_DEAD) THEN GOSUB CLR
401                           END                         ;en5
402                        END                            ;en3
403                     ELSE                                  ; If fail-safe, HIHI, or HI
404                        BEGIN                          ;be10
405   0248                 IF (PSTAT = 0) THEN
406                           BEGIN                       ;be11
407   0250                    IF (PAI:AI_MEAS >= PALARM) THEN GOSUB ALM
408                           END                         ;en11
409                        ELSE
410                           BEGIN                       ;be12
411   0251                    IF (PSTAT = 1) THEN
412                              BEGIN                    ;be13
413   0253                       IF (FTRIP4) THEN GOSUB ACK1
414   0256                       IF PTYPE != 1 THEN IF AND(PAI:AI_ALMCT,32) THEN GOSUB ACK1
415                              END                      ;en13
416   0258                    IF (PAI:AI_MEAS <= PALARM - PAI:AI_DEAD) THEN GOSUB CLR
417                           END                         ;en12
418                        END                            ;en10
419   0259              NXT = PSTAT
420   0261              IF (PPASS) & (PSTAT = 2) THEN NXT = 0
421                     END                               ;en2
422                     ELSE
423                       BEGIN
424   0262                LET PSTAT = 3
425   0263                NXT = 0
426                       END
427   0264           WHITE = OR(WHITE,NXT)
428                  END                                  ;en1
429   0265           ELSE LET PSTAT = 0
430   0266           ILOOP = ILOOP + 1
431   0267           STEP PALARM BY 1
432   0268           STEP PSTAT BY 1
433   0269           STEP PTYPE BY 1
434   0270           STEP PREV BY 1
435   0271           STEP PPASS BY 1
436   0272           RETURN
437          ;
438          ;       END ANALOG WHITE SUBROUTINE
439          ;
440          ;       BEGIN CLEAR, ALARM, ACKNOWLEDGE, MESSAGE, ERROR, AND TERMINATE SUBROUTINES:
```

```
441             ;
442     0273    CLR:    LET PSTAT = 0                                   ; Int set to zer0
443                     ;
444                     ; If not white, then unack:
445                     ;
446     0276            IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
447     0277            FCLEAR = 1                                      ; Clear flag
448     0278            GOSUB ACKK                                      ; Print message
449     0279            RETURN
450             ;
451     0280    ALM:    LET PSTAT = 1                                   ; Int set to one
452                     ;
453                     ; If not white, then unack:
454                     ;
455     0283            IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
456                     ;
457                     ; Turn audible alarm on if white or yellow:
458                     ;
459     0284            CRIT
460     0286            IF AND(PTYPE,3) THEN FALMON = 1
461     0287            PDEV:DEV_TMP0 = 2
462     0288            GOSUB ACKK                                      ; Print message
463     0289            RETURN
464             ;
465     0290    ACK1:   LET PSTAT = 2                                   ; Ack the alarm
466     0291            FACK = 1                                        ; Set ACK flag for C program
467             ;
468             ;       ALARM TYPER STRINGS:
469             ;
470     0293    ACKK:   IF ILOOP = 0 THEN STRNG0 = "E-C-1 PC RADIATION HI/FAIL"
471     0295            IF ILOOP = 23 THEN STRNG0 = "E-C-1 RAW WATER INLET PRESS HI"
472     0297            IF ILOOP = 24 THEN STRNG0 = "E-C-3 USED RAW WATER FLOW LOW"
473     0299            IF ILOOP = 25 THEN STRNG0 = "E-C-1 OUTLET TEMPERATURE HIGH"
474     0301            IF ILOOP = 34 THEN STRNG0 = "START EC1BATCH PGM "
475     0303            IF ILOOP = 35 THEN STRNG0 = "EC1BATCH WRONG UNIT"
476     0305            IF ILOOP = 36 THEN STRNG0 = "TERMINATE EC1BATCH "
477             ;
478             ;  If acknowledging alarm, then:
479             ;
480     0306    AMSG:   IF PTYPE = 1 THEN
481                       BEGIN
482                       ;
483                       ; If white alarm, print:
484                       ;
485     0308              IF (FACK = 0) & (FCLEAR = 0) & (FSTART = 0) THEN PRINT ADI,"WHITE ALARM %p",STRNG0
486     0309              IF (FACK) THEN
487                         BEGIN
488                         ;
489                         ; If white, print it is ack and reset the flag:
490                         ;
491     0310                PRINT ADI,"%p ALARM ACKNOWLEDGE",STRNG0
492     0311                FACK = 0
493                         END
494                       END
495             ;
```

```
496             ;  If CLEAR, print it is clear and reset flag:
497             ;
498     0312            IF (FCLEAR) THEN
499                        BEGIN
500     0313               PRINT ADI,"%p ALARM CLEAR",STRNG0
501     0314               FCLEAR = 0
502                        END
503             ;
504             ;  If STARTUP, print startup message:
505             ;
506     0315            IF (FSTART) THEN
507                        BEGIN
508     0316               FSTART = 0
509     0317               PRINT ADI,"%p",STRNG0
510                        END
511     0318            RETURN
512             ;
513             ;  If ERROR - print message, start timer, and set flag:
514             ;
515     0320    ERRSUB:  IF(ERRNUM() = -187)THEN ERSRET
516                      ELSE
517                       BEGIN
518     0321              IF ERRN != ERRNUM() THEN
519                          BEGIN
520     0322                 TIMERON STIME0
521     0323                 TIMERCLR STIME0
522     0324                 FSN0 = 1
523     0325                 ERRN = ERRNUM()
524     0326                 ERRL = ERRLIN()
525     0327                 PRINT ADI,"ERROR # %i AT LINE %i",errn,errl
526     0328                 ERSCLR
527                          END
528                      END
529     0329             GOTO AGAIN
530             ;
531             ;  If FABORT = 1, then abort program:
532             ;
533             ; TERM:  FABORT = 0
534     0330    TERM:    STRNG0 = "EC1BATCH TERMINATED    "
535     0331             ILOOP = STAR + 2
536     0332             FSTART = 1
537     0333             GOSUB ACKK
538                      END

      Total Errors Detected      0     Total Warnings      0
           Sequence Program Size = 6288 Bytes, 393 Pages
                21 Forward Reference(s)
```

RWBATCH

```
;       R W B A T C H . S E Q                                   VERSION 2
;
;       Function:       Sequence program for Lamp function,
;                       message display on Graphic number 011,
;                       and printer messages.
;
;       Written By      : R. N. Kersey
;       Date            : 05-31-92
;       Facility        : 242-A Evaporator Bldg.
;       Machine         : VAX 4104-A
;       Op. System      : VAX/VMS 5.5-2AH
;       D/3 Version     : 6.3-2 BASELINE
;       Compiler        : SIC
;       Build command   : @[320,20]RWBATCH
;
;       UNIT = 11, SKDKEY = 43
;
;       Assoc. UPF:     [300,1]_PRCESS.___      PRCESS.LST
;       Assoc. RPF:     [300,1]_242A.___        242A.LST
;       SDB:            [300,1]PCM0A.SDB        PCM0ASDB.CMD
;
;       POINTER   SDB VAR.      FUNCTION
;       -------   ---------     ------------------------------------------
;       PSTAT     INTn          Current EPN alarm state is an integer tied to
;                               graphic flashing string of value:
;       .         (n=0-100)             0 = previously not alarmed
;       .                               1 = previously alarmed, not ack
;       .                               2 = previously alarmed, is ack
;       .                               3 = alarm is inhibited
;       .                               4 = input is off scan
;       .                               5 = input is bad
;       .         LEAKn         Special INT variable for multiple white alarms on an EPN
;       PDIN      'EPN'         ---
;       PTYPE     TYPEn         alarm type 0 = none, 1 = white, 2 = yellow,
;                                  3 = yellow and yellow, 4 = red,
;                                  6 = yellow and red
;       PREV      FRVn          Reverse acting (not fail safe) flag (1=set)
;       PSKID     SKIDn         Used for analog alarms
;       PPASS     FPASn         Turn lamp off when alarm is ACKnowledged
;       PALARM    FALMn         Analog alarm flag
;       ---       FNING         Unused error flag
;
; MODIFICATION LOG:
;
;       11-AUG-91       - Fixed order of variables so alarms will       M. Teats
;                         appear on graphic (PDI, TI, and TSL alarms
;                         were not working)
;                       - Added comments
;
;       30-DEC-92       Fixed white alarm print message.        M. Teats
;
;       07-JUN-93       Added comments and baselined for        M. Teats
;                       D/3 version 3.3.  CCN# 93JUN01-01.
;
;       15-DEC-94       Revised header to indicate the new VMS          J. Berger
```

```
                ;                       system. Rebaselined document. Removed
                ;                       obsolite Select() statement
                ;
                ;  END MOD LOG
                ;
                UNIT    RWATER
                        COMMON XYZFRM,PB2
                ;
                        TSL == 'TSL242A81'
                        AUTO1 == 'FQI-RW-1':CB_FLG:1            ; For FQI-RESET AUTO mode check
                        AUTO1B == 'FQI-RW-1B':CB_FLG:1          ; For FQI-RESET AUTO mode check
                        CALOUT1 == 'FQI-RW-1':CB_OTVL:1         ; For FQI-RESET calc block outvalue
                        CALOUT1B == 'FQI-RW-1B':CB_OTVL:1       ; For FQI-RESET calc block outvalue
                ;
0000            NOP
0001            LET STAR = 33                   ; Start message
0002            LET SKDKEY = 43         ; SKID Key
0003            LET UNITT = 11                  ; SEQ Unit number
0004            ASSIGN PDEV = 'RW-ALARM' ; Alarm device
0005            ASSIGN PTYPE = TYPE33
0006            PTYPE = 0                       ; No start message yet
0007            LET ADI = 0                     ; Alarm destination index
0008            LET FSN0 = 0                    ; No alarm yet
0009            TIMEROFF STIME0         ; Alarm timer off
0010            TIMERCLR STIME0         ; Alarm timer clear
0011            STRNG0 = " "                    ; Blank 30 character string
0012            LET ERRN = 0                    ; Set error number to zero
0013            LET ERRL = 0                    ; Set error line to zero
0014            FTRIP4 = 0                      ; FTRIP4 = 1 when audible horn is to be turned on
0015            GETUNIT TRIP                    ; Which sequence unit is program running in
0016            IF TRIP != UNITT THEN           ; Abort if running in wrong seq. unit
                   BEGIN
0017               FSTART = 1
0018               ILOOP = STAR + 1
0019               GOSUB ACKK
0020               FSTART = 1
0021               PRINT ADI," >>>>>  ATTEMPT TO RUN RWBATCH IN WRONG UNIT"
0022               GOTO TERM
                   END
                ;       IF (SELECT()=0) THEN GOTO AGAIN
                ;
                ;       B E G I N   I N I T I A L I Z I N G
                ;
0023    INIT:   LET FABORT = 0                  ; If FABORT = 1 then abort program
0024            LET FSTART = 0                  ; Program start flag = 0
0025            LET SD0 = 0                     ; FQI-RESET manual mode flag = 0
0026            LET SD1 = 0                     ; FQI-RESET manual mode flag = 0
0027            LAMP ADI,SKDKEY,0               ; Turn off the annunciator key
0028            LSG0 = " "                      ; Clear the string variables
0029            LSG1 = " "
0030            LSG2 = " "
0031            LSG3 = " "
0032            LSG4 = " "
0033            PDEV:DEV_TMP0 = 1                               ; Set alarm device to non-alarm state
0034            IF AND(PDEV:DEV_ALMCT,16) THEN                  ; If alarm device is inhibited, remove the inhibit
```

```
111  0035            PDEV:DEV_ALMCT=XOR(PDEV:DEV_ALMCT,16)
112  0036          IF AND(PDEV:DEV_ALMCT,32) THEN                ; If alarm device is ack, clear it
113  0037             PDEV:DEV_ALMCT = XOR(PDEV:DEV_ALMCT,32)
114            ;
115            ;     AIN, DIN, DEV -  ILOOP, TYPE, LIMITS:
116            ;
117            ;     Digitals:
118            ;     --------
119            ;     ILOOP   EPN             TYPE
120            ;       0     LDS-BFP-1       1
121            ;       1     LSL-RWSMP       1
122            ;       2     LDS-BFP-2       1
123            ;       3     RSH-RW-1        4
124            ;       4     RXS-RW-1        2
125            ;
126            ;     Analog Reds or Yellows:
127            ;     ----------------------
128            ;     ILOOP   EPN             TYPE    LIMIT
129            ;       5     PDI-RW-1        0       NA
130            ;       6     PI-RW-1         2       60 Low, 120 Hi
131            ;       7     PDI-RW-2        0       NA
132            ;       8     RI-RW-1         4       NA
133            ;       9     PI-RW-2         0       NA
134            ;
135            ;     Analog Whites:
136            ;     ------------
137            ;     ILOOP   EPN             TYPE    AI      LIMIT
138            ;       10    PDI-RW-1        1       AI0     13.0    Hi
139            ;       11    PDI-RW-2        1       AI1.    13.0    Hi
140            ;       12    PI-RW-2         1       AI2     60.0    Low
141            ;
142            ;     PCM1 DIN:
143            ;     --------
144            ;     ILOOP   EPN             TYPE
145            ;       13    TSL242A81       2
146            ;
147            ;     PCM0 DEV:
148            ;     --------
149            ;     ILOOP   EPN             TYPE
150            ;       14    HV-RW-1         2
151            ;       15    HV-RW-2         2
152            ;
153            ;     Set all PCM0 flags and integers:
154            ;
155  0038            ASSIGN PDIN = 'LDS-BFP-1'                   ; Assign pointer to first DIN
156  0039            ASSIGN PTYPE = TYPE0
157  0040            ASSIGN PREV = FRV0
158  0041            ASSIGN PSTAT = INT0
159  0042            ASSIGN PSKID = SKID0
160  0043            ASSIGN PPASS = FPAS0
161            ;
162  0044            FOR ILOOP = 0 TO 4                          ; For PCM0 DINs
163  0045            LET PPASS = 0
164  0046            LET PREV = 0
165  0048            IF PSTAT > 5 THEN PSTAT = 0                 ; Reset current status if undefined
```

```
166   0049          LET PSKID = 0                                ; Default previous status to no alarms
167   0050          LET PTYPE = 1                                ; Default to white alarm
168   0051          IF AND(PDIN:DIN_DFLG,3) THEN                 ; Check for alarms (Must be red or yellow)
169                    BEGIN
170   0052             PTYPE = 2                                 ; Default to Yellow
171   0054             IF AND(PDIN:DIN_DFLG,1) THEN LET PREV = 1 ; Reverse acting
172   0056             IF AND(PDIN:DIN_DFLG,4) THEN LET PTYPE = 4 ; Critical red alarm
173                    END
174   0057          STEP PDIN BY 1
175   0058          STEP PTYPE BY 1
176   0059          STEP PREV BY 1
177   0060          STEP PSTAT BY 1
178   0061          STEP PSKID BY 1
179   0062          STEP PPASS BY 1
180   0063          NEXT ILOOP
181         ;
182   0064          FOR ILOOP = 5 TO 15                          ; For everything else
183   0065          PTYPE = 0
184   0066          PREV = 0
185   0067          PSTAT = 0
186   0068          PSKID = 0
187   0069          PPASS = 0
188   0070          STEP PTYPE BY 1
189   0071          STEP PREV BY 1
190   0072          STEP PSTAT BY 1
191   0073          STEP PSKID BY 1
192   0074          STEP PPASS BY 1
193   0075          NEXT ILOOP
194         ;
195         ;       Set alarm type if not zero for above analogs:
196         ;
197   0076          LET TYPE6 = 2
198   0077          LET TYPE8 = 4
199   0078          LET TYPE10 = 1
200   0079          LET TYPE11 = 1
201   0080          LET TYPE12 = 1
202   0081          LET TYPE13 = 2
203   0082          LET TYPE14 = 2
204   0083          LET TYPE15 = 2
205         ;
206   0084          LET FRV12 = 1                                ; PI-RW-2 low
207         ;
208         ;       Set White alarm limits:
209         ;
210   0085          LET AI0 = 13.0                               ; PDI-RW-1 hi
211   0086          LET AI1 = 13.0                               ; PDI-RW-2 hi
212   0087          LET AI2 = 60.0                               ; PI-RW-2 low
213         ;
214         ;       I N I T I A L I Z I N G   D O N E
215         ;
216         ;       S E N D   P R O G R A M   S T A R T   M E S S A G E
217         ;
218   0088          ANALOG = 0
219   0089          ILOOP = STAR
220   0090          FSTART = 1
```

```
221   0091          GOSUB ACKK
222   0092          ON ERROR GOSUB ERRSUB
223         ;
224         ;       B E G I N   P R O G R A M   L O O P
225         ;
226   0093  AGAIN:  MARK STEP "WTUNTL",0,0
227         ;       WAITUNTIL SELECT() | FABORT      ; Wait until micro is selected or program aborted
228         ;       IF (FABORT) THEN GOTO TERM       ; Terminate program if abort flag set
229   0094          MARK STEP "RNNING",0,0
230   0095          WAIT 1
231         ;
232   0096          IF (FSNO) THEN                   ; If alarm flag is set
233   0097             IF STIMEO > 60 THEN           ; If alarm flag active for 1 minute
234                         BEGIN                    ; Reset timer and turn off flag
235   0098                  TIMEROFF STIMEO
236   0099                  TIMERCLR STIMEO
237   0100                  LET FSNO = 0
238   0101                  LET ERRN = 0
239   0102                  LET ERRL = 0
240                         END
241         ;
242         ;       If RW-ALARM $T1 is alarmed and it's ALMCT indicates alarms are acknowledged:
243         ;
244   0103          IF (PDEV:DEV_TMPO = 2) & AND(PDEV:DEV_ALMCT,32) THEN
245                   BEGIN
246   0104            PDEV:DEV_TMPO = 1              ; Put RW-ALARM in ack state
247   0105            LET FTRIP4 = 1                 ; Set for unit alarm ack, system to graphic
248   0106            LET ACK = 1                    ; Set for unit alarm ack, graphic to system
249   0107            CRIT
250   0108            LAMP ADI,SKDKEY,1              ; Turn lamp on
251                   END
252         ;
253         ;       BEGIN DIGITALS (PCMO):
254         ;
255   0109          ASSIGN PDIN = 'LDS-BFP-1'
256   0110          ASSIGN PTYPE = TYPEO
257   0111          ASSIGN PREV = FRVO
258   0112          ASSIGN PSTAT = INTO
259   0113          ASSIGN PSKID = SKIDO
260   0114          ASSIGN PPASS = FPASO
261   0115          LET LAST = 0
262   0116          LET NXT = 0
263   0117          LMPVAR = 0
264   0118          BITVAL = 0
265         ;
266   0119  LOP:    FOR ILOOP = 0 TO 4
267         ;       IF !SELECT() THEN GOTO AGAIN
268   0120          IF (PTYPE > 0) THEN                                    ; If some type of alarm
269                   BEGIN                                          ;be1
270   0122            IF AND(PDIN:DIN_ALMCT,16) THEN LET PSTAT = 3   ; If EPN Inhibited
271   0124            IF AND(PDIN:DIN_ALMCT,4) THEN LET PSTAT = 5            ; If EPN is Bad input
272   0126            IF !AND(PDIN:DIN_FLG,8) THEN LET PSTAT = 4             ; If EPN is disabled
273   0127            IF (PSTAT < 3) THEN                                    ; If not inhibited
274                         BEGIN                                    ;be2
275   0128                  IF (PSTAT = 1) THEN
```

```
276                          BEGIN                                    ;be3
277    0129                  IF (PTYPE != 1) THEN                              ; If not white
278    0131                     IF AND(PDIN:DIN_ALMCT,32) THEN GOSUB ACK1
279    0133                  IF (FTRIP4) THEN GOSUB ACK1
280                          END                                      ;en3
281    0136                IF PSTAT = 0 THEN IF (PDIN:DIN_VAL = PREV) THEN GOSUB ALM
282    0137                IF (PSTAT) THEN
283                             BEGIN                                ;be4
284    0139                     IF (PDIN:DIN_VAL != PREV) THEN GOSUB CLR
285    0140                     NXT = PSTAT
286    0142                     IF (PPASS) & (PSTAT = 2) THEN NXT = 0
287    0143                     LAST = OR(LAST,NXT)
288                             END                                  ;en4
289                         END                                      ;en2
290    0145             ELSE IF !AND(PDIN:DIN_ALMCT,16)&!AND(PDIN:DIN_ALMCT,4)&AND(PDIN:DIN_FLG,8)THEN LET PSTAT = 0
291                      END                                        ;en1
292    0146          STEP PDIN BY 1
293    0147          STEP PTYPE BY 1
294    0148          STEP PREV BY 1
295    0149          STEP PSTAT BY 1
296    0150          STEP PSKID BY 1
297    0151          STEP PPASS BY 1
298    0152          NEXT ILOOP
299          ;-
300          ;        END DIGITALS (PCM0)
301          ;
302          ;        BEGIN ANALOG RED AND YELLOWS:
303          ;
304    0153          STRNG0 = "RUN"
305    0154          LET ANALOG = 0
306    0155          ASSIGN PAI = 'PDI-RW-1'
307    0156          ASSIGN PSKID = SKID5
308    0157          ASSIGN PTYPE = TYPE5
309          ;
310    0158          FOR ILOOP = 5 TO 9
311          ;        IF !SELECT() THEN GOTO AGAIN
312    0159          IF (PTYPE) THEN                                  ; If some alarm exists
313                     BEGIN                                 ;be1
314    0160             BITVAL = PAI:AI_ALMCT
315    0161             IF !AND(BITVAL,16) THEN                       ; If not inhibited
316                        BEGIN                              ;be2
317    0162                IF AND(BITVAL,15) THEN                    ; If alarm exists and is current
318                           BEGIN                           ;be3
319    0164                   IF AND(BITVAL,32) THEN NXT = 2          ; If alarm is ack
320                           ELSE                                    ; If alarm is unack
321                              BEGIN                        ;be4
322    0165                      IF PSKID !=BITVAL THEN
323                                 BEGIN                     ;be5
324    0167                         IF BITVAL < 3 THEN FALMON = 1
325    0168                         LET PSKID = BITVAL
326    0169                         PDEV:DEV_TMP0 = 2
327    0170                         NXT = 1
328                                 END                       ;en5
329    0171                      ELSE NXT = 1
330                              END                          ;en4
```

```
331                          END                                ;en3
332                       ELSE                                        ; Else there is no alarm
333                          BEGIN
334    0172                  NXT = 0
335    0173                  LET PSKID = 0
336                          END
337                       END                                ;en2
338                    ELSE                                          ; Else alarm is inhibited
339                       BEGIN
340    0174               NXT = 0
341    0175               LET PSKID = 0
342                       END
343    0176            ANALOG = OR(ANALOG,NXT)
344                    END                                ;en1
345    0177         STEP PSKID BY 1
346    0178         STEP PTYPE BY 1
347    0179         STEP PAI BY 1
348    0180         NEXT ILOOP
349            ;
350            ;     END ANALOG RED AND YELLOW
351            ;
352            ;     BEGIN ANALOG WHITE ALARMS:
353            ;
354    0181         WHITE = 0
355    0182         ASSIGN PTYPE = TYPE10
356    0183         ASSIGN PSTAT = INT10
357    0184         ASSIGN PALARM = AI0
358    0185         ASSIGN PPASS = FPAS10
359    0186         ASSIGN PREV  = FRV10
360    0187         ILOOP = 10
361            ;
362    0188         ASSIGN PAI = 'PDI-RW-1'
363    0189         GOSUB ANLG                                    ; Hi
364    0190         ASSIGN PAI = 'PDI-RW-2'
365    0191         GOSUB ANLG                                    ; Hi
366    0192         ASSIGN PAI = 'PI-RW-2'
367    0193         GOSUB ANLG                                    ; Low
368            ;
369            ;     END ANALOG WHITE ALARMS
370            ;
371            ;     BEGIN PCM1 DIN (TSL242A81):
372            ;
373    0195         IF (ERRNUM() = -187) THEN GOTO SKP1                  ; Skip TSL242A81 if PCM1 micros down
374    0196         IF TYPE13 > 0 THEN                                   ; If some type of alarm
375                    BEGIN                                ;be1
376    0198            IF AND('TSL':DIN_ALMCT,16) THEN INT13 = 3         ; If EPN is Inhibited
377    0200            IF AND('TSL':DIN_ALMCT,4) THEN INT13 = 5          ; If EPN is Bad input
378    0202            IF !AND('TSL':DIN_FLG,8) THEN INT13 = 4           ; If EPN is disabled
379    0203            IF (INT13 < 3) THEN                               ; If not inhibited
380                         BEGIN                           ;be2
381    0204                  IF (INT13 = 1) THEN
382                             BEGIN                       ;be3
383    0205                     IF (TYPE13 != 1) THEN
384    0207                        IF AND('TSL':DIN_ALMCT,32) THEN GOSUB ACK2
385    0209                     IF (FTRIP4) THEN GOSUB ACK2
```

```
386                              END                                        ;en3
387    0210                    IF (INT13 = 0) THEN
388    0212                          IF ('TSL':DIN_VAL = FRV13) THEN GOSUB ALM2
389    0213                    IF (INT13) THEN
390                                BEGIN                                    ;be4
391    0215                        IF ('TSL':DIN_VAL != FRV13) THEN GOSUB CLR2
392    0216                        NXT = INT13
393    0218                        IF (FPAS13) & (INT13 = 2) THEN NXT =.0
394    0219                        LAST = OR(LAST,NXT)
395                                END                                      ;en4
396                            END                                          ;en2
397    0221                ELSE IF !AND('TSL':DIN_ALMCT,16)&!AND('TSL':DIN_ALMCT,4)&AND('TSL':DIN_FLG,8)THEN INT13 = 0
398                        END                                              ;en1
399            ;
400            ;       END PCM1 DIN
401            ;
402            ;       BEGIN HANDVALVE DEVICE ALARMS:
403            ;
404    0222    HV:     DEV0 = 0
405    0223            ASSIGN PDIN = 'HV-RW-1'
406    0224            ASSIGN PTYPE = TYPE14
407    0225            ASSIGN PSTAT = INT14
408            ;
409    0226            FOR ILOOP = 14 TO 15
410    0227            IF PTYPE > 0 THEN                                    ; If some type of alarm
411                       BEGIN                                             ;be1
412    0229               IF AND(PDIN:DEV_ALMCT,16) THEN LET PSTAT = 3      ; If EPN is Inhibited
413    0231               IF AND(PDIN:DEV_ALMCT,4) THEN LET PSTAT = 5       ; If EPN is Bad input
414    0233               IF !AND(PDIN:DEV_FLG,8) THEN LET PSTAT = 4        ; If EPN is disabled
415    0234               IF (PSTAT < 3) THEN                               ; If not inhibited
416                            BEGIN                                        ;be2
417    0235                    IF (PSTAT = 1) THEN
418                               BEGIN                                     ;be3
419    0236                       IF (PTYPE != 1) THEN
420    0238                          IF AND(PDIN:DEV_ALMCT,32) THEN GOSUB ACK1
421    0240                       IF (FTRIP4) THEN GOSUB ACK1
422                               END                                       ;en3
423    0242                    IF AND(PDIN:DEV_CMD,4) THEN GOSUB ALM1
424    0243                    IF (PSTAT) THEN
425                                BEGIN                                    ;be4
426    0245                        IF !AND(PDIN:DEV_CMD,4) THEN GOSUB CLR
427    0246                        NXT = PSTAT
428    0248                        IF (PPASS) & (PSTAT = 2) THEN NXT = 0
429    0249                        LAST = OR(LAST,NXT)
430                                END                                      ;en4
431                            END                                          ;en2
432    0251                ELSE IF !AND(PDIN:DEV_ALMCT,16)&!AND(PDIN:DEV_ALMCT,4)&AND(PDIN:DEV_FLG,8)THEN LET PSTAT = 0
433                        END                                              ;en1
434    0252            STEP PDIN BY 1
435    0253            STEP PTYPE BY 1
436    0254            STEP PREV BY 1
437    0255            STEP PSTAT BY 1
438    0256            STEP PSKID BY 1
439    0257            STEP PPASS BY 1
440    0258            NEXT ILOOP
```

```
441           ;
442           ;       END HANDVALVE DEVICE ALARMS
443           ;
444     0259  SKP1:   MARK STEP "BELOW",0,0
445     0260          SORT = 0
446     0261          FTRIP4 = 0
447     0263          IF LAST > 3 THEN PRINT ADI," LAMP VARIABLE IS GREATER THAN 3!"
448     0264          CURENT = OR(WHITE,ANALOG)
449     0265          CURENT = OR(LAST,CURENT)
450     0266          CURENT = OR(DEVO,CURENT)
451     0268          IF AND(CURENT,1) THEN LMPVAR = 2
452     0270             ELSE IF AND(CURENT,2) THEN LMPVAR = 1
453     0271                     ELSE LMPVAR = 0
454     0272          LAMP ADI,SKDKEY,LMPVAR
455     0273          MARK STEP "BOTTOM",0,0
456     0274          NOCRIT
457           ;
458           ;       BEGIN FQI-RESET:
459           ;
460           ;       Function:       For totalizers that have a CALC block which performs the
461           ;                       integration, set the output of the CALC block to zero
462           ;                       after the FQI loop has been placed in AUTO from MANUAL.
463           ;                       Reset both FQI-RW-1 and -1B if either one is reset.
464           ;
465     0276          IF !AND(AUTO1,64) THEN SD0 = 1                  ; If manual, set manual flag
466     0277          IF AND(AUTO1,64) & AND(SD0,1) THEN              ; For manual-AUTO transition
467                      BEGIN
468     0278             SD0 = 0
469     0279             CALOUT1 = 0.0
470     0280             CALOUT1B = 0.0
471                      END
472           ;
473     0282          IF !AND(AUTO1B,64) THEN SD1 = 1                 ; If manual, set manual flag
474     0283          IF AND(AUTO1B,64) & AND(SD1,1) THEN             ; For manual-AUTO transition
475                      BEGIN
476     0284             SD1 = 0
477     0285             CALOUT1B = 0.0
478     0286             CALOUT1 = 0.0
479                      END
480           ;
481           ;       END FQI-RESET
482           ;
483     0288          IF AND('RW-ALARM':DEV_FLG,64) THEN PUTMANL 'RW-ALARM'
484     0289          GOTO AGAIN
485           ;
486           ;       E N D   P R O G R A M   L O O P
487           ;
488           ;       BEGIN ANALOG WHITE SUBROUTINE:
489           ;
490           ; ANLG:  IF !SELECT() THEN GOTO AGAIN
491     0290  ANLG:   IF (PTYPE = 1) THEN                             ; If some alarm
492                      BEGIN                                 ;be1
493     0291             BITVAL = PAI:AI_ALMCT
494     0292             IF !(AND(BITVAL,16)) THEN                  ; If not Inhibited
495                        BEGIN                               ;be2
```

```
496   0294              IF PSTAT = 3 THEN LET PSTAT = 0
497   0295              IF (PREV) THEN                              ; If rev acting, LOLO, or LO
498                        BEGIN                                 ;be3
499   0296                 IF (PSTAT = 0) THEN                     ; If not previously alarmed
500                           BEGIN                              ;be4
501   0298                    IF (PAI:AI_MEAS <= PALARM) THEN GOSUB ALM
502                           END                                ;en4
503                        ELSE
504                           BEGIN                              ;be5
505   0299                    IF (PSTAT = 1) THEN                ; If previously alarmed
506                              BEGIN                           ;be6
507   0301                       IF (FTRIP4) THEN GOSUB ACK1     ; If ACK flag set, then ack it
508   0304                       IF PTYPE !=1 THEN IF AND(PAI:AI_ALMCT,32) THEN GOSUB ACK1
509                              END                             ;en6
510   0306                    IF (PAI:AI_MEAS >= PALARM + PAI:AI_DEAD) THEN GOSUB CLR
511                           END                                ;en5
512                        END                                   ;en3
513                     ELSE                                     ; If fail-safe, HIHI, or HI
514                        BEGIN                                 ;be10
515   0307                 IF (PSTAT = 0) THEN
516                           BEGIN                              ;be11
517   0309                    IF (PAI:AI_MEAS >= PALARM) THEN GOSUB ALM
518                           END                                ;en11
519                        ELSE
520                           BEGIN                              ;be12
521   0310                    IF (PSTAT = 1) THEN
522                              BEGIN                           ;be13
523   0312                       IF (FTRIP4) THEN GOSUB ACK1
524   0315                       IF PTYPE != 1 THEN IF AND(PAI:AI_ALMCT,32)THEN GOSUB ACK1
525                              END                             ;en13
526   0317                    IF (PAI:AI_MEAS <= PALARM - PAI:AI_DEAD) THEN GOSUB CLR
527                           END                                ;en12
528                        END                                   ;en10
529   0318              NXT = PSTAT
530   0320              IF (PPASS) & (PSTAT = 2) THEN NXT = 0
531                     END                                      ;en2
532                     ELSE
533                       BEGIN
534   0321                LET PSTAT = 3
535   0322                NXT = 0
536                       END
537   0323            WHITE = OR(WHITE,NXT)
538                   END                                        ;en1
539   0324            ELSE LET PSTAT = 0
540   0325            STEP PALARM BY 1
541   0326            STEP PSTAT BY 1
542   0327            STEP PTYPE BY 1
543   0328            STEP PREV BY 1
544   0329            STEP PPASS BY 1
545   0330            ILOOP = ILOOP + 1
546   0331            RETURN
547           ;
548           ;       END ANALOG WHITE SUBROUTINE
549           ;
550           ;       BEGIN CLEAR, ALARM, ACKNOWLEDGE, MESSAGE, ERROR, AND TERMINATE SUBROUTINES:
```

```
551             ;
552     0332    CLR:    LET PSTAT = 0                                           ; PCM0 DIN, white AI, DEV
553                     ;
554                     ; If not white, then unack:
555                     ;
556     0335            IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
557     0336            LET FCLEAR = 1
558     0337            GOSUB ACKK
559     0338            RETURN
560             ;
561     0339    CLR2:   LET INT13 = 0                                           ; PCM1 DIN
562                     ;
563                     ; If not white, then unack:
564                     ;
565     0342            IF TYPE13 != 1 THEN IF AND('TSL':DIN_ALMCT,32)THEN 'TSL':DIN_ALMCT = XOR('TSL':DIN_ALMCT,32)
566     0343            LET FCLEAR = 1
567     0344            GOSUB ACKK
568     0345            RETURN
569             ;
570             ; If not white, then unack:
571             ;
572     0348    ALM:    IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
573             ;
574     0349    ALM1:   CRIT
575     0350            LET PSTAT = 1
576     0352            IF AND(PTYPE,3) THEN FALMON = 1
577     0353            PDEV:DEV_TMP0 = 2
578     0354            GOSUB ACKK
579     0355            RETURN
580             ;
581     0356    ALM2:   LET INT13 = 1                                           ; PCM1 DIN
582                     ;
583                     ; If not white, then unack:
584                     ;
585     0359            IF TYPE13 != 1 THEN IF AND('TSL':DIN_ALMCT,32)THEN 'TSL':DIN_ALMCT = XOR('TSL':DIN_ALMCT,32)
586     0360            'RW-ALARM':DEV_TMP0 = 2
587     0361            FALMON = 1
588     0362            GOSUB ACKK
589     0363            RETURN
590             ;
591     0364    ACK1:   LET PSTAT = 2
592     0365            LET FACK = 1
593     0366            GOSUB ACKK
594     0367            RETURN
595             ;
596     0368    ACK2:   LET INT13 = 2                                           ; PCM1 DIN
597     0369            LET FACK = 1
598             ;
599             ;       For multiple AI white alarms on one EPN (use with string packet 78):
600             ;
601     0371    ACKK:       IF (INT12 & INT12 < 3) THEN LET LEAK0 = INT12 + 2           ; PI-RW-2 Lo
602     0373                  ELSE IF INT12 = 3 THEN LET LEAK0 = 9                      ; PI-RW-2 INHA
603     0374                    ELSE LET LEAK0 = 0                                      ; No alarm
604             ;
605             ;       End multiple AI white alarms on one EPN
```

```
606             ;
607             ;       ALARM TYPER STRINGS:
608             ;
609     0376    MSSG:    IF ILOOP = 0 THEN STRNGO = "RW BLDG BFP-1 LEAK    "       ; LDS-BFP-1
610     0378            IF ILOOP = 1 THEN STRNGO = "RW BLDG SUMP LEVEL LO "       ; LSL-RWSMP
611     0380             IF ILOOP = 2 THEN STRNGO = "RW BLDG BFP-2 LEAK    "       ; LDS-BFP-2
612     0382             IF ILOOP = 3 THEN STRNGO = "RW SUPPLY RAD HI      "       ; RSH-RW-1
613     0384             IF ILOOP = 4 THEN STRNGO = "RW SUPPLY RAD SYS FAIL"       ; RXS-RW-1
614     0386             IF ILOOP = 10 THEN STRNGO = "RW BLDG F-RW-1 DP HI  "      ; PDI-RW-1
615     0388             IF ILOOP = 11 THEN STRNGO = "RW BLDG F-RW-2 DP HI  "      ; PDI-RW-2
616     0390             IF ILOOP = 12 THEN STRNGO = "F-RW-1/2 IN PRESS LO  "      ; PI-RW-2
617     0392             IF ILOOP = 13 THEN STRNGO = "RW BLDG TEMP LO       "      ; TSL242A81
618     0394             IF ILOOP = 14 THEN STRNGO = "HV-RW-1 OPEN 30 + SEC."
619     0396             IF ILOOP = 15 THEN STRNGO = "HV-RW-2 OPEN 30 + SEC."
620     0398             IF ILOOP = 33 THEN STRNGO = "RWBATCH PGM STARTING "
621     0400             IF ILOOP = 34 THEN STRNGO = "RWBATCH WRONG UNIT   "
622     0402             IF ILOOP = 35 THEN STRNGO = "RWBATCH TERMINATING  "
623             ;
624     0403    AMSG:    IF PTYPE = 1 THEN
625                         BEGIN
626                         ;
627                         ; If white alarm, print:
628                         ;
629     0405                IF (FACK = 0) & (FCLEAR = 0) & (FSTART = 0) THEN PRINT ADI,"WHITE ALARM %p",STRNGO
630     0406                IF (FACK) THEN
631                            BEGIN
632                            ;
633                            ; If white, print it is ack and reset flag:
634                            ;
635     0407                   PRINT ADI,"%p ALARM ACKNOWLEDGE",STRNGO
636     0408                   LET FACK = 0
637                            END
638                         END
639             ;
640             ;  If CLEAR, print it is clear and reset flag:
641             ;
642     0409            IF (FCLEAR) THEN
643                        BEGIN
644     0410               PRINT ADI,"%p ALARM CLEAR",STRNGO
645     0411               LET FCLEAR = 0
646                        END
647             ;
648             ;  If STARTUP, print startup message:
649             ;
650     0412            IF (FSTART) THEN
651                        BEGIN
652     0413               LET FSTART = 0
653     0414               PRINT ADI,"%p",STRNGO
654                        END
655     0415            RETURN
656             ;
657             ;  If error - print message, start timer, and set flag:
658             ;
659     0416    ERRSUB:    IF ERRN != ERRNUM() THEN
660                           BEGIN
```

```
661   0417              TIMERON STIME0
662   0418              TIMERCLR STIME0
663   0419              LET FSN0 = 1
664   0420              LET ERRN = ERRNUM()
665   0421              LET ERRL = ERRLIN()
666   0422              PRINT ADI,"ERROR # %i AT LINE %i",errn,errl
667                     END
668   0423              ERSCLR
669   0424            GOTO AGAIN
670             ;
671             ;  If FABORT = 1, then abort program:
672             ;
673             ; TERM:  LET FABORT = 0
674   0425      TERM:   STRNG0 = "RWBATCH TERMINATED    "
675   0426              LET ILOOP = STAR + 2
676   0427              LET FSTART = 1
677   0428              SSNG0 = "TERM"
678   0429              GOSUB ACKK
679                     END

      Total Errors Detected      0     Total Warnings      0
           Sequence Program Size = 8224 Bytes, 514 Pages
                33 Forward Reference(s)
```

URWBATCH

```
;       U R W B A T C H . S E Q                                  VERSION 2
;
;       Function:       Sequence program for Lamp function,
;                       message display on Graphic number 017,
;                       and printer messages.
;
;       Written By      : R. N. Kersey
;       Date            : 05-31-92
;       Facility        : 242-A Evaporator Bldg.
;       Machine         : VAX 4104-A
;       Op. System      : VAX/VMS 5.5-2AH
;       D/3 Version     : 6.3-2 BASELINE
;       Compiler        : SIC
;       Build command   : @[320,20]URWBATCH
;
;       UNIT = 12, SKDKEY = 42
;
;       Assoc. UPF:     [300,1]_PRCESS.___      PRCESS.LST
;       Assoc. RPF:     [300,1]_242A.___        242A.LST
;       SDB:            [300,1]PCM0A.SDB        PCM0ASDB.CMD
;
;       POINTER   SDB VAR.      FUNCTION
;       -------   ---------     ----------------------------------------
;       PSTAT     INTn          Current EPN alarm state is an integer tied to
;                               graphic flashing string of value:
;       .         (n=0-100)             0 = previously not alarmed
;       .                               1 = previously alarmed, not ack
;       .                               2 = previously alarmed, is ack
;       .                               3 = alarm is inhibited
;       .                               4 = input is off scan
;       .                               5 = input is bad
;       .         LEAKn         Special INT variable for multiple white alarms on an EPN
;       PDIN      'EPN'         ---
;       PTYPE     TYPEn         alarm type 0 = none, 1 = white, 2 = yellow,
;                                  3 = yellow and yellow, 4 = red,
;                                  6 = yellow and red
;       PREV      FRVn          Reverse acting (not fail safe) flag (1=set)
;       PSKID     SKIDn         Used for analog alarms
;       PPASS     FPASn         Turn lamp off when alarm is ACKnowledged
;       PALARM    FALMn         Analog alarm flag
;       ---       FNING         Unused error flag
;
; MODIFICATION LOG:
;
;        02-JUN-92      Deleted O/C display bar feature.        M. Teats
;
;        08-JUL-92      Removed synchronized PID feature.               M. Teats
;
;        30-DEC-92      Fixed white alarm print message.        M. Teats
;
;        08-JUN-93      Added comments.  CCN# 93JUN01-01.               M. Teats
;
;-mct1+   29-JUN-93      Remove IBAD on RC2-PIG and baselined for M. Teats
;                        D/3 version 3.3.  CCN# 93JUN29-01.
;
```

```
56          ;       15-DEC-94       Revised header to indicate the new VMS          J. Berger
57          ;                       system. Rebaselined document. Removed
58          ;                       obsolite Select() statements.
59          ;
60          ;  END MOD LOG
61          ;
62          UNIT    URW
63          ;
64                  COMMON PB2
65                  DEVON == 'RC2-SAMP':DEV_STAT            ; For FQIC-RESET device on/off check
66                  CALOUT == 'FQIC-RC2':CB_OTVL:3          ; For FQIC-RESET calc block outvalue
67                  RC2_PIG == 'RC2-PIG':DEV_TMP0
68          ;
69  0000            NOP
70  0001            STAR = 24                       ; Start message
71  0002            LET SKDKEY = 42         ; SKID Key
72  0003            LET UNITT = 12                  ; SEQ Unit number
73  0004            ASSIGN PDEV = 'URW-ALARM'       ; Alarm device
74  0005            ASSIGN PTYPE = TYPE24
75  0006            ADI = 0                         ; Alarm destination index
76  0007            FSN0 = 0                        ; No alarm yet
77  0008            TIMEROFF STIME0         ; Alarm timer off
78  0009            TIMERCLR STIME0         ; Alarm timer clear
79  0010            STRNG0 = " "                    ; Blank 30 character string
80  0011            ERRN = 0                        ; Set error number to zero
81  0012            ERRL = 0                        ; Set error line to zero
82  0013            FTRIP4 = 0                      ; FTRIP4 = 1 when audible horn is to be turned on
83  0014            GETUNIT TRIP                    ; Which sequence unit is program running in
84  0015            IF TRIP != UNITT THEN           ; Abort if running in wrong seq. unit
85                     BEGIN
86  0016               FSTART = 1
87  0017               ILOOP = STAR + 1
88  0018               PTYPE = 0
89  0019               GOSUB ACKK
90  0020               GOTO TERM
91                     END
92          ;       IF !SELECT() THEN GOTO AGAIN
93          ;
94          ;       B E G I N   I N I T I A L I Z I N G
95          ;
96  0021            STRNG0 = "INIT"
97  0022    INIT:   FABORT = 0                      ; If FABORT = 1, then abort program
98  0023            FSTART = 0                      ; Program start flag = 0
99  0024            SD0 = 0                         ; For FQIC-RESET, RC2-SAMP device status flag
100 0025            LAMP ADI,SKDKEY,0               ; Turn off the annunciator key
101 0026            LSG0 = " "                      ; Clear the string variables
102 0027            LSG1 = " "
103 0028            LSG2 = " "
104 0029            LSG3 = " "
105 0030            LSG4 = " "
106 0031            PDEV:DEV_TMP0 = 1                               ; Set alarm device to non-alarm state
107 0032            IF AND(PDEV:DEV_ALMCT,16) THEN                  ; If alarm device is inhibited, remove the inhibit
108 0033              PDEV:DEV_ALMCT=XOR(PDEV:DEV_ALMCT,16)
109 0034            IF AND(PDEV:DEV_ALMCT,32) THEN                  ; If alarm device is ack, clear it
110 0035               PDEV:DEV_ALMCT = XOR(PDEV:DEV_ALMCT,32)
```

```
111             ;
112             ;       First, set all flags and integers:
113             ;
114     0036            ASSIGN PDIN = 'FSL-RC2-1'                       ; Assign pointer to first DIN
115     0037            ASSIGN PSTAT = INTO
116     0038            ASSIGN PTYPE = TYPE0
117     0039            ASSIGN PREV = FRV0
118     0040            ASSIGN PSKID = SKID0
119     0041            ASSIGN PPASS = FPAS0
120             ;
121     0042            FOR ILOOP = 0 TO 15
122     0043            PPASS = 0
123     0044            PREV = 0
124     0046            IF PSTAT > 5 THEN PSTAT = 0                     ; Reset current status if undefined
125     0047            PSKID = 0                                       ; Default previous status to no alarms
126     0048            PTYPE = 0                                       ; Default to no alarms
127     0049            IF ILOOP < 4 THEN                               ; For DINs
128                        BEGIN
129     0050               IF AND(PDIN:DIN_DFLG,3) THEN                 ; Check for alarms if defined (must be red or yellow)
130                           BEGIN
131     0051                  PTYPE = 2                                 ; Default to yellow alarm
132     0053                  IF AND(PDIN:DIN_DFLG,1) THEN LET PREV = 1 ; Reverse acting
133     0055                  IF AND(PDIN:DIN_DFLG,4) THEN LET PTYPE = 4 ; Set to red critical alarm
134                           END
135                        END
136     0057            IF ILOOP > 3 THEN PTYPE = 0
137     0058            STEP PPASS BY 1
138     0059            STEP PREV BY 1
139     0060            STEP PSTAT BY 1
140     0061            STEP PSKID BY 1
141     0062            STEP PTYPE BY 1
142     0063            NEXT ILOOP
143             ;
144     0064            TYPE0 = 0
145     0065            TYPE1 = 4
146     0066            TYPE2 = 2
147     0067            TYPE3 = 2
148     0068            TYPE4 = 1
149     0069            TYPE10 = 4
150     0070            TYPE11 = 2
151     0071            TYPE12 = 2
152     0072            TYPE15 = 1
153     0073            TYPE16 = 0
154     0074            TYPE17 = 2
155             ;
156     0075            FRV0 = 0
157     0076            FRV1 = 0
158     0077            FRV2 = 0
159     0078            FRV3 = 1
160     0079            FRV15 = 1
161     0080            FRV16 = 0
162     0081            FRV17 = 1
163             ;
164     0082            AI0 = 1200.0
165             ;
```

```
166          ;       E N D   I N I T I A L I Z I N G
167          ;
168          ;       S E N D   P R O G R A M   S T A R T   M E S S A G E
169          ;
170    0083          ANALOG = 0
171    0084          ILOOP = STAR
172    0085          PTYPE = 0
173    0086          FSTART = 1
174    0087          GOSUB ACKK
175    0088          ON ERROR GOSUB ERRSUB
176          ;
177          ;       B E G I N   P R O G R A M   L O O P
178          ;
179    0089  AGAIN:  MARK STEP "WTUNTL",0,0
180          ;       WAITUNTIL SELECT() ¦ FABORT      ; Wait until micro is selected or program aborted
181          ;       IF ( FABORT) THEN GOTO TERM      ; Terminate program if abort flag set
182    0090          MARK STEP "RNNING",0,0
183    0091          WAIT 1
184    0092          IF (FSNO) THEN                   ; If alarm flag is set
185    0093             IF STIME0 > 60 THEN           ; If alarm flag active for 1 minute
186                         BEGIN                     ; Reset timer and turn off flag
187    0094                 TIMEROFF STIME0
188    0095                 TIMERCLR STIME0
189    0096                 LET FSNO = 0
190    0097                 LET ERRN = 0
191    0098                 LET ERRL = 0
192                         END
193          ;
194          ;       If URW-ALARM $T1 is alarmed and it's ALMCT indicates alarms are acknowledged:
195          ;
196    0099          IF (PDEV:DEV_TMP0 = 2) & AND(PDEV:DEV_ALMCT,32) THEN
197                      BEGIN
198    0100              PDEV:DEV_TMP0 = 1            ; Put URW-ALARM in ack state
199    0101              LET FTRIP4 = 1               ; Set for unit alarm ack, system to graphic
200    0102              LET ACK = 1                  ; Set for unit alarm ack, graphic to system
201    0103              CRIT
202    0104              LAMP ADI,SKDKEY,1            ; Turn lamp on
203                      END
204          ;
205          ;       BEGIN DINs:
206          ;
207    0105          ASSIGN PDIN = 'FSL-RC2-1'
208    0106          ASSIGN PTYPE = TYPE0
209    0107          ASSIGN PREV = FRV0
210    0108          ASSIGN PSTAT = INT0
211    0109          ASSIGN PPASS = FPAS0
212    0110          LET CURENT = 0                   ; Used to set lamp status
213    0111          LET LAST = 0
214    0112          LET ANALOG = 0
215    0113          LET NXT = 0
216    0114          LMPVAR = 0
217    0115          BITVAL = 0
218          ;
219    0116  LOP:    FOR ILOOP = 0 TO 3
220          ;       IF !SELECT() THEN GOTO AGAIN
```

```
221   0117          IF ( PTYPE > 0 ) THEN                                    ; If some type of alarm
222                    BEGIN                                                ;be1
223   0119             IF AND(PDIN:DIN_ALMCT,16) THEN LET PSTAT = 3          ; If EPN is inhibited
224   0121             IF AND(PDIN:DIN_ALMCT,4) THEN LET PSTAT = 5           ; If EPN is bad input
225   0123             IF !AND(PDIN:DIN_FLG,8) THEN LET PSTAT = 4            ; If EPN is disabled
226   0124             IF ( PSTAT < 3 ) THEN                                ; If not inhibited
227                         BEGIN                                           ;be2
228   0125                  IF (PSTAT = 1) THEN
229                            BEGIN                                        ;be3
230   0126                     IF (PTYPE != 1) THEN
231   0128                        IF AND(PDIN:DIN_ALMCT,32)THEN GOSUB ACK1
232   0130                     IF (FTRIP4) THEN GOSUB ACK1
233                            END                                          ;en3
234   0133                  IF PSTAT = 0 THEN IF(PDIN:DIN_VAL = PREV) THEN GOSUB ALM
235   0134                  IF (PSTAT) THEN
236                               BEGIN                                     ;be4
237   0136                        IF PDIN:DIN_VAL != PREV THEN GOSUB CLR
238   0137                        NXT = PSTAT
239   0139                        IF (PPASS) & (PSTAT = 2 ) THEN NXT = 0
240   0140                        LAST = OR(LAST,NXT)
241                               END                                       ;en4
242                         END                                             ;en2
243   0142               ELSE IF !AND(PDIN:DIN_ALMCT,16)&!AND(PDIN:DIN_ALMCT,4)&AND(PDIN:DIN_FLG,8)THEN LET PSTAT = 0
244                      END                                               ;en1
245   0143          STEP PDIN BY 1
246   0144          STEP PTYPE BY 1
247   0145          STEP PREV BY 1
248   0146          STEP PSTAT BY 1
249   0147          STEP PPASS BY 1
250   0148          NEXT ILOOP
251           ;
252           ;     END DINs
253           ;
254           ;     BEGIN ANALOG RED AND YELLOWS:
255           ;
256   0149          ASSIGN PSKID = SKID4
257   0150          ASSIGN PAI = 'FIC-EC1-1'
258   0151          FOR JLOOP = 4 TO 14
259           ;     IF !SELECT() THEN GOTO AGAIN
260   0152          IF (PTYPE) THEN                                          ; If some alarm exists
261                    BEGIN                                          ;be1
262   0153             BITVAL = PAI:AI_ALMCT
263   0154             IF !AND(BITVAL,16) THEN                               ; If not inhibited
264                         BEGIN                                     ;be2
265   0155                  IF AND(BITVAL,15) THEN                           ; If alarm exists and is current
266                               BEGIN                               ;be3
267   0157                        IF AND(BITVAL,32) THEN NXT = 2             ; If alarm is ack
268                               ELSE                                       ; If alarm is unack
269                                  BEGIN                            ;be4
270   0158                           IF PSKID !=BITVAL THEN
271                                        BEGIN                      ;be5
272   0160                                 IF BITVAL < 3 THEN FALMON = 1
273   0161                                 LET PSKID = BITVAL
274   0162                                 PDEV:DEV_TMP0 = 2
275   0163                                 NXT = 1
```

```
276                                   END                                ;en5
277    0164                        ELSE NXT = 1
278                                END                                   ;en4
279                             END                                      ;en3
280                          ELSE                                              ; Else there is no alarm
281                             BEGIN
282    0165                     NXT = 0
283    0166                     LET PSKID = 0
284                             END
285                          END                                         ;en2
286                       ELSE                                              ; Else alarm is inhibited
287                          BEGIN
288    0167                  NXT = 0
289    0168                  LET PSKID = 0
290                          END
291    0169                  ANALOG = OR(ANALOG,NXT)
292                          END                                         ;en1
293    0170               STEP PREV BY 1
294    0171               STEP PSTAT BY 1
295    0172               STEP PSKID BY 1
296    0173               STEP PTYPE BY 1
297    0174               STEP PAI BY 1
298    0175               NEXT JLOOP
299            ;
300            ;          END ANALOG RED AND YELLOWS
301            ;
302            ;          BEGIN ANALOG WHITE ALARMS:
303            ;
304    0176               WHITE = 0
305    0177               ASSIGN PALARM = AI0
306    0178               ASSIGN PREV = FRV15
307    0179               ASSIGN PSTAT = INT15
308    0180               ASSIGN PTYPE = TYPE15
309            ;
310    0181               ASSIGN PAI = 'FIC-EC1-1'
311    0182               ILOOP = 15
312    0183               GOSUB ANLG
313                       ;
314                       ; Use LEAK0 with string packet 78 in URW graphic for FIC-EC1-1 LO:
315                       ;
316    0185               IF INT15 = 3 ¦ INT16 = 3 THEN LET LEAK0 = 9
317    0187                  ELSE IF (INT16) THEN LET LEAK0 = INT16 + 4
318    0189                     ELSE IF(INT15) THEN LET LEAK0 = INT15 + 2
319    0190                        ELSE LET LEAK0 = 0
320            ;
321            ;          END ANALOG WHITE ALARMS
322            ;
323            ;          BEGIN RC2-PIG ALARM:
324            ;
325    0191    RC2:       ASSIGN PTYPE = TYPE17
326    0192               ASSIGN PREV = FRV17
327    0193               ASSIGN PSTAT = INT17
328    0194               ASSIGN PPASS = FPAS17
329            ;          IF !SELECT() THEN GOTO AGAIN
330    0195               IF ( PTYPE > 0 ) THEN                                    ; If some type of alarm
```

```
331                     BEGIN                                                   ;be1
332     0197            IF AND('RC2-PIG':DEV_ALMCT,16) THEN LET PSTAT = 3               ; If EPN is inhibited
333             ;-mct1+ IF AND('RC2-PIG':DEV_ALMCT,4) THEN BLET PSTAT = 5               ; No IBAD on devices
334     0199            IF !AND('RC2-PIG':DEV_FLG,8) THEN LET PSTAT = 4                 ; If EPN is disabled
335     0200            IF ( PSTAT < 3 ) THEN                                   ; If not inhibited
336                         BEGIN                                               ;be2
337     0201                IF (PSTAT = 1) THEN
338                            BEGIN                                            ;be3
339     0202                   IF (PTYPE != 1) THEN
340     0204                      IF AND('RC2-PIG':DEV_ALMCT,32) THEN GOSUB ACK1
341     0206                   IF (FTRIP4) THEN GOSUB ACK1
342                            END                                              ;en3
343     0208                IF PSTAT = 0 THEN IF ('RC2-PIG':DEV_ALMCT = 7) THEN
344                            BEGIN                                            ;be4
345     0209                   CRIT
346     0210                   LET PSTAT = 1
347     0211                   'URW-ALARM':DEV_TMPO = 2
348     0212                   FALMON = 1
349     0213                   LAMP ADI,SKDKEY,2
350     0214                   GOSUB ACKK
351     0215                   NOCRIT
352                            END                                              ;en4
353     0216                IF (PSTAT) THEN
354                               BEGIN                                         ;be5
355     0217                      IF !AND('RC2-PIG':DEV_ALMCT,7) THEN
356                                  BEGIN                                      ;be6
357     0218                         LET PSTAT = 0
358     0219                         LET FCLEAR = 1
359     0220                         GOSUB ACKK
360                                  END                                        ;en6
361     0221                      NXT = PSTAT
362     0223                      IF (PPASS) & (PSTAT = 2) THEN NXT = 0
363     0224                      LAST = OR(LAST,NXT)
364                               END                                           ;en5
365                         END                                                 ;en2
366     0226            ELSE IF !AND('RC2-PIG':DEV_ALMCT,16)&AND('RC2-PIG':DEV_FLG,8)THEN LET PSTAT = 0
367     0228            IF PSTAT = 5 THEN PSTAT = 0
368                     END                                                     ;en1
369             ;
370             ;       END RC2-PIG ALARM
371             ;
372     0229            MARK STEP "BELOW",0,0
373     0230            FTRIP4 = 0
374     0232            IF LAST > 3 THEN PRINT ADI," LAMP VARIABLE IS GREATER THAN 3!"
375     0233            CURENT = OR(LAST,WHITE)
376     0234            CURENT = OR(CURENT,ANALOG)
377     0236            IF AND(CURENT,1) THEN LMPVAR = 2
378     0238               ELSE IF AND(CURENT,2) THEN LMPVAR = 1
379     0239                       ELSE LMPVAR = 0
380     0240            LAMP ADI,SKDKEY,LMPVAR
381     0241            MARK STEP "BOTTOM",0,0
382     0242            NOCRIT
383             ;
384             ;       BEGIN LOGIC FOR GRAPHICS:
385             ;
```

```
386   0243          AT0 = AND(RC2_PIG,1)
387   0244          AT1 = AND(RC2_PIG,2)
388   0245          AT2 = AND(RC2_PIG,4)
389   0246          AT3 = AND(RC2_PIG,8)
390   0247          AT4 = AND(RC2_PIG,16)
391   0248          AT5 = AND('RC2-SAMP':DEV_TMP0,32)
392   0249          AT6 = AND(RC2_PIG,64)
393   0250          AT7 = and('RC2-SAMP':DEV_TMP0,128)
394   0251          AT8 = and(rc2_pig,256)
395   0252          SA3 = AND('RC2-PIG':DEV_STAT,8)          ; DI/DRAIN
396   0253          SA4 = AND('RC2-PIG':DEV_STAT,16) ; DI/VN/DR
397   0254          SA5 = AND('RC2-PIG':DEV_STAT,32) ; INTERLOCK
398   0255          SA6 = AND('RC2-PIG':DEV_STAT,64) ; LOW FLOW
399   0256          GT0 = AND(PB2GLOBAL:DEV_TMP1,2)
400         ;
401         ;       END LOGIC FOR GRAPHICS
402         ;
403         ;       BEGIN FQIC-RESET:
404         ;
405         ;       Function:       For FQIC loops that have a CALC block which performs the
406         ;                       integration, set the output of the CALC block to zero
407         ;                       after the RC2-SAMP device is turned off.
408         ;
409   0258          IF AND(DEVON,2) THEN SD0 = 1                    ; If on (2), set device stat flag
410   0259          IF AND(DEVON,1) & AND(SD0,1) THEN               ; For on-off transition
411                   BEGIN
412   0260            SD0 = 0
413   0261            PRINT AD1,"... FQIC-RC2 LAST READING =  %f GAL",CALOUT
414   0262            CALOUT = 0.0
415                   END
416         ;
417         ;       END FQIC-RESET
418         ;
419         ;       DEVs To Lock In Manual:
420         ;
421   0264          IF AND('URW-ALARM':DEV_FLG,64) THEN PUTMANL 'URW-ALARM'
422   0266          IF AND('RC2-SAMP':DEV_FLG,64) THEN PUTMANL 'RC2-SAMP'
423   0268          IF AND('RC2-PIG':DEV_FLG,64) THEN PUTMANL 'RC2-PIG'
424         ;
425   0269          GOTO AGAIN
426         ;
427         ;       E N D   P R O G R A M   L O O P
428         ;
429         ;       BEGIN ANALOG WHITE SUBROUTINE:
430         ;
431         ; ANLG:  IF !SELECT() THEN GOTO AGAIN
432   0270  ANLG:   IF (PTYPE = 1) THEN                                ; If some alarm
433                   BEGIN                                   ;be1
434   0271            BITVAL = PAI:AI_ALMCT
435   0272            IF !(AND(BITVAL,16)) THEN                       ; If not inhibited
436                     BEGIN                                 ;be2
437   0274              IF PSTAT = 3 THEN LET PSTAT = 0
438   0275              IF (PREV) THEN                                ; If rev acting, LOLO, or LO
439                       BEGIN                               ;be3
440   0276                IF (PSTAT = 0) THEN                         ; If not previously alarmed
```

```
441                              BEGIN                                    ;be4
442    0278                      IF (PAI:AI_MEAS <= PALARM) THEN GOSUB ALM
443                              END                                      ;en4
444                           ELSE
445                              BEGIN                                    ;be5
446    0279                      IF (PSTAT = 1) THEN                              ; If previously alarmed
447                                 BEGIN                                 ;be6
448    0281                         IF (FTRIP4) THEN GOSUB ACK1                   ; If ACK flag set, then ack it
449    0284                         IF PTYPE !=1 THEN IF AND(PAI:AI_ALMCT,32) THEN GOSUB ACK1
450                                 END                                   ;en6
451    0286                      IF (PAI:AI_MEAS >= PALARM + PAI:AI_DEAD) THEN GOSUB CLR
452                              END                                      ;en5
453                           END                                         ;en3
454                        ELSE                                                ; If fail-safe, HIHI, or HI
455                           BEGIN                                       ;be10
456    0287                   IF (PSTAT = 0) THEN
457                              BEGIN                                    ;be11
458    0289                      IF (PAI:AI_MEAS >= PALARM) THEN GOSUB ALM
459                              END                                      ;en11
460                           ELSE
461                              BEGIN                                    ;be12
462    0290                      IF (PSTAT = 1) THEN
463                                 BEGIN                                 ;be13
464    0292                         IF (FTRIP4) THEN GOSUB ACK1
465    0295                         IF PTYPE != 1 THEN IF AND(PAI:AI_ALMCT,32) THEN GOSUB ACK1
466                                 END                                   ;en13
467    0297                      IF (PAI:AI_MEAS <= PALARM - PAI:AI_DEAD) THEN GOSUB CLR
468                              END                                      ;en12
469                           END                                         ;en10
470    0298                NXT = PSTAT
471    0300                IF (PPASS) & (PSTAT = 2) THEN NXT = 0
472                        END                                           ;en2
473                        ELSE
474                          BEGIN
475    0301                  LET PSTAT = 3
476    0302                  NXT = 0
477                          END
478    0303             WHITE = OR(WHITE,NXT)
479                     END                                              ;en1
480    0304             ELSE LET PSTAT = 0
481    0305             STEP PALARM BY 1
482    0306             STEP PSTAT BY 1
483    0307             STEP PTYPE BY 1
484    0308             STEP PREV BY 1
485    0309             STEP PPASS BY 1
486    0310             ILOOP = ILOOP + 1
487    0311             RETURN
488             ;
489             ;       END ANALOG WHITE SUBROUTINE
490             ;
491             ;       BEGIN CLEAR, ALARM, ACKNOWLEDGE, MESSAGE, ERROR, AND TERMINATE SUBROUTINES:
492             ;
493    0312     CLR:    LET PSTAT = 0
494    0315             IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
495    0316             LET FCLEAR = 1
```

```
496   0317          GOSUB ACKK
497   0318          RETURN
498         ;
499   0319  ALM:    LET PSTAT = 1
500   0322          IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
501   0323          PDEV:DEV_TMP0 = 2
502   0324          LAMP ADI,SKDKEY,2
503   0326          IF AND(PTYPE,3)THEN FALMON = 1
504   0327          CRIT
505   0328          GOSUB ACKK
506   0329          RETURN
507         ;
508   0330  ACK1:   LET PSTAT = 2
509   0331          LET FACK = 1
510         ;
511         ;       ALARM TYPER STRINGS:
512         ;
513   0333  ACKK:   IF ILOOP = 0 THEN STRNG0 = "URW SAMPLER FLOW LOW"
514   0335          IF ILOOP = 1 THEN STRNG0 = "URW SAMPLER RADIATION HI"
515   0337          IF ILOOP = 2 THEN STRNG0 = "URW SYSTEM FAIL"
516   0339          IF ILOOP = 3 THEN STRNG0 = "RC2 RADIATION BYPASS ON "
517   0341          IF ILOOP = 15 THEN STRNG0 = "E-C-1 CONDENSER URW FLOW LO"
518   0343          IF ILOOP = 16 THEN STRNG0 = "E-C-1 CONDENSER URW FLOW HI"
519   0345          IF ILOOP = 17 THEN STRNG0 = "RC1-PIG LOW FLOW"
520   0347          IF ILOOP = 24 THEN STRNG0 = "START URWBATCH PGM "
521   0349          IF ILOOP = 25 THEN STRNG0 = "URWBATCH WRONG UNIT"
522   0351          IF ILOOP = 26 THEN STRNG0 = "TERMINATE URWBATCH "
523         ;
524   0352  AMSG:   IF PTYPE = 1 THEN
525                    BEGIN
526   0354             IF (FACK = 0) & (FCLEAR = 0) & (FSTART = 0) THEN PRINT ADI,"WHITE ALARM %p",STRNG0
527   0355             IF (FACK) THEN
528                        BEGIN
529   0356                 PRINT ADI,"%p ALARM ACKNOWLEDGE",STRNG0
530   0357                 LET FACK = 0
531                        END
532                    END
533   0358          IF (FCLEAR) THEN
534                    BEGIN
535   0359             PRINT ADI,"%p ALARM CLEAR",STRNG0
536   0360             LET FCLEAR = 0
537                    END
538   0361          IF (FSTART) THEN
539                    BEGIN
540   0362             LET FSTART = 0
541   0363             PRINT ADI,"%p",STRNG0
542                    END
543   0364          RETURN
544         ;
545   0365  ERRSUB: IF ERRN != ERRNUM() THEN
546                    BEGIN
547   0366             TIMERON STIME0
548   0367             TIMERCLR STIME0
549   0368             LET FSN0 = 1
550   0369             LET ERRN = ERRNUM()
```

```
551   0370              LET ERRL = ERRLIN()
552   0371              PRINT ADI,"ERROR # %i AT LINE %i",errn,errl
553                     END
554   0372        ERSCLR
555   0373              GOTO AGAIN
556             ;
557             ; TERM:  LET FABORT = 0
558   0374      TERM:    LET STRNG0 = "URWBATCH TERMINATED    "
559   0375               LET ILOOP = STAR + 2
560   0376               LET FSTART = 1
561   0377               LET STRNG0 = "TERM"
562   0378               GOSUB ACKK
563                     END
```

Total Errors Detected 0 Total Warnings 0
Sequence Program Size = 7248 Bytes, 453 Pages
23 Forward Reference(s)

VENTBATCH

```
;       V E N T B A T C H . S E Q                                VERSION 2
;
;       Function:       Sequence program for Lamp function,
;                       message display on Graphic number 020,
;                       and printer messages.
;
;       Written By      : R. N. Kersey
;       Date            : 05-31-92
;       Facility        : 242-A Evaporator Bldg.
;       Machine         : VAX 4104-A
;       Op. System      : VAX/VMS 5.5-2AH
;       D/3 Version     : 6.3-2 BASELINE
;       Compiler        : SIC
;       Build command   : @[320,20]VENTBATCH
;
;       UNIT = 13, SKDKEY = 39
;
;       Assoc. UPF:     [300,1]_PRCESS.___      PRCESS.LST
;       Assoc. RPF:     [300,1]_242A.___        242A.LST
;       SDB:            [300,1]PCMOA.SDB        PCMOASDB.CMD
;
;       POINTER   SDB VAR.      FUNCTION
;       -------   ---------     ----------------------------------------
;       PSTAT     INTn          Current EPN alarm state is an integer tied to
;                               graphic flashing string of value:
;       .         (n=0-100)             0 = previously not alarmed
;       .                               1 = previously alarmed, not ack
;       .                               2 = previously alarmed, is ack
;       .                               3 = alarm is inhibited
;       .                               4 = input is off scan
;       .                               5 = input is bad
;       .         LEAKn         Special INT variable for multiple white alarms on an EPN
;       PDIN      'EPN'         ---
;       PTYPE     TYPEn         alarm type 0 = none, 1 = white, 2 = yellow,
;                                  3 = yellow and yellow, 4 = red,
;                                  6 = yellow and red
;       PREV      FRVn          Reverse acting (not fail safe) flag (1=set)
;       PSKID     SKIDn         Used for analog alarms
;       PPASS     FPASn         Turn lamp off when alarm is ACKnowledged
;       PALARM    FALMn         Analog alarm flag
;       ---       FNING         Unused error flag
;
; MODIFICATION LOG:
;
;       29-DEC-92       Added ammonia monitor signals, O/C Display      M. Teats
;                       Bar code for TDIC-HC11, and comments.
;
;       31-DEC-92       Fixed white alarm print message.        M. Teats
;
;       27-JAN-93       Added HTR-PRMI, INTERLOK statuses for           M. Teats
;                       HC1-INLK1 (SB3, SB5)
;
;       08-JUN-93       Added comments and baselined for        M. Teats
;                       D/3 version 3.3.  CCN# 93JUN01-01.
;
```

```
56          ;       15-DEC-94       Revised header to indicate th enew VMS          J. Berger
57          ;                       system. Rebaselined document. Remove
58          ;                       obsolite Select() statements.
59          ;
60          ;  END MOD LOG
61          ;
62          UNIT    VVENT
63          ;
64                  COMMON PB2
65                  FLAGBITS == PSPAR0
66                  EXC_TMP1 == 'EXC1-INLK':DEV_TMP1
67          ;
68    0000          NOP
69    0001          STAR = 35                       ; Start message
70    0002          LET SKDKEY = 39         ; SKID Key
71    0003          LET UNITT = 13                  ; SEQ Unit number
72    0004          ASSIGN PDEV = 'VV-ALARM' ; Alarm device
73    0005          ASSIGN PTYPE = TYPE35
74    0006          LET ADI = 0                     ; Alarm destination index
75    0007          LET FSNO = 0                    ; No alarms yet
76    0008          TIMEROFF STIME0         ; Alarm timer off
77    0009          TIMERCLR STIME0         ; Alarm timer clear
78    0010          LET STRNG0 = " "                ; Blank 30 character string
79    0011          LET ERRN = 0                    ; Set error number to zero
80    0012          LET ERRL = 0                    ; Set error line to zero
81    0013          LET FTRIP4 = 0                  ; FTRIP4 = 1 when audible horn is to be turned on
82    0014          GETUNIT TRIP                    ; Which sequence unit is program running in
83    0015          IF TRIP != UNITT THEN           ; Abort if running in wrong seq. unit
84                     BEGIN
85    0016             FSTART = 1
86    0017             ILOOP = STAR + 1
87    0018             PTYPE = 0
88    0019             GOSUB ACKK
89    0020             GOTO TERM
90                     END
91          ;       IF !SELECT() THEN GOTO AGAIN
92          ;
93          ;       B E G I N   I N I T I A L I Z I N G :
94          ;
95    0021          STRNG0 = "INIT"
96    0022    INIT: LET FABORT = 0                  ; If FABORT = 1 then abort program
97    0023          LET FSTART = 0                  ; Program start flag = 0
98    0024          LAMP ADI,SKDKEY,0               ; Turn off the annunciator key
99    0025          LET LSG0 = " "                  ; Clear the string variables
100   0026          LET LSG1 = " "
101   0027          LET LSG2 = " "
102   0028          LET LSG3 = " "
103   0029          LET LSG4 = " "
104   0030          PDEV:DEV_TMP0 = 1                               ; Set alarm device to non-alarm state
105   0031          IF AND(PDEV:DEV_ALMCT,16) THEN                  ; If alarm device is inhibited, remove the inhibit
106   0032            PDEV:DEV_ALMCT=XOR(PDEV:DEV_ALMCT,16)
107   0033          IF AND(PDEV:DEV_ALMCT,32) THEN                  ; If alarm device is ack, clear it
108   0034             PDEV:DEV_ALMCT = XOR(PDEV:DEV_ALMCT,32)
109         ;
110         ;       First, set all flags and integers:
```

```
111             ;
112             ;       ILOOP 0-16 are DINS, 17-32 are AINS red and yellow, 33-34 are AIN white
113             ;
114             ;       EPN             TYPE    INT     ILOOP
115             ;
116             ;       PDSH-FC51       4       INT0    0
117             ;       PDSL-FC51       2       INT1    1
118             ;       ZS-FC5-1        2       INT2    2
119             ;       PDSH-FC52       4       INT3    3
120             ;       ZS-FC5-2        2       INT4    4
121             ;       YS-EXC-1        0       INT5    5
122             ;       YS-EXC3RL       2       INT6    6
123             ;       TSHH-HC13       4       INT7    7
124             ;       RXS-VVALP       2       INT8    8
125             ;       RSH-VVALP       4       INT9    9
126             ;       RXS-VVB/G       2       INT10   10
127             ;       RSH-VVB/G       4       INT11.  11
128             ;       QXS-VV-1        2       INT12   12.
129             ;       (SPARE)         0       INT13   13      SKIP
130             ;       PDSH-AS-1       2       INT14   14
131             ;       PDSL-AS-2       2       INT15   15
132             ;       AXS-NH3-1       2       INT16   16
133             ;
134             ;       PDI-FC5-2       2       SKID17  17
135             ;       PDI-FC5-1       2       SKID18  18
136             ;       PDI-DUC1        2       SKID19  19
137             ;       II-EXC-1        0       SKID20  20
138             ;       PDI-FC6-1       2       SKID21  21
139             ;       FQI-AS-5        0       SKID22  22
140             ;       TI-HC1-1        0       SKID23  23
141             ;       TI-HC1-2        2       SKID24  24
142             ;       TI-FC5-1        2       SKID25  25
143             ;       TDIC-HC11       6       SKID26  26
144             ;       FI-AS-5         6       SKID27  27
145             ;       PDI-FC512       6       SKID28  28
146             ;       RI-VVALP        4       SKID29  29
147             ;       RI-VVB/G        4       SKID30  30
148             ;       II-AS-1         0       SKID31  31
149             ;       AI-NH3-1        2       SKID32  32
150             ;
151             ;       II-EXC-1        1       SKID33  33      LO      1.0
152             ;       II-AS-1         1       SKID34  34      LO      2.0
153             ;
154     0035            ASSIGN PDIN = 'PDSH-FC51'                       ; Assign pointer to first DIN
155     0036            ASSIGN PSTAT = INT0
156     0037            ASSIGN PTYPE = TYPE0
157     0038            ASSIGN PREV = FRV0
158     0039            ASSIGN PSKID = SKID0
159     0040            ASSIGN PPASS = FPAS0
160             ;
161     0041            FOR ILOOP = 0 TO 34
162     0042            LET PPASS = 0
163     0043            LET PREV = 0
164     0045            IF PSTAT > 5 THEN PSTAT = 0                     ; Reset current status if undefined
165     0046            LET PSKID = 0                                   ; Default previous status to no alarms
```

```
166   0047          LET PTYPE = 0                                   ; Default to no alarms
167   0049          IF ILOOP = 13 THEN GOTO SKP                     ; (Spare)
168   0050          IF ILOOP <= 16 THEN                             ; For DINs
169                    BEGIN
170   0051             LET PTYPE = 2                                ; Default to yellow alarm
171   0052             IF AND(PDIN:DIN_DFLG,3)THEN                  ; Check for alarms (must be a red or yellow)
172                       BEGIN
173   0053                LET PTYPE = 2                             ; Default to yellow alarm
174   0055                IF AND(PDIN:DIN_DFLG,1) THEN LET PREV = 1 ; Reverse acting
175   0057                IF AND(PDIN:DIN_DFLG,4) THEN LET PTYPE = 4 ; Set to red critical
176                       END
177                    END
178   0059          IF (ILOOP >= 17 & ILOOP <= 32) THEN PTYPE = 4   ; For AINs red and yellow
179   0061    SKP:  IF ILOOP < 16 THEN STEP PDIN BY 1               ; For DINs, point to next DIN
180   0063          IF ILOOP >= 33 THEN PTYPE = 1                   ; For AIN white
181   0064          STEP PPASS BY 1
182   0065          STEP PREV BY 1
183   0066          STEP PSTAT BY 1
184   0067          STEP PSKID BY 1
185   0068          STEP PTYPE BY 1
186   0069          NEXT ILOOP
187         ;
188   0070          TYPE5 = 0              ; YS-EXC-1
189   0071          TYPE13 = 0             ; (SPARE)
190   0072          TYPE20 = 0             ; II-EXC-1
191   0073          TYPE22 = 0             ; FQI-AS-5
192   0074          TYPE23 = 0             ; TI-HC1-1
193   0075          TYPE31 = 0             ; II-AS-1
194         ;
195   0076          FRV0 = 0               ; PDSH-FC51
196   0077          FRV1 = 0               ; PDSL-FC51
197   0078          FRV2 = 1               ; ZS-FC5-1
198   0079          FRV3 = 0               ; PDSH-FC52
199   0080          FRV4 = 1               ; ZS-FC5-2
200   0081          FRV5 = 0               ; YS-EXC-1
201   0082          FRV6 = 0               ; YS-EXC3RL
202   0083          FRV7 = 0               ; TSHH-HC13
203   0084          FRV8 = 0               ; RXS-VVALP
204   0085          FRV9 =0                ; RSH-VVALP
205   0086          FRV10 = 0              ; RXS-VVB/G
206   0087          FRV11 = 0              ; RSH-VVB/G
207   0088          FRV12 = 0              ; QXS-VV-1
208   0089          FRV13 = 0              ; (SPARE)
209   0090          FRV14 = 0              ; PDSH-AS-1
210   0091          FRV15 = 0              ; PDSL-AS-2
211   0092          FRV16 = 0              ; AXS-NH3-1
212   0093          FRV33 = 1              ; II-EXC-1 LO
213   0094          FRV34 = 1              ; II-AS-1 LO
214         ;
215   0095          AI0 = 1.0              ; II-EXC-1 LO    ILOOP 33
216   0096          AI1 = 2.0              ; II-AS-1 LO     ILOOP 34
217         ;
218   0097          AI19 = 0.0             ; For TDIC-HC11 PID output, O/C Display Bar
219         ;
220         ;       E N D   I N I T I A L I Z I N G
```

```
221           ;
222           ;     S E N D   P R O G R A M   S T A R T   M E S S A G E
223           ;
224    0098           ANALOG = 0
225    0099           ILOOP = STAR
226    0100           FSTART = 1
227    0101           GOSUB ACKK
228    0102           ON ERROR GOSUB ERRSUB
229           ;
230           ;     B E G I N   P R O G R A M   L O O P :
231           ;
232    0103   AGAIN:  MARK STEP "WTUNTL",0,0
233           ;      WAITUNTIL SELECT() ¦ FABORT      ; Wait until micro is selected or program aborted
234           ;      IF (FABORT) THEN GOTO TERM       ; Terminate program is FABORT flag set
235    0104           MARK STEP "RNNING",0,0
236    0105           WAIT 1
237    0106           IF (FSNO) THEN                   ; If alarm flag is set
238    0107              IF STIMEO > 60 THEN           ; If alarm flag is active for 1 minute
239                          BEGIN                     ; Reset timer and turn off flag
240    0108                  TIMEROFF STIMEO
241    0109                  TIMERCLR STIMEO
242    0110                  FSNO = 0
243    0111                  ERRN = 0
244    0112                  ERRL = 0
245                          END
246           ;
247           ;      If VV-ALARM $T1 is alarmed and it's ALMCT indicates alarms are acknowledged:
248           ;
249    0113           IF (PDEV:DEV_TMPO = 2) & AND(PDEV:DEV_ALMCT,32) THEN
250                     BEGIN
251    0114              PDEV:DEV_TMPO = 1             ; Put VV-ALARM in ACKNWLDG state
252    0115              FTRIP4 = 1                    ; Set for Unit alarm acknowledgement, system to graphic
253    0116              ACK = 1                       ; Set for Unit alarm acknowledgement, graphic to system
254    0117              CRIT
255    0118              LAMP ADI,SKDKEY,1             ; Turn LAMP on
256                      END
257           ;
258           ;      BEGIN DINs:
259           ;
260    0119           ASSIGN PDIN = 'PDSH-FC51'
261    0120           ASSIGN PTYPE = TYPEO
262    0121           ASSIGN PREV = FRVO
263    0122           ASSIGN PSTAT = INTO
264    0123           ASSIGN PPASS = FPASO
265    0124           LET CURENT = 0                   ; Used to set lamp status
266    0125           LET LAST = 0
267    0126           LET ANALOG = 0
268    0127           LET NXT = 0
269    0128           LMPVAR = 0
270    0129           BITVAL = 0
271           ;
272    0130   LOP:    FOR ILOOP = 0 TO 16
273           ;      IF !SELECT() THEN GOTO AGAIN
274    0131           IF ( PTYPE > 0 ) THEN                              ; If some type of alarm
275                     BEGIN                                        ;be1
```

```
276  0133              IF AND(PDIN:DIN_ALMCT,16) THEN LET PSTAT = 3   ; If EPN inhibited
277  0135              IF AND(PDIN:DIN_ALMCT,4) THEN LET PSTAT = 5           ; If EPN is bad input
278  0137              IF !AND(PDIN:DIN_FLG,8) THEN LET PSTAT = 4            ; If EPN is disabled
279  0138              IF ( PSTAT < 3 ) THEN                                 ; If not inhibited
280                         BEGIN                                          ;be2
281  0139                   IF (PSTAT = 1) THEN
282                            BEGIN                                       ;be3
283  0140                      IF (PTYPE != 1) THEN
284  0142                         IF AND(PDIN:DIN_ALMCT,32)  THEN GOSUB ACK1
285  0144                      IF (FTRIP4) THEN GOSUB ACK1
286                            END                                         ;en3
287  0147                   IF PSTAT = 0 THEN IF(PDIN:DIN_VAL = PREV) THEN GOSUB ALM
288  0148                   IF (PSTAT) THEN
289                               BEGIN                                    ;be4
290  0150                         IF (PDIN:DIN_VAL != PREV) THEN GOSUB CLR
291  0151                         NXT = PSTAT
292  0153                         IF (PPASS) & (PSTAT = 2 ) THEN NXT = 0
293  0154                         LAST = OR(LAST,NXT)
294                               END                                      ;en4
295                         END                                            ;en2
296  0156                ELSE IF !AND(PDIN:DIN_ALMCT,16)&!AND(PDIN:DIN_ALMCT,4)&AND(PDIN:DIN_FLG,8)THEN LET PSTAT = 0
297                      END                                               ;en1
298  0157              STEP PDIN BY 1
299  0158              STEP PTYPE BY 1
300  0159              STEP PREV BY 1
301  0160              STEP PSTAT BY 1
302  0161              STEP PPASS BY 1
303  0162              NEXT ILOOP
304            ;
305            ;       END DINs
306            ;
307            ;       BEGIN ANALOG RED AND YELLOWS:
308            ;
309  0163              ASSIGN PAI = 'PDI-FC5-2'
310  0164              ASSIGN PREV = FRV17
311  0165              ASSIGN PSTAT = INT17.
312  0166              ASSIGN PSKID = SKID17
313  0167              ASSIGN PTYPE = TYPE17
314  0168              FOR JLOOP = 17 TO 32
315            ;       IF !SELECT() THEN GOTO AGAIN
316  0169              IF (PTYPE) THEN                                       ; If some alarm exists
317                      BEGIN                                 ;be1
318  0170                BITVAL = PAI:AI_ALMCT
319  0171                IF !AND(BITVAL,16) THEN                           ; If not inhibited
320                         BEGIN                              ;be2
321  0172                   IF AND(BITVAL,15) THEN                         ; If alarm exists and is current
322                            BEGIN                           ;be3
323  0174                      IF AND(BITVAL,32) THEN NXT = 2              ; If alarm is ack
324                            ELSE                                        ; If alarm is unack
325                               BEGIN                        ;be4
326  0175                         IF PSKID !=BITVAL THEN
327                                  BEGIN                     ;be5
328  0177                            IF BITVAL < 3 THEN FALMON = 1
329  0178                            LET PSKID = BITVAL
330  0179                            PDEV:DEV_TMP0 = 2
```

```
331   0180                              NXT = 1
332                                     END                          ;en5
333   0181                          ELSE NXT = 1
334                                 END                              ;en4
335                               END                                ;en3
336                             ELSE                                       ; Else there is no alarm
337                                BEGIN
338   0182                         NXT = 0
339   0183                         LET PSKID = 0
340                                END
341                             END                                  ;en2
342                           ELSE                                       ; Else alarm is inhibited
343                              BEGIN
344   0184                       NXT = 0
345   0185                       LET PSKID = 0
346                              END
347   0186                    ANALOG = OR(ANALOG,NXT)
348                           END                                    ;en1
349   0187                  STEP PAI BY 1
350   0188                  STEP PREV BY 1
351   0189                  STEP PSTAT BY 1
352   0190                  STEP PSKID BY 1
353   0191                  STEP PTYPE BY 1
354   0192                  NEXT JLOOP
355             ;
356             ;           END ANALOG RED AND YELLOWS
357             ;
358             ;           BEGIN ANALOG WHITE ALARMS:
359             ;
360   0193      WHT:        ASSIGN PALARM = AI0
361   0194                  ASSIGN PSTAT = INT33
362   0195                  ASSIGN PREV = FRV33
363   0196                  ASSIGN PPASS = FPAS33
364   0197                  ASSIGN PTYPE = TYPE33
365   0198                  WHITE = 0
366             ;
367   0199                  ILOOP = 33
368   0200                  ASSIGN PAI = 'II-EXC-1'
369   0201                  GOSUB ANLG                ; 33
370   0202                  ASSIGN PAI = 'II-AS-1'
371   0203                  GOSUB ANLG                ; 34
372                         ;
373                         ;  For hi or lo DP alarm on F-AS-1, set alarm box to yellow:
374                         ;
375   0205                  IF (INT14 != 3 & INT14 >0) ¦ (INT15 !=3 & INT15 > 0 ) THEN LET LEAK3 = 1
376   0206                     ELSE LET LEAK3 = 0
377             ;
378             ;           END ANALOG WHITE ALARMS
379             ;
380   0207                  MARK STEP "BELOW",0,0
381   0208                  FTRIP4 = 0
382   0210                  IF LAST > 3 THEN PRINT ADI," LAMP VARIABLE IS GREATER THAN 3!"
383   0211                  CURENT = OR(LAST,WHITE)
384   0212                  CURENT = OR(CURENT,ANALOG)
385   0214                  IF AND(CURENT,1) THEN LMPVAR = 2
```

```
386  0216              ELSE IF AND(CURENT,2) THEN LMPVAR = 1
387  0217                   ELSE LMPVAR = 0
388  0218          LAMP ADI,SKDKEY,LMPVAR
389  0219          MARK STEP "BOTTOM",0,0
390  0220          NOCRIT
391           ;
392           ;    BEGIN O/C DISPLAY BAR:
393           ;
394           ;    Fucntion:      Upon loss of permissive, put the control loop in manual
395           ;                   and set the PID block output to zero.
396           ;
397  0222          IF AND('EX-C-1':DEV_STAT,2) THEN DLEAK3 = 1                    ; Permissive must be Hi (1)
398  0223          IF !AND('EX-C-1':DEV_STAT,2) & AND(DLEAK3,1) THEN         ; Permissive lost and 1st pass
399                   BEGIN
400  0224             PUTMANL 'TDIC-HC11'
401  0225             PRINT ADI," PERMISSIVE LOST >>> TDIC-HC11 PUT IN MANUAL"
402  0226             PUTOUT 'TDIC-HC11',AI19
403  0227             PRINT ADI," PERMISSIVE LOST >>> TDIC-HC11 OUTPUT SET TO ZERO"
404  0228             DLEAK3 = 0
405                   END
406           ;
407           ;    END O/C DISPLAY BAR
408           ;
409           ;    BEGIN LOGIC FOR GRAPHICS:
410           ;
411  0229          ASSIGN FLAGBITS = AT23
412  0230          BVAL = 128
413  0231              FOR IN1 = 23 TO 28
414  0232              FLAGBITS = AND(BVAL,EXC_TMP1)
415  0233              BVAL=BVAL{1
416  0234              STEP FLAGBITS BY 1
417  0235              NEXT IN1
418           ;
419  0236          AT31 = AND('EX-C-1':DEV_TMP1,32768)
420  0237          AT0 = 'YS-EXC3RL':DIN_VAL
421  0238          AT1 = AND('EX-C-1':DEV_TMP0,1)
422  0239          AT2 = 'YS-EXC-1':DIN_VAL
423  0240          GT14 = AND(PB2GLOBAL:DEV_TMP0,16384)
424  0241          SA0 = AND('EX-C-1':DEV_STAT,16)
425  0242          SA1 = AND('EX-C-1':DEV_STAT,4)
426  0243          SB3 = AND('HC1-INLK1':DEV_STAT,8)          ; HTR-PRMI
427  0244          SB5 = AND('HC1-INLK1':DEV_STAT,32)         ; INTERLOK
428           ;
429           ;    END LOGIC FOR GRAPHICS
430           ;
431           ;    DEVICES TO LOCK IN MANUAL:
432           ;
433  0246          IF AND('VV-ALARM':DEV_FLG,64) THEN PUTMANL 'VV-ALARM'
434  0248          IF AND('EX-C-1':DEV_FLG,64) THEN PUTMANL 'EX-C-1'
435           ;
436  0249          GOTO AGAIN
437           ;
438           ;    E N D   P R O G R A M   L O O P
439           ;
440           ;    BEGIN ANALOG WHITE SUBROUTINE:
```

```
441           ;
442           ; ANLG:  IF !SELECT() THEN GOTO AGAIN
443     0250  ANLG:    IF (PTYPE = 1) THEN                                  ; If some alarm
444                      BEGIN                                          ;be1
445     0251             BITVAL = PAI:AI_ALMCT
446     0252             IF !(AND(BITVAL,16)) THEN                            ; If not inhibited
447                        BEGIN                                        ;be2
448     0254               IF PSTAT = 3 THEN LET PSTAT = 0
449     0255               IF (PREV) THEN                                     ; If rev acting, LOLO, or LO
450                          BEGIN                                      ;be3
451     0256                 IF (PSTAT = 0) THEN                              ; If not previously alarmed
452                            BEGIN                                    ;be4
453     0258                   IF (PAI:AI_INVL <= PALARM) THEN GOSUB ALM
454                            END                                      ;en4
455                        ELSE
456                            BEGIN                                    ;be5
457     0259                   IF (PSTAT = 1) THEN                            ; If previously alarmed
458                              BEGIN                                  ;be6
459     0261                     IF (FTRIP4) THEN GOSUB ACK1                ; If ACK flag set, then ack it
460     0264                     IF PTYPE !=1 THEN IF AND(PAI:AI_ALMCT,32) THEN GOSUB ACK1
461                              END                                    ;en6
462     0266                   IF (PAI:AI_INVL >= PALARM + PAI:AI_DEAD) THEN GOSUB CLR
463                            END                                      ;en5
464                        END                                          ;en3
465                      ELSE                                               ; If fail-safe, HIHI, or HI
466                          BEGIN                                      ;be10
467     0267                 IF PSTAT = 0  THEN
468                            BEGIN                                    ;be11
469     0269.                  IF (PAI:AI_INVL >= PALARM) THEN GOSUB ALM
470                            END                                      ;en11
471                        ELSE
472                            BEGIN                                    ;be12
473     0270                   IF (PSTAT = 1) THEN
474                              BEGIN                                  ;be13
475     0272                     IF (FTRIP4) THEN GOSUB ACK1
476     0275                     IF PTYPE != 1 THEN IF AND(PAI:AI_ALMCT,32) THEN GOSUB ACK1
477                              END                                    ;en13
478     0277                   IF (PAI:AI_INVL <= PALARM - PAI:AI_DEAD) THEN GOSUB CLR
479                            END                                      ;en12
480                        END                                          ;en10
481     0278               NXT = PSTAT
482     0280               IF (PPASS) & (PSTAT = 2) THEN NXT = 0
483                        END                                          ;en2
484                        ELSE
485                          BEGIN
486     0281                 LET PSTAT = 3
487     0282                 NXT = 0
488                          END
489     0283             WHITE = OR(WHITE,NXT)
490                      END                                            ;en1
491     0284             ELSE LET PSTAT = 0
492     0285             STEP PALARM BY 1
493     0286             STEP PSTAT BY 1
494     0287             STEP PTYPE BY 1
495     0288             STEP PREV BY 1
```

```
496   0289          STEP PPASS BY 1
497   0290          ILOOP = ILOOP + 1
498   0291          RETURN
499         ;
500         ;       END ANALOG WHITE SUBROUTINE
501         ;
502         ;       BEGIN CLEAR, ALARM, ACKNOWLEDGE, MESSAGE, ERROR, AND TERMINATE SUBROUTINES:
503         ;
504   0292  CLR:    LET PSTAT = 0
505   0295          IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
506   0296          LET FCLEAR = 1
507   0297          GOSUB ACKK
508   0298          RETURN
509         ;
510   0299  ALM:    LET PSTAT = 1
511   0302          IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
512   0304          IF AND(PTYPE,3)THEN FALMON = 1
513   0305          PDEV:DEV_TMP0 = 2
514   0306          LAMP ADI,SKDKEY,2
515   0307          CRIT
516   0308          GOSUB ACKK
517   0309          RETURN
518         ;
519   0310  ACK1:   LET PSTAT = 2
520   0311          LET FACK = 1
521         ;
522         ;       ALARM TYPER STRINGS:
523         ;
524   0313  ACKK:   IF ILOOP = 0 THEN STRNG0 = "VVENT 1ST HEPA DP HI "
525   0315          IF ILOOP = 1 THEN STRNG0 = "VVENT 1ST HEPA DP LO "
526   0317          IF ILOOP = 2 THEN STRNG0 = "VVENT 1ST DMPR CLOSED"
527   0319         IF ILOOP = 3 THEN STRNG0 = "VVENT 2ND HEPA DP HI "
528   0321          IF ILOOP = 4 THEN STRNG0 = "VVENT 2ND DMPR CLOSED"
529   0323          IF ILOOP = 5 THEN STRNG0 = "VVENT EXHAUSTER ON   "
530   0325          IF ILOOP = 6 THEN STRNG0 = "VVENT EXH FAN LOCAL  "
531   0327          IF ILOOP = 7 THEN STRNG0 = "VVENT HEATER OVERTEMP"
532   0329          IF ILOOP = 8 THEN STRNG0 = "VVENT EXH ALPHA  FAIL"
533   0331          IF ILOOP = 9 THEN STRNG0 = "VVENT EXH ALPHA RD HI"
534   0333          IF ILOOP = 10 THEN STRNG0 = "VVENT  BETA/GAM FAIL"
535   0335          IF ILOOP = 11 THEN STRNG0 = "VVENT BETA/GAMMA HI "
536   0337          IF ILOOP = 12 THEN STRNG0 = "VVENT RAD SYS FAIL  "
537         ;       IF ILOOP = 13 THEN STRNG0 = "VVNT AIR SPL PMP OFF"              ; (SPARE)
538   0339          IF ILOOP = 14 THEN STRNG0 = "P-AS-1 DELTA P HI   "
539   0341          IF ILOOP = 15 THEN STRNG0 = "P-AS-1 DELTA P LOW  "
540   0343          IF ILOOP = 16 THEN STRNG0 = "AMMONIA MONITOR FAIL"
541   0345          IF ILOOP = 33 THEN STRNG0 = "II-EXC-1 AMPS LO    "
542   0347          IF ILOOP = 34 THEN STRNG0 = "II-AS-1 AMPS LO     "
543   0349          IF ILOOP = 35 THEN STRNG0 = "START VENTBATCH PGM "
544   0351          IF ILOOP = 36 THEN STRNG0 = "VENTBATCH WRONG UNIT"
545   0353          IF ILOOP = 37 THEN STRNG0 = "TERMINATE VENTBATCH "
546         ;
547   0354  AMSG:   IF PTYPE = 1 THEN
548                    BEGIN
549   0356             IF (FACK = 0) & (FCLEAR = 0) & (FSTART = 0) THEN PRINT ADI,"WHITE ALARM %p",STRNG0
550   0357             IF (FACK) THEN
```

```
551                         BEGIN
552     0358                PRINT ADI,"%p ALARM ACKNOWLEDGE",STRNGO
553     0359                LET FACK = 0
554                         END
555                       END
556     0360              IF (FCLEAR) THEN
557                         BEGIN
558     0361                PRINT ADI,"%p ALARM CLEAR",STRNGO
559     0362                LET FCLEAR = 0
560                         END
561     0363              IF (FSTART) THEN
562                         BEGIN
563     0364                LET FSTART = 0
564     0365                PRINT ADI,"%p",STRNGO
565                         END
566     0366              RETURN
567             ;
568             ;         BEGIN ERROR SUBROUTINE:
569             ;
570     0367    ERRSUB:   IF ERRN = ABS(ERRNUM()) THEN
571                       BEGIN
572     0368              FSNO = 1
573             ;
574             ; errors are handled differently depending on their severity:
575             ;
576             ; ERR
577             ; NO.     ERROR DESCRIPTION                                    ACTION
578             ; ---     -------------------------------------------          -------------
579             ;  19     PROGRAM ALREADY IN KICK SUBROUTINE                   (rekick)
580             ;  32     UNDEFINED COMMUNICATIONS PACKET TYPE                 (ignore)
581             ;  41    PCM DB HEADER NOT LOADED                      (ignore)
582             ;  69    IPN NOT DEFINED                                      (ignore)
583             ;  162   NO SUCH PROGRAM IN THIS PCM                          (ignore)
584             ;  165    SEQUENCE PROGRAM ALREADY RUNNING             (ignore)
585             ;  170    SERIAL NUMBER MISMATCH                               (wait forever)
586             ;  179   ILLEGAL SUBROUTINE STACK CLEAR                       (ignore)
587             ;  187    BACKUP PCM IS DOWN                                   (ignore)
588             ;  193    NO SUCH PROGRAM IN THIS PCM                          (ignore)
589             ;  198    ILLEGAL SUBROUTINE STACK NESTING ERROR               (ignore)
590             ;  199    ILLEGAL RDF NAME                                     (wait forever)
591             ;  200    RDF NOT LOADED IN MEMORY                     (wait forever)
592             ;  201   RECIPE LOAD NOT ENABLED                              (wait forever)
593             ;  222   BOARDS DEAD                                          (ignore)
594             ;  226    BOARD ERROR (ie: broken skid panel)                  (ignore)
595             ;  245    BOTH PCMS DOWN, SENT REDUNDANT HIGHWAY ACCESS        (ignore)
596             ;
597     0369              ERRN = ABS(ERRNUM())                                 ; get error number
598     0370              ERRL = ERRLIN()                                      ; get error line number
599             ;
600             ; if error is not serious, clear and return to program:
601             ;
602     0371              IF (ERRN=187) ¦ (ERRN=32) ¦ (ERRN=69) ¦ (ERRN=222) THEN
603     0372                  ERSRET
604     0373              IF (ERRN=41) ¦ (ERRN=165) ¦ (ERRN=162) ¦ (ERRN=179) THEN
605     0374                  ERSRET
```

```
606    0375            IF (ERRN=226) ¦ (ERRN=198) ¦ (ERRN=245) THEN
607    0376                ERSRET
608    0377            IF (ERRN=193) ¦ (ERRN=19) THEN
609    0378                ERSRET
610            ;
611            ; else report the error, and take corrective action:
612            ;
613    0379            PRINT ADI,"ERROR NUMBER %i AT LINE %i ***",ERRN,ERRL
614                    END
615    0380            ERSCLR                                          ; Corrective action
616    0381            GOTO AGAIN                                      ; Goto done and cleanup
617            ;
618            ;       END ERROR SUBROUTINE
619            ;
620            ; TERM:  LET FABORT = 0
621    0382    TERM:   LET STRNG0 = "VENTBATCH TERMINATED    "
622    0383            LET ILOOP = STAR + 2
623    0384            LET FSTART = 1
624    0385            LET STRNG0 = "TERM"
625    0386            GOSUB ACKK
626                    END

       Total Errors Detected      0     Total Warnings      0
            Sequence Program Size = 7728 Bytes, 483 Pages
                 21 Forward Reference(s)
```

PCBATCH

Sequence Instruction Compiler Listing 15:36:03 24-Feb-95 Page 1

```
;       P C B A T C H . S E Q                                   VERSION 3
;
;       Function:       Sequence program for Lamp function,
;                       message display on Graphic number 018,
;                       and printer messages.
;
;       Written By      : R. N. Kersey
;       Date            : 05-31-92
;       Facility        : 242-A Evaporator Bldg.
;       Machine         : VAX 4104-A
;       Op. System      : VAX/VMS 5.5-2AH
;       D/3 Version     : 6.3-2 BASELINE
;       Compiler        : SIC
;       Build command   : @[320,20]PCBATCH
;
;       UNIT = 14, SKDKEY = 37
;
;       Assoc. UPF:     [300,1]_PRCESS.___      PRCESS.LST
;       Assoc. RPF:     [300,1]_242A.___        242A.LST
;       SDB:            [300,1]PCM0A.SDB        PCM0ASDB.CMD
;
;       POINTER   SDB VAR.      FUNCTION
;       -------   ---------     ----------------------------------------
;       PSTAT     INTn          Current EPN alarm state is an integer tied to
;                               graphic flashing string of value:
;                 (n=0-100)             0 = previously not alarmed
;                 .                     1 = previously alarmed, not ack
;                 .                     2 = previously alarmed, is ack
;                 .                     3 = alarm is inhibited
;                 .                     4 = input is off scan
;                 .                     5 = input is bad
;                 .       LEAKn         Special INT variable for multiple white alarms on an EPN
;       PDIN      'EPN'         ---
;       PTYPE     TYPEn         alarm type 0 = none, 1 = white, 2 = yellow,
;                                  3 = yellow and yellow, 4 = red,
;                                  6 = yellow and red
;       PREV      FRVn          Reverse acting (not fail safe) flag (1=set)
;       PSKID     SKIDn         Used for analog alarms
;       PPASS     FPASn         Turn lamp off when alarm is ACKnowledged
;       PALARM    FALMn         Analog alarm flag
;       ---       FNING         Unused error flag
;
;  MODIFICATION LOG:
;
;TA     note:  lines starting with ';OLD' are Original Lines to be
;TA             reimplemented when operational testing is done.  Lines
;TA             with ';TA' are Temporary Additions to be removed when
;TA             operational testing is done.  Related changes are in FEEDBATCH.
;TA             These changes are for level control of FIC-C1005 instead of
;TA             FIC-CA1-1.
;
;       28-AUG-92       Added NOCRIT below MARK STEP BOTTOM             M. Teats
;
;       12-SEP-92       Changed LI-C100-2 to LI-C100-1 and changed      M. Teats
;                       alarm level from 0.95 to 0.98
```

```
56              ;
57              ;       30-DEC-92       Fixed white alarm print message.        M. Teats
58              ;
59              ;       26-JAN-93       Added P-C100 and A-C100 device statuses,  M. Teats
60              ;                       (SB3,SB4,SB5,SC3,SC4,SC5)
61              ;
62              ;       01-APR-93       In INITIALIZATION fixed flags ASSIGNED to       M. Teats
63              ;                       pointers.  Changed FALM0 and SKID0 to
64              ;                       FALM5 and SKID5.
65              ;
66              ;       08-JUN-93       Added comments.  CCN# 93JUN01-01.               M. Teats
67              ;
68              ;-mct1+  24-JUN-93      Changed "CONFIRM ON" to "SHUTDOWN" in alarm      M. Teats
69              ;                       typer string section, and baselined for D/3
70              ;                       version 3.3.  CCN# 93JUN24-01.
71              ;
72              ;-mct2+  12-JAN-94      Deleted II-P-C100 white alarm per               M. Teats
73              ;                       CCN# 94JAN12-01.
74              ;
75              ;       15-DEC-94       Revised header to indicate the new VMS          J. Berger
76              ;                       system. Rebaselined document. Removed
77              ;                       obsolite Select() statement.
78              ;
79              ;
80              ;  END OF MOD LOG
81              ;
82              UNIT    PC
83              ;
84                      COMMON PB2,CA1,EC1,IX                   ;TA; CA1 for synch FIC-C1005 > LIC
85                      DEVON == 'RC3-SAMP':DEV_STAT            ; For FQIC-RESET device on/off check
86                      CALOUT == 'FQIC-RC3':CB_OTVL:3          ; For FQIC-RESET calc block outvalue
87                      RC3_PIG == 'RC3-PIG';DEV_TMP0
88                      PC100 == 'P-C100':DEV_TMP1
89                      PCINLK == 'PC-INLK1':DEV_TMP0
90                      C100_ACK == 'PC-ALARM':DEV_TMP0         ; ALARM ACK FOR SKID KEY 37
91                      IX_ACK == 'IX-ALARM':DEV_TMP0           ; ALARM ACK FOR SKID KEY 36
92                      FIC1SP == 'FIC-C1005':PID_SPVL:2 ;TA; For synch FIC-C1005 > LIC
93              ;
94      0000            NOP
95      0001            LET STAR = 24                   ; Start message
96      0002            LET UNITT = 14                  ; SEQ Unit number
97      0003            LET SKDKEY = 37         ; SKID Key
98      0004            ASSIGN PDEV = 'PC-ALARM' ; Alarm device
99      0005            ASSIGN PTYPE = TYPE24
100     0006            LET ADI = 0                     ; Alarm destination index
101     0007            LET FSN0 = 0                    ; No alarms yet
102     0008            TIMEROFF STIME0         ; Alarm timer off
103     0009            TIMERCLR STIME0         ; Alarm timer clear
104     0010            STRNG0 = " "                    ; Blank 30 character string
105     0011            LET ERRN = 0                    ; Set error number to zero
106     0012            LET ERRL = 0                    ; SEt error line to zero
107     0013            LET FTRIP4 = 0                  ; FTRIP4 = 1 when audible horn is to be turned on
108     0014            GETUNIT TRIP                    ; Which sequence unit is program running in
109     0015            IF TRIP != UNITT THEN           ; Abort if running in wrong seq. unit
110                        BEGIN
```

```
111   0016              FSTART = 1
112   0017              ILOOP = STAR + 1
113   0018              PTYPE = 0
114   0019              GOSUB ACKK
115   0020              GOTO TERM
116                     END
117           ;       IF !SELECT() THEN GOTO AGAIN
118           ;
119           ;       B E G I N   I N I T I A L I Z I N G :
120           ;
121           ;               EPN             VAR    LOOP#     COLOR
122           ;               ---------       ----   -----     -----
123           ;       DEVs:   PC ALARM
124           ;               A-C100
125           ;               P-C100
126           ;               PC-INLK1
127           ;
128           ;       DINs:   YS-A-C100       INT0   0         W
129           ;               LSL-C1001       INT1   1         Y
130           ;               LSL-SP-1        INT2   2         Y
131           ;               YS-P-C100       INT3   3         W
132           ;               YS-RC3-2        INT4   4         Y
133           ;                                                       VAR    LOOP#     ALARM SETPT
134           ;                                                       ---    -----     -----------
135           ;       AINs:   II-A-C100       SKID5  5         -
136           ;               WFIC-C100       SKID6  6         RY
137           ;               LI-C100-G       SKID7  7         -
138           ;               LI-C100-1       SKID8  8         W       INT17  17        .95
139           ;               LI-C100-2       SKID9  9         -
140           ;               TI-C100-1       SKID10 10        Y
141           ;-mct2+         II-P-C100       SKID11 11        W       INT18  18        5.0    (not used)
142           ;               FIC-C1005       SKID12 12        -
143           ;               FQI-C100        SKID13 13        -
144           ;               PI-FC1-3        SKID14 14        -
145           ;               PDI-FC1-1       SKID15 15        RY
146           ;               PI-FC1-4        SKID16 16        -
147           ;
148   0021            STRNG0 = "INIT"
149   0022    INIT:   LET FABORT = 0                  ; If FABORT = 1 then abort program
150   0023            LET FSTART = 0                  ; Program start flag = 0
151   0024            LET SD0 = 0                     ; For FQIC-RESET RC3-SAMP device status flag
152   0025            LAMP AD1,SKDKEY,0               ; Turn off the annunciator key
153   0026            LET LSG0 = " "                  ; Clear the string variables
154   0027            LET LSG1 = " "
155   0028            LET LSG2 = " "
156   0029            LET LSG3 = " "
157   0030            LET LSG4 = " "
158   0031            C100_ACK = 1                                    ; Set alarm device to non-alarm state
159   0032            IF AND('PC-ALARM':DEV_ALMCT,16) THEN            ; If alarm device is inhibited, remove the inha
160   0033              'PC-ALARM':DEV_ALMCT=XOR('PC-ALARM':DEV_ALMCT,16)
161   0034            IF AND('PC-ALARM':DEV_ALMCT,32) THEN            ; If alarm device is ack, clear it
162   0035               'PC-ALARM':DEV_ALMCT = XOR('PC-ALARM':DEV_ALMCT,32)
163           ;
164           ;       First, set all flags and integers:
165           ;
```

```
166   0036          ASSIGN PDIN = 'YS-A-C100'                    ; Assign pointer to first DIN
167   0037          ASSIGN PPASS = FPAS0
168   0038          ASSIGN PREV = FRV0
169   0039          ASSIGN PTYPE = TYPE0
170   0040          ASSIGN PSTAT = INT0
171         ;
172   0041          FOR ILOOP = 0 TO 4                           ; For DINs
173   0042          LET PPASS = 0
174   0043          LET PREV = 0
175   0045          IF PSTAT > 5 THEN PSTAT = 0                  ; Reset current status if undefined
176   0046          LET PTYPE = 2                                ; Default to yellow alarm
177   0047          IF AND(PDIN:DIN_DFLG,3) THEN                 ; Check for alarms (must be a red or yellow)
178                   BEGIN
179   0048            LET PTYPE = 2                              ; Default to Yellow alarm
180   0050            IF AND(PDIN:DIN_DFLG,1) THEN LET PREV = 1  ; Reverse acting
181   0052            IF AND(PDIN:DIN_DFLG,4) THEN LET PTYPE = 4 ; Set to red critical
182                   END
183   0053          STEP PPASS BY 1
184   0054          STEP PREV BY 1
185   0055          STEP PTYPE BY 1
186   0056          STEP PSTAT BY 1
187   0057          NEXT ILOOP
188         ;
189   0058          ASSIGN PALARM = FALM5                        ; For AIs
190   0059          ASSIGN PSKID = SKID5
191   0060          FOR JLOOP = 5 TO 16
192   0061          PALARM = 0
193   0062          PSKID = 0
194   0063          STEP PALARM BY 1
195   0064          STEP PSKID BY 1
196   0065          NEXT JLOOP
197         ;
198   0066          TYPE0 = 1
199   0067          TYPE3 = 1
200   0068          TYPE6  = 6
201   0069          TYPE10 = 2
202   0070          TYPE14 = 2
203   0071          TYPE15 = 6
204   0072          TYPE17 = 1
205         ;-mct2+ TYPE18 = 1              commented out
206         ;
207   0073          FRV0 = 0
208   0074          FRV1 = 0
209   0075          FRV2 = 0
210   0076          FRV3 = 0
211   0077          FRV4 = 1
212   0078          FRV17 = 1
213         ;-mct2+ FRV18 = 1              commented out
214         ;
215   0079          FPAS0 = 1
216   0080          FPAS3 = 1
217         ;
218   0081          AI0 = .98
219         ;-mct2+ AI1 = 5.0              commented out
220         ;
```

```
221.          ;OLD     FLG14 = 0                ; Flag for PID auto mode, for synch PID
222           ;OLD     FLG15 = 0                ; Flag for PID manual mode, for synch PID
223     0082           FLG13 = 0                ;TA; PID casc (1) flag, for synch FIC-C1005 > LIC
224     0083           FLG14 = 0                ;TA; PID auto (1) flag, for synch FIC-C1005 > LIC
225     0084           FLG15 = 0                ;TA; PID manual (1) flag, for synch FIC-C1005 > LIC
226           ;
227           ;        E N D   I N I T I A L I Z I N G
228           ;
229           ;        S E N D   P R O G R A M   S T A R T   M E S S A G E
230           ;
231     0085           ANALOG = 0
232     0086           ILOOP = STAR
233     0087           FSTART = 1
234     0088           FPASS = 1
235     0089           GOSUB ACKK
236     0090           ON ERROR GOSUB ERRSUB
237           ;
238           ;        B E G I N   P R O G R A M   L O O P :
239           ;
240     0091  AGAIN:   MARK STEP "WTUNTL",0,0
241           ;        IF SELECT() THEN FNING = 1
242           ;           ELSE FNING = 0
243           ;        WAITUNTIL SELECT() ¦ FABORT      ; Wait until micro selected or program aborted
244           ;        IF (FABORT) THEN GOTO TERM       ; Terminate program if abort flag set
245           ;
246     0092           MARK STEP "RNNING",0,0
247     0093           WAIT 1
248     0094           IF (FSNO) THEN                   ; If alarm flag is set
249     0095              IF STIMEO > 60 THEN           ; If alarm flag active for 1 minute
250                          BEGIN                     ; Reset timer and clear flag
251     0096                 TIMEROFF STIMEO
252     0097                 TIMERCLR STIMEO
253     0098                 LET FSNO = 0
254     0099                 LET ERRN = 0
255     0100                 LET ERRL = 0
256                          END
257           ;
258           ;        If PC-ALARM $T1 is alarmed and it's ALMCT indicates alarms are acknowledged:
259           ;
260     0101           IF (C100_ACK = 2) & AND('PC-ALARM':DEV_ALMCT,32) THEN
261                       BEGIN
262     0102              C100_ACK = 1                  ; Put PC-ALARM in ack state
263     0103              FTRIPO = 1
264     0104              FTRIP4 = 1                    ; Set for unit alarm ack
265     0105              ACK = 1                       ; Set for unit alarm ack
266     0106              CRIT
267     0107              LAMP ADI,SKDKEY,1             ; Turn lamp on
268                       END
269           ;
270           ;        BEGIN DINs:
271           ;
272     0108           ASSIGN PDIN = 'YS-A-C100'
273     0109           ASSIGN PTYPE = TYPEO
274     0110           ASSIGN PREV = FRVO
275     0111           ASSIGN PSTAT = INTO
```

```
276   0112            ASSIGN PPASS = FPAS0
277   0113            LET LAST = 0
278   0114            LET NXT = 0
279   0115            BITVAL = 0
280           ;
281   0116    LOP:    FOR ILOOP = 0 TO 4
282           ;       IF !SELECT() THEN GOTO AGAIN
283   0117            IF ( PTYPE > 0 ) THEN                                   ; If some type of alarm
284                      BEGIN                                          ;be1
285   0119               IF AND(PDIN:DIN_ALMCT,16) THEN LET PSTAT = 3   ; If EPN is inhibited
286   0121               IF AND(PDIN:DIN_ALMCT,4) THEN LET PSTAT = 5            ; If EPN is bad input
287   0123               IF !AND(PDIN:DIN_FLG,8) THEN LET PSTAT = 4             ; If EPN is disabled
288   0124               IF ( PSTAT < 3 ) THEN                          ; If not inhibited
289                           BEGIN                                     ;be2
290   0125                    IF (PSTAT = 1) THEN
291                              BEGIN                                  ;be3
292   0126                       IF (PTYPE != 1) THEN
293   0128                          IF AND(PDIN:DIN_ALMCT,32) THEN GOSUB ACK1
294   0130                       IF (FTRIP4) THEN GOSUB ACK1
295                              END                                    ;en3
296   0133                    IF PSTAT = 0 THEN IF(PDIN:DIN_VAL = PREV) THEN GOSUB ALM
297   0134                    IF (PSTAT) THEN
298                                 BEGIN                               ;be4
299   0136                          IF (PDIN:DIN_VAL != PREV) THEN GOSUB CLR
300   0137                          NXT = PSTAT
301   0139                          IF (PPASS) & (PSTAT = 2) THEN NXT = 0
302   0140                          LAST = OR(LAST,NXT)
303                                 END                                 ;en4
304                           END                                       ;en2
305   0142                 ELSE IF !AND(PDIN:DIN_ALMCT,16)&!AND(PDIN:DIN_ALMCT,4)&AND(PDIN:DIN_FLG,8)THEN LET PSTAT = 0
306                        END                                          ;en1
307   0143            STEP PDIN BY 1
308   0144            STEP PTYPE BY 1
309   0145            STEP PREV BY 1
310   0146            STEP PSTAT BY 1
311   0147            STEP PPASS BY 1
312   0148            NEXT ILOOP
313           ;
314           ;       END DINs
315           ;
316           ;       BEGIN ANALOG RED AND YELLOWS:
317           ;
318   0149    ALOG:   LET ANALOG = 0
319   0150            ASSIGN PAI = 'II-A-C100'
320   0151            ASSIGN PSKID = SKID5
321   0152            ASSIGN PALARM = FALM5
322   0153            FOR JLOOP = 5 TO 16
323           ;       IF !SELECT() THEN GOTO AGAIN
324   0154            IF !(PALARM) THEN
325                     BEGIN                                  ;be1
326   0155              BITVAL = PAI:AI_ALMCT
327   0156              IF !AND(BITVAL,16) THEN                       ; If not inhibited
328                       BEGIN                                ;be2
329   0157                IF AND(BITVAL,15) THEN                      ; If alarm exists and is current
330                         BEGIN                              ;be3
```

```
331  0159                IF AND(BITVAL,32) THEN NXT = 2          ; If alarm is ack
332                        ELSE                                   ; If alarm is unack
333                        BEGIN                          ;be4
334  0160                  IF PSKID !=BITVAL THEN
335                          BEGIN                        ;be5
336  0162                    IF BITVAL < 3 THEN FALMON = 1
337  0163                    LET PSKID = BITVAL
338  0164                    C100_ACK = 2
339  0165                    LAMP_ADI,SKDKEY,2
340  0166                    NXT = 1
341                          END                          ;en5
342  0167                  ELSE NXT = 1
343                        END                            ;en4
344                      END                              ;en3
345                    ELSE                                       ; Else there is no alarm
346                      BEGIN
347  0168                NXT = 0
348  0169                LET PSKID = 0
349                      END
350                    END                                ;en2
351                    ELSE                                       ; Else alarm is inhibited
352                      BEGIN
353  0170                NXT = 0
354  0171                LET PSKID = 0
355                      END
356  0172              ANALOG = OR(ANALOG,NXT)
357                    END                                ;en1
358  0173          STEP PSKID BY 1
359  0174          STEP PAI BY 1
360  0175          STEP PALARM BY 1
361  0176          NEXT JLOOP
362          ;
363          ;       END ANALOG RED AND YELLOWS
364          ;
365          ;       BEGIN ANALOG WHITE ALARMS:
366          ;
367  0177          WHITE = 0
368  0178  WHT:    ASSIGN PALARM = AI0
369  0179          ASSIGN PSTAT = INT17
370  0180          ASSIGN PTYPE = TYPE17
371  0181          ASSIGN PREV = FRV17
372  0182          ASSIGN PPASS = FPAS17
373  0183          ILOOP = 17
374  0184          ASSIGN PAI = 'LI-C100-1'
375  0185          GOSUB ANLG
376          ;-mct2+ ASSIGN PAI = 'II-P-C100' commented out
377          ;-mct2+ GOSUB ANLG                    commented out
378          ;
379          ;       END ANALOG WHITE ALARMS
380          ;
381  0186          LET FTRIP4 = 0
382  0188          IF LAST > 3 THEN PRINT ADI," LAMP VARIABLE IS GREATER THAN 3!"
383  0189          CURENT = OR(LAST,WHITE)
384  0190          CURENT = OR(CURENT,ANALOG)
385  0192          IF AND(CURENT,1) THEN LMPVAR = 2
```

```
386  0194              ELSE IF AND(CURENT,2) THEN LMPVAR = 1
387  0195                   ELSE LMPVAR = 0
388  0196           LAMP ADI,SKDKEY,LMPVAR
389  0197           MARK STEP "BOTTOM",0,0
390  0198           NOCRIT
391            ;
392            ;      BEGIN LOGIC FOR GRAPHICS:
393            ;
394  0199           SA0 = OR('RSH-RC3-1':DIN_VAL,'RSH-CA1-1':DIN_VAL)
395  0200           SA1 = 'YS-RC3-2':DIN_VAL
396  0201           SA2 = AND('RC3-PIG':DEV_STAT,4)          ; PIG FLSH
397  0202           SA3 = AND('RC3-PIG':DEV_STAT,8)          ; DI/DRAIN
398  0203           SA4 = AND('RC3-PIG':DEV_STAT,32) ; INTERLOCK
399  0204           SA5 = AND('RC3-SAMP':DEV_STAT,8) ; RC3-SAMP  SAMPLE
400            ;
401  0205           SB0 = AND('HV-RC3-3':DEV_STAT,4) ; RESET
402  0206           SB1 = AND('HV-RC3-3':DEV_STAT,16)        ; CHGSTATE
403  0207           SB2 = AND('HV-RC3-3':DEV_STAT,32)        ; INTERLOCK
404            ;
405  0208           SB3 = AND('P-C100':DEV_STAT,4)           ; RESET
406  0209           SB4 = AND('P-C100':DEV_STAT,16)          ; SHUTDOWN
407  0210           SB5 = AND('P-C100':DEV_STAT,32)          ; INTERLOCK
408            ;
409  0211           SC3 = AND('A-C100':DEV_STAT,4)           ; RESET
410  0212           SC4 = AND('A-C100':DEV_STAT,16)          ; SHUTDOWN
411  0213           SC5 = AND('A-C100':DEV_STAT,32)          ; INTERLOCK
412            ;
413  0214           GT0 = AND(PB2GLOBAL:DEV_TMP1,2)
414  0215           AT0 = AND(RC3_PIG,1)                     ; TIMER ON
415  0216           AT1 = AND(RC3_PIG,2)                     ; TIME OUT
416  0217           AT2 = AND(RC3_PIG,4)
417  0218           AT3 = AND(RC3_PIG,8)
418  0219           AT4 = AND(RC3_PIG,16)
419  0220           AT5 = AND('RC3-SAMP':DEV_TMP0,32)
420  0221           AT6 = AND(RC3_PIG,64)
421  0222           AT7 = AND('P-C100':DEV_TMP0,2)
422  0223           AT8 = 'YS-P-C100':DIN_VAL
423  0224           AT9 = XOR(AT0,AT1)
424  0225           AT10= AND(RC3_PIG,1024)
425  0226           AT17= AND('HV-RC3-3':DEV_TMP1,2)
426  0227           AT20= AND('HV-RC3-3':DEV_CMD,2)
427  0228           AT22= AND('HV-RC3-3':DEV_TMP1,32)
428  0229           AT21= AND(AT20,AT22)
429  0230           AT11= AND('RSH-RC3-1':DIN_VAL,AT21)
430  0231           AT23= AND(PC100,128)
431  0232           AT24= AND(PC100,256)
432  0233           AT25= AND(PC100,512)
433  0234           AT26= AND(PC100,1024)
434  0235           AT27= AND(PC100,2048)
435  0236           AT28= AND(PC100,4096)
436  0237           AT29= AND(PC100,8192)
437  0238           AT31= AND(PC100,32768)
438            ;
439  0239           BT1 = AND(PCINLK,2)
440  0240           BT2 = AND(PCINLK,4)
```

```
441   0241          BT3 = AND(PCINLK,8)
442   0242          BT4 = AND(PCINLK,16)
443   0243          BT5 = AND(PCINLK,32)
444         ;
445   0244          GT13 = AND(8192,PB2GLOBAL:DEV_TMP0)
446   0245          CT0 = AND('PC-INLK1':DEV_TMP0,1)
447   0246          CT1 = AND('PC-INLK1':DEV_TMP0,2)
448   0247          CT23 = AND('P-C100':DEV_TMP1,128)
449   0248          CT24 = AND('P-C100':DEV_TMP1,256)
450   0249          CT25 = AND('P-C100':DEV_TMP1,512)
451   0250          CT26 = AND('P-C100':DEV_TMP1,1024)
452   0251          CT27 = AND('P-C100':DEV_TMP1,2048)
453   0252          CT28 = AND('P-C100':DEV_TMP1,4096)
454         ;
455         ;       END LOGIC FOR GRAPHICS
456         ;
457         ;       B E G I N   F Q I C - R E S E T :
458         ;
459         ;       Function:       For FQIC loops that have a CALC block which performs the
460         ;                       integration, set the output of the CALC block to zero
461         ;                       after the RC3-SAMP device is turned off.
462         ;
463   0254          IF AND(DEVON,2) THEN SD0 = 1                    ; If on (2), set device stat flag
464   0255          IF AND(DEVON,1) & AND(SD0,1) THEN               ; For on-off transition
465                    BEGIN
466   0256             SD0 = 0
467   0257             PRINT AD1,"... FQIC-RC3 LAST READING =  %f GAL",CALOUT
468   0258             CALOUT = 0.0
469                    END
470         ;
471         ;       E N D   F Q I C - R E S E T
472         ;
473         ;OLD    B E G I N   S Y N C H   P I D   C A S C A D E D   S E T P T :
474         ;OLD
475         ;OLD    Function:  Closes slave controller's PID (SPID) cascade,
476         ;OLD               sets cascade controller's PID (CPID) to auto.
477         ;OLD               (NOTE:  the continuous database sets the CPID
478         ;OLD               output equal to the SPID setpoint only on a
479         ;OLD               manual to auto transition in the SPID.)
480         ;OLD
481         ;OLD    IF AND('FIC-C1005':CB_FLG:2,64) THEN            ; If in auto
482         ;OLD       BEGIN                                              ;be1
483         ;OLD         IF (FLG14=0) & (FLG15=1) THEN              ; If previously manual
484         ;OLD            BEGIN                                           ;be2
485         ;OLD            BCLOSCASC 'FIC-C1005'
486         ;OLD            IF !AND('WFIC-C100':CB_FLG:2,64) THEN BPUTAUTO 'WFIC-C100'
487         ;OLD            BLET FLG14 = 1
488         ;OLD            BLET FLG15 = 0
489         ;OLD            END                                             ;en2
490         ;OLD       END                                                  ;en1
491         ;OLD    ELSE                                            ; If in manual
492         ;OLD       BEGIN                                              ;be4
493         ;OLD         IF ((FLG14=1)&(FLG15=0)) | ((FLG14=0)&(FLG15=0)) THEN  ; If prev auto or init
494         ;OLD            BEGIN                                           ;be5
495         ;OLD            BLET FLG14 = 0
```

```
496          ;OLD            BLET FLG15 = 1
497          ;OLD            END                                                ;en5
498          ;OLD        END                                                    ;en4
499    0259            IF AND('FIC-C1005':CB_FLG:2,64) THEN            ;TA; If in auto
500                    BEGIN                                                   ;TA;be1
501    0260              IF (FLG14=0) & (FLG15=0) THEN                 ;TA If initializing
502                      BEGIN                                                 ;TA;be2
503    0262              IF AND('FIC-C1005':PID_SPVS:2,64) THEN LET FLG13 = 1  ;TA; If casc closed
504    0263              LET FLG14 = 1                                 ;TA
505    0264              LET FLG15 = 0                                 ;TA
506                      END                                                   ;TA;en2
507    0265              IF AND('FIC-C1005':PID_SPVS:2,64) & (FLG13=0) THEN    ;TA; If clos casc + prev auto-open
508                      BEGIN                                                 ;TA;be3
509    0266                IF !AND('LIC-CA1-1':CB_FLG:1,64) THEN  ;TA; If LIC-CA1-1 man
510                        BEGIN                                               ;TA;be3.1
511    0267                PUTOUT 'LIC-CA1-1',FIC1SP                 ;TA
512    0268                PUTAUTO 'LIC-CA1-1'           ;TA
513                        END                                                 ;TA;en3.1
514    0269                IF !AND('LIC-CA1-2':CB_FLG:1,64) THEN  ;TA; If LIC-CA1-2 man
515                        BEGIN                                               ;TA;be3.2
516    0270                PUTOUT 'LIC-CA1-2',FIC1SP                 ;TA
517    0271                PUTAUTO 'LIC-CA1-2'           ;TA
518                        END                                                 ;TA;en3.2
519    0272              LET FLG13 = 1
520                      END                                                   ;TA;en3
521    0273              IF !AND('FIC-C1005':PID_SPVS:2,64) & (FLG13=1) THEN ;TA; If open casc + prev auto-clos
522                      BEGIN                                                 ;TA;be4
523    0275              IF AND('LIC-CA1-1':CB_FLG:1,64) THEN PUTMANL 'LIC-CA1-1'  ;TA
524    0277              IF AND('LIC-CA1-2':CB_FLG:1,64) THEN PUTMANL 'LIC-CA1-2'  ;TA
525    0278              LET FLG13 = 0
526                      END                                                   ;TA;en4
527    0279              IF (FLG14=0) & (FLG15=1) THEN                 ;TA; If previously manual
528                      BEGIN                                                 ;TA;be5
529    0280              CRIT                                                  ;TA
530    0281              PUTOUT 'LIC-CA1-1',FIC1SP                     ;TA
531    0282              PUTOUT 'LIC-CA1-2',FIC1SP                     ;TA
532    0283              CLOSCASC 'FIC-C1005'                          ;TA
533    0285              IF !AND('LIC-CA1-1':CB_FLG:1,64) THEN PUTAUTO 'LIC-CA1-1' ;TA
534    0287              IF !AND('LIC-CA1-2':CB_FLG:1,64) THEN PUTAUTO 'LIC-CA1-2' ;TA
535    0288              LET FLG13 = 1                                 ;TA
536    0289              LET FLG14 = 1                                 ;TA
537    0290              LET FLG15 = 0                                 ;TA
538    0291              NOCRIT                                                ;TA
539                      END                                                   ;TA;en5
540                  END                                                       ;TA;en1
541                ELSE                                                ;TA; If in manual
542                    BEGIN                                                   ;TA;be7
543    0292              IF (FLG14=1) & (FLG15=0) THEN                 ;TA; If previously auto
544                      BEGIN                                                 ;TA;be8
545    0294              IF AND('LIC-CA1-1':CB_FLG:1,64) THEN PUTMANL 'LIC-CA1-1'  ;TA
546    0296              IF AND('LIC-CA1-2':CB_FLG:1,64) THEN PUTMANL 'LIC-CA1-2'  ;TA
547    0297              LET FLG13 = 0                                 ;TA
548                      END                                                   ;TA;en8
549    0298              IF ((FLG14=1)&(FLG15=0)) ¦ ((FLG14=0)&(FLG15=0)) THEN ;TA; If prev auto or init
550                      BEGIN                                                 ;TA;be9
```

```
551   0299                    LET FLG14 = 0                              ;TA
552   0300                    LET FLG15 = 1                              ;TA
553                           END                                                ;TA;en9
554                       END                                                    ;TA;en7
555           ;OLD
556           ;OLD      E N D   S Y N C H   P I D   C A S C A D E D   S E T P T
557           ;OLD
558   0301            GOTO AGAIN
559           ;
560           ;       E N D   P R O G R A M   L O O P
561           ;
562           ;       BEGIN ANALOG WHITE SUBROUTINE:
563           ;
564           ; ANLG:  IF !SELECT() THEN GOTO AGAIN
565   0302    ANLG:   IF (PTYPE = 1) THEN                                    ; If some alarm
566                     BEGIN                                    ;be1
567   0303              BITVAL = PAI:AI_ALMCT
568   0304              IF !(AND(BITVAL,16)) THEN                            ; If not inhibited
569                       BEGIN                                  ;be2
570   0306                IF PSTAT = 3 THEN LET PSTAT = 0
571   0307                IF (PREV) THEN                                     ; If rev acting, LOLO, or LO
572                         BEGIN                                ;be3
573   0308                  IF (PSTAT = 0) THEN                              ; If previously not alarmed
574                           BEGIN                              ;be4
575   0310                    IF (PAI:AI_INVL <= PALARM) THEN GOSUB ALM
576                           END                                ;en4
577                         ELSE
578                           BEGIN                              ;be5
579   0311                    IF (PSTAT = 1) THEN                            ; If previously alarmed
580                             BEGIN                            ;be6
581   0313                      IF (FTRIP4) THEN GOSUB ACK1                  ; If ACK flag set, then ack it
582   0316                      IF PTYPE !=1 THEN IF AND(PAI:AI_ALMCT,32) THEN GOSUB ACK1
583                             END                              ;en6
584   0318                    IF (PAI:AI_INVL >= PALARM + PAI:AI_DEAD) THEN GOSUB CLR
585                           END                                ;en5
586                       END                                  ;en3
587                     ELSE                                               ; If fail-safe, HIHI, or HI
588                       BEGIN                                ;be10
589   0319                  IF (PSTAT = 0) THEN
590                           BEGIN                              ;be11
591   0321                    IF (PAI:AI_INVL >= PALARM) THEN GOSUB ALM
592                           END                                ;en11
593                         ELSE
594                           BEGIN                              ;be12
595   0322                    IF (PSTAT = 1) THEN
596                             BEGIN                            ;be13
597   0324                      IF (FTRIP4) THEN GOSUB ACK1
598   0327                      IF PTYPE != 1 THEN IF AND(PAI:AI_ALMCT,32) THEN GOSUB ACK1
599                             END                              ;en13
600   0329                    IF (PAI:AI_INVL <= PALARM - PAI:AI_DEAD) THEN GOSUB CLR
601                           END                                ;en12
602                       END                                  ;en10
603   0330              NXT = PSTAT
604   0332              IF (PPASS) & (PSTAT = 2) THEN NXT = 0
605   0333              WHITE = OR(WHITE,NXT)
```

```
606                        END                                          ;en2
607    0335                IF AND(BITVAL,16) THEN LET PSTAT = 3
608                      END                                            ;en1
609    0336              ELSE LET PSTAT = 0
610    0337              STEP PALARM BY 1
611    0338              STEP PSTAT BY 1
612    0339              STEP PTYPE BY 1
613    0340              STEP PREV BY 1
614    0341              STEP PPASS BY 1
615    0342              ILOOP = ILOOP + 1
616    0343              RETURN
617             ;
618             ;        END ANALOG WHITE SUBROUTINE
619             ;
620             ;        BEGIN CLEAR, ALARM, ACKNOWLEDGE, MESSAGE, ERROR, AND TERMINATE SUBROUTINES:
621             ;
622    0344     CLR:     LET PSTAT = 0
623    0347              IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
624    0348              LET FCLEAR = 1
625    0349              GOSUB ACKK
626    0350              RETURN
627             ;
628    0351     ALM:     LET PSTAT = 1
629    0354              IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
630    0355              C100_ACK = 2
631    0357              IF AND(PTYPE,3)THEN FALMON = 1
632    0358              LAMP ADI,SKDKEY,2
633    0359              CRIT
634    0360              GOSUB ACKK
635    0361              RETURN
636             ;
637    0362     ACK1:    LET PSTAT = 2
638    0363              LET FACK = 1
639             ;
640             ;        ALARM TYPER STRINGS:
641             ;
642    0365     ACKK:    IF ILOOP = 0 THEN STRNG0 = "TK-C100 AGITATOR SHUTDOWN"            ;-mct1+
643    0367              IF ILOOP = 1 THEN STRNG0 = "TK-C100 OVERFLOW TRAP LO"
644    0369              IF ILOOP = 2 THEN STRNG0 = "TK-C100 VVENT SEAL POT LVL LO"
645    0371            IF ILOOP = 3 THEN STRNG0 = "P-C100 PUMP SHUTDOWN"          ;-mct1+
646    0373              IF ILOOP = 4 THEN STRNG0 = "P-C100 RAD INTERLOCK BYPASS ON"
647    0375              IF ILOOP = 17 THEN STRNG0 = "LI-C100-1 TK-C100 INTERFACE LO"
648             ;-mct2+  IF ILOOP = 18 THEN STRNG0 = "II-P-C100: PUMP C100 AMPS HI"        commented out
649    0377              IF ILOOP = 24 THEN STRNG0 = "START PCBATCH PGM "
650    0379              IF ILOOP = 25 THEN STRNG0 = "PCBATCH WRONG UNIT"
651    0381              IF ILOOP = 26 THEN STRNG0 = "TERMINATE PCBATCH"
652             ;
653    0382     AMSG:    IF PTYPE = 1 THEN
654                        BEGIN
655    0384                IF (FACK = 0) & (FCLEAR = 0) & (FSTART = 0) THEN PRINT ADI,"WHITE ALARM %p",STRNG0
656    0385                IF (FACK) THEN
657                          BEGIN
658    0386                  PRINT ADI,"%p ALARM ACKNOWLEDGE",STRNG0
659    0387                  LET FACK = 0
660                          END
```

```
661                         END
662     0388            IF (FCLEAR) THEN
663                         BEGIN
664     0389                PRINT ADI,"%p ALARM CLEAR",STRNG0
665     0390                LET FCLEAR = 0
666                         END
667     0391            IF (FSTART) THEN
668                         BEGIN
669     0392                LET FSTART = 0
670     0393                PRINT ADI,"%p",STRNG0
671                         END
672     0394            RETURN
673             ;
674     0395    ERRSUB: IF ERRN != ERRNUM() THEN
675                         BEGIN
676     0396                TIMERON STIME0
677     0397                TIMERCLR STIME0
678     0398                LET FSN0 = 1
679     0399                LET ERRN = ERRNUM()
680     0400                LET ERRL = ERRLIN()
681     0401                PRINT ADI,"ERROR # %i AT LINE %i",errn,errl
682                         END
683     0402            ERSCLR
684     0403            GOTO AGAIN
685             ;
686             ; TERM:  LET FABORT = 0
687     0404    TERM:   LET STRNG0 = "PCBATCH TERMINATED    "
688     0405            LET ILOOP = STAR + 2
689     0406            LET FSTART = 1
690     0407            LET STRNG0 = "TERM"
691     0408            GOSUB ACKK
692                     END

Total Errors Detected      0     Total Warnings      0
     Sequence Program Size = 8112 Bytes, 507 Pages
          19 Forward Reference(s)
```

IXBATCH

Sequence Instruction Compiler Listing 15:35:08 24-Feb-95 Page 1

```
1           ;       I X B A T C H . S E Q                                   VERSION 2
2           ;
3           ;       Function:       Sequence program for Lamp function,
4           ;                       message display on Graphic number 044,
5           ;                       and printer messages.
6           ;
7           ;       Written By      : R. N. Kersey
8           ;       Date            : 05-31-92
9           ;       Facility        : 242-A Evaporator Bldg.
10          ;       Machine         : VAX 4104-A
11          ;       Op. System      : VAX/VMS 5.5-2AH
12          ;       D/3 Version     : 6.3-2 BASELINE
13          ;       Compiler        : SIC
14          ;       Build command   : @[320,20]IXBATCH
15          ;
16          ;       UNIT = 15, SKDKEY = 36
17          ;
18          ;       Assoc. UPF:     [300,1]_PRCESS.___      PRCESS.LST
19          ;       Assoc. RPF:     [300,1]_242A.___        242A.LST
20          ;       SDB:            [300,1]PCM0A.SDB        PCM0ASDB.CMD
21          ;
22          ;       POINTER   SDB VAR.      FUNCTION
23          ;       -------   ---------     ----------------------------------------
24          ;       PSTAT     INTn          Current EPN alarm state is an integer tied to
25          ;                               graphic flashing string of value:
26          ;       .         (n=0-100)             0 = previously not alarmed
27          ;       .                               1 = previously alarmed, not ack
28          ;       .                               2 = previously alarmed, is ack
29          ;       .                               3 = alarm is inhibited
30          ;       .                               4 = input is off scan
31          ;       .                               5 = input is bad
32          ;       .         LEAKn         Special INT variable for multiple white alarms on an EPN
33          ;       PDIN      'EPN'         ---
34          ;       PTYPE     TYPEn         alarm type 0 = none, 1 = white, 2 = yellow,
35          ;                                  3 = yellow and yellow, 4 = red,
36          ;                                  6 = yellow and red
37          ;       PREV      FRVn          Reverse acting (not fail safe) flag (1=set)
38          ;       PSKID     SKIDn         Used for analog alarms
39          ;       PPASS     FPASn         Turn lamp off when alarm is ACKnowledged
40          ;       PALARM    FALMn         Analog alarm flag
41          ;       ---       RNING         Unused error flag
42          ;
43          ;  MODIFICATION LOG:
44          ;
45          ;        30-DEC-92      Fixed white alarm print message.        M. Teats
46          ;
47          ;        08-JUN-93      Added comments.  CCN# 93JUN01-01.              M. Teats
48          ;
49          ;        21-JUN-93      Removed 3 comment lines.  CCN# 93JUN21-02.     M. Teats
50          ;
51          ;-mct1+  29-JUN-93      Remove IBAD on RC3-PIG, set FRV33 to 1,        M. Teats
52          ;                       and baselined for D/3 version 3.3.
53          ;                       CCN# 93JUN29-01.
54          ;
55          ;       15-DEC-94       Revised header to indicate the new VMS         J. Berger
```

```
56          ;                       system. Rebaselined dcument. Removed
57          ;                       obsolite Select() statemens.
58          ;
59          ;  END MOD LOG
60          ;
61          UNIT     IX
62          ;
63                   COMMON PB2,EC1
64                   RC3_PIG == 'RC3-PIG':DEV_TMP0
65                   IX_ACK == 'IX-ALARM':DEV_TMP0
66                   RC3 == WHITE
67          ;
68   0000            GTO = AND(PB2GLOBAL:DEV_TMP1,2)
69   0001            NOP
70   0002            LET STAR   = 24          ; Start message
71   0003            LET SKDKEY = 36          ; SKID Key
72   0004            LET UNITT  = 15          ; SEQ Unit number
73   0005            ASSIGN PDEV = 'IX-ALARM' ; Alarm device
74   0006            ASSIGN PTYPE = TYPE24
75   0007            LET ADI = 0                      ; Alarm destination index
76   0008            LET FSN0 = 0                     ; No alarm yet
77   0009            TIMEROFF STIME0          ; Alarm timer off
78   0010            TIMERCLR STIME0          ; Alarm timer clear
79   0011            STRNG0 = " "                     ; Blank 30 character string
80   0012            LET ERRN = 0                     ; Set error number to zero
81   0013            LET ERRL = 0                     ; Set error line to zero
82   0014            LET FTRIP4 = 0                   ; FTRIP4 = 1 when audible horn is to be turned on
83   0015            GETUNIT TRIP                     ; Which sequence unit is program running in
84   0016            IF TRIP != UNITT THEN            ; Abort if running in wrong seq. unit
85                      BEGIN
86   0017               FSTART = 1
87   0018               ILOOP = STAR + 1
88   0019               PTYPE = 0
89   0020               GOSUB ACKK
90   0021               GOTO TERM
91                      END
92          ;        IF !SELECT() THEN GOTO AGAIN
93          ;
94          ;        B E G I N   I N I T I A L I Z I N G :
95          ;
96   0022            STRNG0 = "INIT"
97   0023   INIT:    LET FABORT = 0                   ; If FABORT = 1 then abort program
98   0024            LET FSTART = 0                   ; Program start flag = 0
99   0025            LET SD0 = 0                      ; For FQIC-RESET RC3-SAMP device status flag
100  0026            LAMP ADI,SKDKEY,0                ; Turn off the annunciator key
101  0027            LET LSG0 = " "                   ; Clear the string variables
102  0028            LET LSG1 = " "
103  0029            LET LSG2 = " "
104  0030            LET LSG3 = " "
105  0031            LET LSG4 = " "
106  0032            PDEV:DEV_TMP0 = 1                                ; Set alarm device to non-alarm state
107  0033            IF AND('PDEV':DEV_ALMCT,16) THEN         ; If alarm device is inhibited, remove the inha
108  0034              'PDEV':DEV_ALMCT=XOR('PDEV':DEV_ALMCT,16)
109  0035            IF AND('PDEV':DEV_ALMCT,32) THEN         ; If alarm device is ack, clear it
110  0036               'PDEV':DEV_ALMCT = XOR('PDEV':DEV_ALMCT,32)
```

```
111             ;
112             ;       First, set all flags and integers:
113             ;
114             ;       DEVs:
115             ;               IX-ALARM
116             ;               HV-RC3-3
117             ;               RC3-SAMP
118             ;               RC3-PIG
119             ;               HV-2-34
120             ;               HV-2-37
121             ;       DINs:
122             ;               ZS-IDX-2        INT0    W
123             ;               PSH-ILF-1       INT1    W
124             ;               YS-RC3-6        INT2    Y
125             ;               RXS-RC3-1       INT3    Y
126             ;               RSH-RC3-1       INT4    R
127             ;               FSL-RC3-1       INT5    -
128             ;               ZS-RC3-4        INT6    W
129             ;               ZS-RC3-5        INT7    W
130             ;               ZS-IDX1NM               -
131             ;               ZS-IDX-1B               -
132             ;               ZS-IDX-3B               -
133             ;               ZS-IDX3NM               -
134             ;               ZS-RC3-3D               -
135             ;               ZS-RC33NM               -
136             ;       AIs:
137             ;               PI-IDX-13       -       8
138             ;               PI-IDX-14       -       9
139             ;               PI-IDX-15       -       10
140             ;               PI-IDX-16       -       11
141             ;               PDI-IDX-2       RY      12
142             ;               PDI-IDX-4       -       13
143             ;               PDI-IDX-3       RY      14
144             ;               PI-IDX-12       -       15
145             ;               PI-IDX-17       -       16
146             ;               PDI-IDX-5       RY      17
147             ;               PDI-IDX-1       RY      18
148             ;               PI-IDX-18       -       19
149             ;               PI-ILS-5        -       20
150             ;               PI-ILS-6        -       21
151             ;               FI-RC3-1        -       22
152             ;               PDI-ILS-1       RY      23
153             ;               FQIC-RC3        -       24
154             ;               FQI-RC3NM       -       25
155             ;               FQI-RC3-D       -       26
156             ;               KY-RC3-1        -       27
157             ;               RI-RC3-1        R       28
158             ;
159     0037            ASSIGN PDIN = 'ZS-IDX-2'
160     0038            ASSIGN PSTAT = INT0
161     0039            ASSIGN PTYPE = TYPE0
162     0040            ASSIGN PREV = FRV0
163     0041            ASSIGN PSKID = SKID0
164     0042            ASSIGN PPASS = FPAS0
165             ;
```

```
166   0043          FOR ILOOP = 0 TO 7                          ; For DINs
167   0044          PPASS = 0
168   0045          PREV  = 0
169   0046          PSTAT = 0
170   0047          PTYPE = 0
171   0048          IF AND(PDIN:DIN_DFLG,3) THEN                ; Check for alarms (must be red or yellow)
172                   BEGIN
173   0049            PTYPE = 2                                 ; Default to yellow alarm
174   0051            IF AND(PDIN:DIN_DFLG,1) THEN PREV = 1   ; Reverse acting
175   0053            IF AND(PDIN:DIN_DFLG,4) THEN PTYPE = 4  ; Set to red critical
176                   END
177   0054          STEP PPASS BY 1
178   0055          STEP PREV BY 1
179   0056          STEP PSTAT BY 1
180   0057          STEP PTYPE BY 1
181   0058          STEP PSKID BY 1
182   0059          NEXT ILOOP
183         ;
184   0060          ASSIGN PALARM = FALM8
185   0061          ASSIGN PSKID  = SKID8
186   0062          FOR JLOOP = 0 TO 32                         ; For AIs
187   0063          PALARM = 0
188   0064          PSKID  = 0
189   0065          PTYPE  = 0
190   0066          STEP PALARM BY 1
191   0067          STEP PSKID BY 1
192   0068          STEP PTYPE BY 1
193   0069          NEXT JLOOP
194         ;
195   0070          TYPE0 = 1
196   0071          TYPE1 = 1
197   0072          TYPE2 = 2
198   0073          TYPE3 = 2
199   0074          TYPE4 = 4
200   0075          TYPE6 = 1
201   0076          TYPE7 = 1
202   0077          TYPE12 = 6
203   0078          TYPE14 = 6
204   0079          TYPE17 = 6
205   0080          TYPE18 = 6
206   0081          TYPE23 = 6
207   0082          TYPE28 = 4
208   0083          TYPE33 = 2
209         ;
210   0084          FRV0 = 1
211   0085          FRV2 = 1
212   0086          FRV6 = 0
213   0087          FRV7 = 0
214   0088          FRV33 = 1       ;-mct1+
215         ;
216         ;       E N D   I N I T I A L I Z I N G
217         ;
218         ;       S E N D   P R O G R A M   S T A R T   M E S S A G E
219         ;
220   0089          ANALOG = 0
```

```
221   0090            ILOOP = STAR
222   0091            FSTART = 1
223   0092            FPASS = 1
224   0093            GOSUB ACKK
225   0094            ON ERROR GOSUB ERRSUB
226           ;
227           ;       B E G I N   P R O G R A M   L O O P :
228           ;
229   0095    AGAIN:  MARK STEP "WTUNTL",0,0
230           ;       IF SELECT() THEN FNING = 1
231           ;         ELSE FNING = 0
232           ;       WAITUNTIL SELECT() ¦ FABORT      ; Wait until micro is selected or program aborted
233           ;       IF (FABORT) THEN GOTO TERM       ; Terminate program if abort flag set
234   0096            MARK STEP "RNNING",0,0
235   0097            WAIT 1
236   0098            FTRIP4 = 0
237   0099            IF (FSN0) THEN                   ; If alarm flag is set
238   0100               IF STIME0 > 60 THEN.          ; If alarm flag active for 1 minute
239                        BEGIN                       ; Reset timer and turn off flag
240   0101                 TIMEROFF STIME0
241   0102                 TIMERCLR STIME0
242   0103                 LET FSN0 = 0
243   0104                 LET ERRN = 0
244   0105                 LET ERRL = 0
245                        END
246           ;
247           ;       If IX-ALARM $T1 is alarmed and it's ALMCT indicates alarms are acknowledged:
248           ;
249   0106            IF AND('IX-ALARM':DEV_ALMCT,32) & IX_ACK = 2 THEN
250                      BEGIN
251   0107               IX_ACK = 1                    ; Put IX-ALARM in ack state
252   0108               FTRIP4 = 1                    ; Set for unit alarm ack
253   0109               ACK = 1                       ; Set for unit alarm ack
254   0110               CRIT
255   0111               LAMP ADI,SKDKEY,1             ; Turn lamp on
256                      END
257           ;
258   0112            LET LAST = 0
259   0113            LET NXT = 0
260   0114            BITVAL = 0
261           ;
262           ;       BEGIN DINs:
263           ;
264   0115    LOP:    ASSIGN PDIN = 'ZS-IDX-2'
265   0116            ASSIGN PTYPE = TYPE0
266   0117            ASSIGN PREV = FRV0
267   0118            ASSIGN PSTAT = INT0
268   0119            ASSIGN PPASS = FPAS0
269   0120            FOR ILOOP = 0 TO 7
270           ;       IF !SELECT() THEN GOTO AGAIN
271   0121            IF ( PTYPE > 0 ) THEN                                  ; If some type of alarm
272                      BEGIN                                          ;be1
273   0123               IF AND(PDIN:DIN_ALMCT,16) THEN LET PSTAT = 3   ; If EPN is inhibited
274   0125               IF AND(PDIN:DIN_ALMCT,4) THEN LET PSTAT = 5            ; If EPN is bad input
275   0127               IF !AND(PDIN:DIN_FLG,8) THEN LET PSTAT = 4             ; If EPN is disabled
```

```
276  0128              IF ( PSTAT < 3 ) THEN                          ; If not inhibited
277                        BEGIN                                      ;be2
278  0129                  IF (PSTAT = 1) THEN
279                           BEGIN                                   ;be3
280  0130                     IF (PTYPE != 1) THEN
281  0132                        IF AND(PDIN:DIN_ALMCT,32) THEN GOSUB ACK1
282  0134                     IF (FTRIP4) THEN GOSUB ACK1
283                           END                                     ;en3
284  0137                  IF PSTAT = 0 THEN IF(PDIN:DIN_VAL = PREV) THEN GOSUB ALM
285  0138                  IF (PSTAT) THEN
286                           BEGIN                                   ;be4
287  0140                     IF (PDIN:DIN_VAL != PREV) THEN GOSUB CLR
288  0141                     NXT = PSTAT
289  0143                     IF (PPASS) & (PSTAT = 2) THEN NXT = 0
290  0144                     LAST = OR(LAST,NXT)
291                           END                                     ;en4
292                        END                                        ;en2
293  0146               ELSE IF !AND(PDIN:DIN_ALMCT,16)&!AND(PDIN:DIN_ALMCT,4)&AND(PDIN:DIN_FLG,8)THEN LET PSTAT = 0
294                     END                                           ;en1
295  0147            STEP PDIN BY 1
296  0148            STEP PTYPE BY 1
297  0149            STEP PREV BY 1
298  0150            STEP PSTAT BY 1
299  0151            STEP PPASS BY 1
300  0152            NEXT ILOOP
301          ;
302          ;       END DINs
303          ;
304  0153            LET STRNG0 = "RUN"
305          ;
306          ;       BEGIN ANALOG RED AND YELLOWS:
307          ;
308  0154    ALOG:   LET ANALOG = 0
309  0155            ASSIGN PAI = 'PI-IDX-13'
310  0156            ASSIGN PSKID = SKID8
311  0157            ASSIGN PALARM = FALM8
312  0158            FOR JLOOP = 8 TO 28
313          ;       IF !SELECT() THEN GOTO AGAIN
314  0159            IF !(PALARM) THEN
315                     BEGIN                                 ;be1
316  0160               BITVAL = PAI:AI_ALMCT
317  0161               IF !AND(BITVAL,16) THEN                          ; If not inhibited
318                        BEGIN                              ;be2
319  0162                  IF AND(BITVAL,15) THEN                        ; If alarm exists and is current
320                           BEGIN                           ;be3
321  0164                     IF AND(BITVAL,32) THEN NXT = 2             ; If alarm is ack
322                              ELSE                                    ; If alarm is unack
323                              BEGIN                        ;be4
324  0165                        IF (PSKID !=BITVAL) THEN
325                                 BEGIN                     ;be5
326  0167                           IF BITVAL < 3 THEN FALMON = 1
327  0168                           LET PSKID = BITVAL
328  0169                           IX_ACK = 2
329  0170                           LAMP ADI,SKDKEY,2
330  0171                           NXT = 1
```

```
331                          END                                    ;en5
332    0172                ELSE NXT = 1
333                        END                                      ;en4
334                      END                                        ;en3
335                    ELSE                                                ; Else there is no alarm
336                      BEGIN
337    0173              NXT = 0
338    0174              LET PSKID = 0
339                      END
340                    END                                          ;en2
341                    ELSE                                                ; Else alarm is inhibited
342                      BEGIN
343    0175              NXT = 0
344    0176              LET PSKID = 0
345                      END
346    0177            ANALOG = OR(ANALOG,NXT)
347                    END                                          ;en1
348    0178          STEP PSKID BY 1
349    0179          STEP PAI BY 1
350    0180          STEP PALARM BY 1
351    0181          NEXT JLOOP
352            ;
353            ;     END ANALOG RED AND YELLOWS
354            ;
355            ;     BEGIN DEVICE ALARM:
356            ;
357    0182    RC3:  RC3 = 0
358    0183          ASSIGN PTYPE = TYPE33
359    0184          ASSIGN PREV  = FRV33
360    0185          ASSIGN PSTAT = INT33
361    0186          ASSIGN PPASS = FPAS33
362            ;     IF !SELECT() THEN GOTO AGAIN
363    0187          IF ( PTYPE > 0 ) THEN                                         ; If some type of alarm
364                    BEGIN                                                ;be1
365    0189            IF AND('RC3-PIG':DEV_ALMCT,16) THEN LET PSTAT = 3           ; If EPN is inhibited
366            ;-mct1+ IF AND('RC3-PIG':DEV_ALMCT,4) THEN BLET PSTAT = 5           ; No IBAD on devices
367    0191            IF !AND('RC3-PIG':DEV_FLG,8) THEN LET PSTAT = 4             ; If EPN is disabled
368    0192            IF ( PSTAT < 3 ) THEN                                ; If not inhibited
369                      BEGIN                                              ;be2
370    0193               IF PSTAT = 1 THEN
371                         BEGIN                                           ;be3
372    0195                 IF AND('RC3-PIG':DEV_ALMCT,32) THEN GOSUB ACK1
373    0197                 IF (FTRIP4) THEN GOSUB ACK1
374                         END                                             ;en3
375    0199               IF !PSTAT THEN IF AND('RC3-PIG':DEV_STAT,64) THEN
376                         BEGIN                                           ;be4
377    0200                 CRIT
378    0201                 LET PSTAT = 1
379    0203                 IF AND('RC3-PIG':DEV_ALMCT,32)THEN 'RC3-PIG':DEV_ALMCT = XOR('RC3-PIG':DEV_ALMCT,32)
380    0204                 IX_ACK = 2
381    0205                 FALMON = 1
382    0206                 LAMP ADI,SKDKEY,2
383    0207                 GOSUB ACKK
384    0208                 NOCRIT
385                         END                                             ;en4
```

```
386  0209                    IF PSTAT THEN
387                            BEGIN                                        ;be5
388  0210                      IF !AND('RC3-PIG':DEV_STAT,64) THEN
389                              BEGIN                                      ;be6
390  0211                        LET PSTAT = 0
391  0212                        LET FCLEAR = 1
392  0213                        GOSUB ACKK
393                              END                                        ;en6
394                            END                                          ;en5
395  0214                    NXT = PSTAT
396  0216                    IF (PPASS) & (PSTAT = 2 ) THEN NXT = 0
397  0217                    RC3 = OR(RC3,NXT)
398                          END                                            ;en2
399  0219                ELSE IF !AND('RC3-PIG':DEV_ALMCT,16)&AND('RC3-PIG':DEV_FLG,8)THEN LET PSTAT = 0
400  0221                IF PSTAT = 5 THEN PSTAT = 0
401                      END                                                ;en1
402           ;
403           ;        END DEVICE ALARM
404           ;
405           ;        BEGIN LOGIC FOR GRAPHICS:
406           ;
407  0222             AT0 = AND(RC3_PIG,1)     ; TIMER ON
408  0223             AT1 = AND(RC3_PIG,2)     ; TIME OUT
409  0224             AT2 = AND(RC3_PIG,4)
410  0225             AT3 = AND(RC3_PIG,8)
411  0226             AT4 = AND(RC3_PIG,16)
412  0227             AT5 = AND('RC3-SAMP':DEV_TMP0,32)
413  0228             AT6 = AND(RC3_PIG,64)
414  0229             AT9 = XOR(AT0,AT1)
415  0230             AT10= AND(RC3_PIG,1024)
416  0231             AT17= AND('HV-RC3-3':DEV_TMP1,2)
417  0232             AT20= AND('HV-RC3-3':DEV_CMD,2)
418  0233             AT22= AND('HV-RC3-3':DEV_TMP1,32)
419  0234             AT21= AND(AT20,AT22)
420  0235             AT11= AND('RSH-RC3-1':DIN_VAL,AT21)
421           ;
422  0236             GT13 = AND(8192,PB2GLOBAL:DEV_TMP0)
423           ;
424  0237             SA0 = OR('RSH-RC3-1':DIN_VAL,'RSH-CA1-1':DIN_VAL)
425  0238             SA2 = AND('RC3-PIG':DEV_STAT,4)          ; PIG FLSH
426  0239             SA3 = AND('RC3-PIG':DEV_STAT,8)          ; DI/DRAIN
427  0240             SA4 = AND('RC3-PIG':DEV_STAT,32) ; INTERLOCK
428  0241             SA5 = AND('RC3-SAMP':DEV_STAT,8) ; RC3-SAMP  SAMPLE
429  0242             SB0 = AND('HV-RC3-3':DEV_STAT,4) ; RESET
430  0243             SB1 = AND('HV-RC3-3':DEV_STAT,16)        ; CHGSTATE
431  0244             SB2 = AND('HV-RC3-3':DEV_STAT,32)        ; INTERLOCK
432  0245             LET FTRIP4 = 0
433           ;
434           ;        END LOGIC FOR GRAPHICS
435           ;
436  0247             IF LAST > 3 THEN PRINT ADI," LAMP VARIABLE IS GREATER THAN 3!"
437  0248             CURENT = OR(LAST,RC3)
438  0249             CURENT = OR(CURENT,ANALOG)
439  0251             IF AND(CURENT,1) THEN LMPVAR = 2
440  0253                ELSE IF AND(CURENT,2) THEN LMPVAR = 1
```

```
441  0254                    ELSE LMPVAR = 0
442  0255          LAMP ADI,SKDKEY,LMPVAR
443  0256          NOCRIT
444  0257          GOTO AGAIN
445          ;
446          ;       E N D   P R O G R A M   L O O P
447          ;
448          ;       BEGIN CLEAR, ALARM, ACKNOWLEDGE, MESSAGE, ERROR, AND TERMINATE SUBROUTINES:
449          ;
450  0258    CLR:  LET PSTAT = 0
451  0261          IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
452  0262          LET FCLEAR = 1
453  0263          GOSUB ACKK
454  0264          RETURN
455          ;
456  0265    ALM:  LET PSTAT = 1
457  0268          IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
458  0269          IX_ACK = 2
459  0271          IF AND(PTYPE,3) THEN FALMON = 1
460  0272          LAMP ADI,SKDKEY,2
461  0273          CRIT
462  0274          GOSUB ACKK
463  0275          RETURN
464          ;.
465  0276    ACK1: LET PSTAT = 2
466  0277          LET FACK = 1
467          ;
468          ;       ALARM TYPER STRINGS:
469          ;
470  0279    ACKK; IF ILOOP = 0 THEN STRNG0 = "ZS-IDX-2   VALVE 1-29 CLOSED"
471  0281          IF ILOOP = 1 THEN STRNG0 = "PSH-ILF-1  FC3 INLET PRES HIGH"
472  0283          IF ILOOP = 2 THEN STRNG0 = "YS-RC3-6   RADIATION BYPASS ON"
473  0285         IF ILOOP = 3 THEN STRNG0 = "RXS-RC3-1  PC RAD SYSTEM FAIL "
474  0287          IF ILOOP = 4 THEN STRNG0 = "RSH-RC3-1  PC SAMPLER RAD HIGH"
475  0289          IF ILOOP = 5 THEN STRNG0 = "FSL-RC3-1  PC SAMPLER FLOW LOW"
476  0291          IF ILOOP = 6 THEN STRNG0 = "VALVE 1-16 TO TK-C100 OPEN"
477  0293          IF ILOOP = 7 THEN STRNG0 = "VALVE 1-17 TO TK-102AW OPEN"
478  0295          IF ILOOP = 33 THEN STRNG0 = "RC3-PIG LOW FLOW "
479  0297          IF ILOOP = 24 THEN STRNG0 = "START PCBATCH PGM "
480  0299          IF ILOOP = 25 THEN STRNG0 = "PCBATCH WRONG UNIT"
481  0301          IF ILOOP = 26 THEN STRNG0 = "TERMINATE PCBATCH"
482          ;
483  0302    AMSG: IF PTYPE = 1 THEN
484                  BEGIN
485  0304            IF (FACK = 0) & (FCLEAR = 0) & (FSTART = 0) THEN PRINT ADI,"WHITE ALARM %p",STRNG0
486  0305            IF (FACK) THEN
487                      BEGIN
488  0306                PRINT ADI,"%p ALARM ACKNOWLEDGE",STRNG0
489  0307                LET FACK = 0
490                      END
491                  END
492  0308          IF (FCLEAR) THEN
493                      BEGIN
494  0309                PRINT ADI,"%p ALARM CLEAR",STRNG0
495  0310                LET FCLEAR = 0
```

```
496                         END
497    0311              IF (FSTART) THEN
498                         BEGIN
499    0312                 LET FSTART = 0
500    0313                 PRINT ADI,"%p",STRNGO
501                         END
502    0314              FACK = 0
503    0315              FCLEAR = 0
504    0316              RETURN
505           ;
506    0317    ERRSUB:   IF ERRN != ERRNUM() THEN
507                         BEGIN
508    0318                 TIMERON STIMEO
509    0319                 TIMERCLR STIMEO
510    0320                 LET FSNO = 1
511    0321                 LET ERRN = ERRNUM()
512    0322                 LET ERRL = ERRLIN()
513    0323                 PRINT ADI,"ERROR # %i AT LINE %i",errn,errl
514                         END
515    0324              ERSCLR
516    0325              GOTO AGAIN
517           ;
518           ; TERM:   LET FABORT = 0
519    0326    TERM:     LET STRNGO = "IXBATCH TERMINATED    "
520    0327              LET ILOOP = STAR + 2
521    0328              LET FSTART = 1
522    0329              LET STRNGO = "TERM"
523    0330              GOSUB ACKK
524                      END

       Total Errors Detected      0      Total Warnings      0
            Sequence Program Size = 6416 Bytes, 401 Pages
                 14 Forward Reference(s)
```

SCBATCH

Sequence Instruction Compiler Listing 15:36:41 24-Feb-95 Page 1

```
1           ;       S C B A T C H . S E Q                                   VERSION 2
2           ;
3           ;       Function:       Sequence program for Lamp function,
4           ;                       message display on Graphic number 014,
5           ;                       and printer messages.
6           ;
7           ;       Written By      : R. N. Kersey
8           ;       Date            : 05-31-92
9           ;       Facility        : 242-A Evaporator Bldg.
10          ;       Machine         : VAX 4104-A
11          ;       Op. System      : VAX/VMS 5.5-2AH
12          ;       D/3 Version     : 6.3-2 BASELINE
13          ;       Compiler        : SIC
14          ;       Build command   : @[320,20]SCBATCH
15          ;
16          ;       UNIT = 16, SKDKEY = 29
17          ;
18          ;       Assoc. UPF:     [300,1]_PRCESS.___      PRCESS.LST
19          ;       Assoc. RPF:     [300,1]_242A.___        242A.LST
20          ;       SDB:            [300,1]PCM0A.SDB        PCM0ASDB.CMD
21          ;
22          ;       POINTER   SDB VAR.      FUNCTION
23          ;       -------   ---------     ----------------------------------------
24          ;       PSTAT     INTn          Current EPN alarm state is an integer tied to
25          ;                               graphic flashing string of value:
26          ;       .         (n=0-100)             0 = previously not alarmed
27          ;       .                               1 = previously alarmed, not ack
28          ;       .                               2 = previously alarmed, is ack
29          ;       .                               3 = alarm is inhibited
30          ;       .                               4 = input is off scan
31          ;       .                               5 = input is bad
32          ;       .         LEAKn         Special INT variable for multiple white alarms on an EPN
33          ;       PDIN      'EPN'         ---
34          ;       PTYPE     TYPEn         alarm type 0 = none, 1 = white, 2 = yellow,
35          ;                                  3 = yellow and yellow, 4 = red,
36          ;                                  6 = yellow and red
37          ;       PREV      FRVn          Reverse acting (not fail safe) flag (1=set)
38          ;       PSKID     SKIDn         Used for analog alarms
39          ;       PPASS     FPASn         Turn lamp off when alarm is ACKnowledged
40          ;       PALARM    FALMn         Analog alarm flag
41          ;       ---       FNING         Unused error flag
42          ;
43          ;  MODIFICATION LOG:
44          ;
45          ;         30-DEC-92       Fixed white alarm print message.          M. Teats
46          ;
47          ;         18-FEB-93       Added BT13 SAMPLE status for RC1-SAMP.            M. Teats
48          ;
49          ;         08-JUN-93       Added comments.  CCN# 93JUN01-01.                 M. Teats
50          ;
51          ;         21-JUN-93       Removed 3 comment lines.  CCN# 93JUN21-02.        M. Teats
52          ;
53          ;-mct1+   29-JUN-93       Remove IBAD on RC3-PIG, and baselined for         M. Teats
54          ;                         D/3 version 3.3.  CCN# 93JUN29-01.
55          ;
```

```
56          ;       15-DEC-94       Revised header to indicate the new VMS          J. Berger
57          ;                       system. Rebaselined document. Removed
58          ;                       obsolite Select() statement.
59          ;
60          ;  END MOD LOG
61          ;
62          UNIT    SC
63          ;
64                  COMMON PB2,WHOLE0
65                  DEVON == 'RC1-SAMP':DEV_STAT            ; For FQIC-RESET device on/off check
66                  CALOUT == 'FQIC-RC1':CB_OTVL:3          ; For FQIC-RESET calc block outvalue
67                  HVEA1 == 'HV-EA1-2':DEV_TMP1
68                  PIG0  == 'RC1-PIG':DEV_TMP0
69                  PIG1  == 'RC1-PIG':DEV_TMP1
70          ;
71  0000            NOP
72  0001            LET STAR = 24                   ; Start message
73  0002            LET SKDKEY = 29         ; SKID Key
74  0003            LET UNITT = 16                  ; SEQ Unit number
75  0004            ASSIGN PDEV = 'SC-ALARM' ; Alarm device
76  0005            ASSIGN PTYPE = TYPE24
77  0006            LET ADI = 0                     ; Alarm destination index
78  0007            LET FSN0 = 0                    ; No alarm yet
79  0008            TIMEROFF STIME0         ; Alarm timer off
80  0009            TIMERCLR STIME0         ; Alarm timer clear
81  0010            LET STRNG0 = " "                ; Blank 30 character string
82  0011            LET ERRN = 0                    ; Set error number to zero
83  0012            LET ERRL = 0                    ; Set error line to zero
84  0013            LET FTRIP4 = 0                  ; FTRIP4 = 1 when audible horn is to be turned on
85  0014            GETUNIT TRIP                    ; Which sequence unit is program running in
86  0015            IF TRIP != UNITT THEN           ; Abort if running in wrong seq. unit
87                     BEGIN
88  0016               FSTART = 1
89  0017               ILOOP = STAR + 1
90  0018               PTYPE = 0
91  0019               GOSUB ACKK
92  0020               GOTO TERM
93                     END
94          ;       IF !SELECT() THEN GOTO AGAIN
95          ;
96          ;       B E G I N   I N I T I A L I Z I N G :
97          ;
98  0021            LET STRNG0 = "INIT"
99  0022    INIT:   LET FABORT = 0                  ; If FABORT = 1 then abort program
100 0023            LET FSTART = 0                  ; Program start flag = 0
101 0024            LET SD0 = 0                     ; For FQIC-RESET RC1-SAMP device status flag
102 0025            LAMP ADI,SKDKEY,0               ; Turn off the annunciator key
103 0026            LET LSG0 = " "                  ; Clear the string variables
104 0027            LET LSG1 = " "
105 0028            LET LSG2 = " "
106 0029            LET LSG3 = " "
107 0030            LET LSG4 = " "
108 0031            PDEV:DEV_TMP0 = 1                               ; Set alarm device to non-alarm state
109 0032            IF AND(PDEV:DEV_ALMCT,16) THEN                  ; If alarm device is inha, remove the inha
110 0033              PDEV:DEV_ALMCT=XOR(PDEV:DEV_ALMCT,16)
```

```
111   0034          IF AND(PDEV:DEV_ALMCT,32) THEN                  ; If alarm device is ack, clear it
112   0035             PDEV:DEV_ALMCT = XOR(PDEV:DEV_ALMCT,32)
113               ;
114               ;   First, set all flags and integers:
115               ;
116   0036          ASSIGN PDIN = 'RSH-EA1-1'
117   0037          ASSIGN PSTAT = INTO
118   0038          ASSIGN PTYPE = TYPEO
119   0039          ASSIGN PREV = FRVO
120   0040          ASSIGN PSKID = SKIDO
121   0041          ASSIGN PPASS = FPASO
122               ;
123   0042          FOR ILOOP = 0 TO 19
124   0043          LET PPASS = 0
125   0044          LET PREV = 0
126   0046          IF PSTAT > 5 THEN PSTAT = 0                     ; Reset current status if undefined
127   0047          LET PSKID = 0                                   ; Default previous status to no alarms
128   0048          LET PTYPE = 0                                   ; Default to no alarm
129   0049          IF ILOOP < 9 THEN                               ; For DINs
130                    BEGIN
131   0050             LET PTYPE = 2                                ; Default to yellow alarm
132   0051             IF AND(PDIN:DIN_DFLG,3) THEN                 ; Check for alarms if defined (must be red or yellow)
133                        BEGIN
134   0052                 LET PTYPE = 2                            ; Default to yellow alarms
135   0054                 IF AND(PDIN:DIN_DFLG,1) THEN LET PREV = 1   ; Reverse acting
136   0056                 IF AND(PDIN:DIN_DFLG,4) THEN LET PTYPE = 4  ; Set to red critical
137                        END
138   0057             STEP PDIN BY 1                               ; Point to next DIN
139                    END
140   0058          ELSE PTYPE = 0
141   0059          STEP PPASS BY 1
142   0060          STEP PREV BY 1
143   0061          STEP PSTAT BY 1
144   0062          STEP PSKID BY 1
145   0063          STEP PTYPE BY 1
146   0064          NEXT ILOOP
147               ;
148   0065          TYPE3 = 0
149   0066          TYPE5 = 0       ; FSL-RC1-1 LOW FLOW ALARM REPLACED BY RC1-PIG ALARM
150   0067          TYPE12 = 4
151   0068          TYPE14 = 4
152   0069          TYPE20 = 2
153               ;
154   0070          FRV0 = 0
155   0071          FRV1 = 0
156   0072          FRV2 = 0
157   0073          FRV3 = 0
158   0074          FRV4 = 1
159   0075          FRV6 = 0
160   0076          FRV7 = 0
161   0077          FRV8 = 0
162   0078          FRV20 = 1
163               ;
164   0079          FPAS20 = 0
165               ;
```

```
166          ;       E N D   I N I T I A L I Z I N G
167          ;
168          ;       S E N D   P R O G R A M   S T A R T   M E S S A G E
169          ;
170    0080          ANALOG = 0
171    0081          ILOOP = STAR
172    0082          FSTART = 1
173    0083          GOSUB ACKK
174    0084          ON ERROR GOSUB ERRSUB
175          ;
176          ;       B E G I N   P R O G R A M   L O O P :
177          ;
178    0085  AGAIN:  MARK STEP "WTUNTL",0,0
179          ; JH01  WAITUNTIL SELECT() ¦ FABORT      ; Wait until micro is selected or program aborted
180          ;       IF (FABORT) THEN GOTO TERM      ; Terminate program if abort flag set
181    0086          MARK STEP "RNNING",0,0
182    0087          WAIT 1
183    0088          IF (FSN0) THEN                  ; If alarm flag is set
184    0089             IF STIME0 > 60 THEN          ; If alarm flag active for 1 minute
185                                                  ; Reset timer and turn off flag
186    0090                BEGIN
187    0091                TIMEROFF STIME0
188    0092                TIMERCLR STIME0
189    0093                FSN0 = 0
190    0094                ERRN = 0
191                         ERRL = 0
192                         END
193          ;
194          ;       If SC-ALARM $T1 is alarmed and it's ALMCT indicates alarms are acknowledged:
195          ;
196    0095          IF (PDEV:DEV_TMP0 = 2) & AND(PDEV:DEV_ALMCT,32) THEN
197                   BEGIN
198    0096           PDEV:DEV_TMP0 = 1              ; Put SC-ALARM in ack state
199          ;        PDEV:DEV_STAT = 1              ; THIS IS TEMPORARY FOR A TEST JFB
200    0097           FTRIP4 = 1                     ; Set for unit alarm ack
201    0098           ACK = 1                        ; Set for unit alarm ack
202    0099           CRIT
203    0100           LAMP ADI,SKDKEY,1              ; Turn lamp on
204                   END
205          ;
206          ;       BEGIN DINs:
207          ;
208    0101          ASSIGN PDIN = 'RSH-EA1-1'
209    0102          ASSIGN PTYPE = TYPE0
210    0103          ASSIGN PREV = FRV0
211    0104          ASSIGN PSTAT = INT0
212    0105          ASSIGN PPASS = FPAS0
213    0106          LET CURENT = 0                  ; Used to set lamp status
214    0107          LET LAST = 0
215    0108          LET ANALOG = 0
216    0109          LET NXT = 0
217    0110          LMPVAR = 0
218    0111          BITVAL = 0
219          ;
220    0112  LOP:    FOR ILOOP = 0 TO 8
221          ;       IF !SELECT() THEN GOTO AGAIN
```

```
221   0113          IF ( PTYPE > 0 ) THEN                                   ; If some type of alarm
222                    BEGIN                                                ;be1
223   0115             IF AND(PDIN:DIN_ALMCT,16) THEN LET PSTAT = 3         ; If EPN is inhibited
224   0117             IF AND(PDIN:DIN_ALMCT,4) THEN LET PSTAT = 5                  ; If EPN is bad input
225   0119             IF !AND(PDIN:DIN_FLG,8) THEN LET PSTAT = 4                   ; If EPN is disabled
226   0120             IF ( PSTAT < 3 ) THEN                                ; If not inhibited
227                         BEGIN                                           ;be2
228   0121                  IF (PSTAT = 1) THEN
229                            BEGIN                                        ;be3
230   0122                     IF (PTYPE != 1) THEN
231   0124                        IF AND(PDIN:DIN_ALMCT,32) THEN GOSUB ACK1
232   0126                     IF (FTRIP4) THEN GOSUB ACK1
233                            END                                          ;en3
234   0129                  IF PSTAT = 0 THEN IF(PDIN:DIN_VAL = PREV) THEN GOSUB ALM
235   0130                  IF (PSTAT) THEN
236                               BEGIN                                     ;be4
237   0132                        IF PDIN:DIN_VAL != PREV THEN GOSUB CLR
238   0133                        NXT = PSTAT
239   0135                        IF (PPASS) & (PSTAT = 2) THEN NXT = 0
240   0136                        LAST = OR(LAST,NXT)
241                               END                                       ;en4
242                         END                                             ;en2
243   0138               ELSE IF !AND(PDIN:DIN_ALMCT,16)&!AND(PDIN:DIN_ALMCT,4)&AND(PDIN:DIN_FLG,8)THEN LET PSTAT = 0
244                      END                                               ;en1
245   0139          STEP PDIN BY 1
246   0140          STEP PTYPE BY 1
247   0141          STEP PREV BY 1
248   0142          STEP PSTAT BY 1
249   0143          STEP PPASS BY 1
250   0144          NEXT ILOOP
251           ;
252           ;     END DINs
253           ;
254           ;     BEGIN ANALOG RED AND YELLOWS:
255           ;
256   0145          ASSIGN PSKID = SKID9
257   0146          ASSIGN PAI = 'FI-EA1-3'
258   0147          FOR JLOOP = 9 TO 19
259           ;     IF !SELECT() THEN GOTO AGAIN
260   0148          IF (PTYPE) THEN                                         ; If some alarm exists
261                    BEGIN                                    ;be1
262   0149             BITVAL = PAI:AI_ALMCT
263   0150             IF !AND(BITVAL,16) THEN                             ; If not inhibited
264                        BEGIN                                ;be2
265   0151                 IF AND(BITVAL,15) THEN                          ; If alarm exists and is current
266                            BEGIN                            ;be3
267   0153                     IF AND(BITVAL,32) THEN NXT = 2              ; If alarm is ack
268                            ELSE                                        ; If alarm is unack
269                               BEGIN                         ;be4
270   0154                        IF PSKID !=BITVAL THEN
271                                   BEGIN                     ;be5
272   0156                            IF BITVAL < 3 THEN FALMON = 1
273   0157                            LET PSKID = BITVAL
274   0158                            PDEV:DEV_TMP0 = 2
275   0159                            NXT = 1
```

```
276                                   END                                ;en5
277    0160                         ELSE NXT = 1
278                                 END                                ;en4
279                               END                                  ;en3
280                            ELSE                                          ; Else there is no alarm
281                               BEGIN
282    0161                       NXT = 0
283    0162                       LET PSKID = 0
284                               END
285                            END                                     ;en2
286                         ELSE                                             ; Else alarm is inhibited
287                            BEGIN
288    0163                    NXT = 0
289    0164                    LET PSKID = 0
290                            END
291    0165                 ANALOG = OR(ANALOG,NXT)
292                         END                                        ;en1
293    0166              STEP PREV BY 1
294    0167              STEP PSTAT BY 1
295    0168              STEP PSKID BY 1
296    0169              STEP PTYPE BY 1
297    0170              STEP PAI BY 1
298    0171              NEXT JLOOP
299            ;
300            ;         END ANALOG RED AND YELLOWS
301            ;
302            ;         BEGIN DEVICE ALARM (RC1-PIG LOW FLOW ALARM):
303            ;
304    0172    RC1:      ASSIGN PTYPE = TYPE20
305    0173              ASSIGN PREV = FRV20
306    0174              ASSIGN PSTAT = INT20
307    0175              ASSIGN PPASS = FPAS20
308            ;         IF !SELECT() THEN GOTO AGAIN
309    0176              IF ( PTYPE > 0 ) THEN
310                         BEGIN                                          ;be1
311    0178                  IF AND('RC1-PIG':DEV_ALMCT,16) THEN LET PSTAT = 3          ; EPN is inha
312            ;-mct1+       IF AND('RC1-PIG':DEV_ALMCT,4) THEN BLET PSTAT = 5          ; No IBAD on devices
313    0180                  IF !AND('RC1-PIG':DEV_FLG,8) THEN LET PSTAT = 4            ; EPN is dis
314    0181                  IF ( PSTAT < 3 ) THEN
315                               BEGIN                                    ;be2
316    0182                        IF PSTAT = 1 THEN
317                                  BEGIN                                 ;be3
318    0183                          IF PTYPE != 1 THEN
319    0185                             IF AND('RC1-PIG':DEV_ALMCT,32) THEN GOSUB ACK1
320    0187                          IF (FTRIP4) THEN GOSUB ACK1
321                                  END                                   ;en3
322    0189                        IF PSTAT = 0 THEN IF 'RC1-PIG':DEV_ALMCT = 7 THEN
323                                  BEGIN                                 ;be4
324    0190                          CRIT
325    0191                          LET PSTAT = 1
326    0192                          'SC-ALARM':DEV_TMP0 = 2
327    0193                          FALMON = 1
328    0194                          LAMP ADI,SKDKEY,2
329    0195                          GOSUB ACKK
330    0196                          NOCRIT
```

```
331                              END                                          ;en4
332     0197                   IF (PSTAT) THEN
333                                BEGIN                                      ;be5
334     0198                       IF !AND('RC1-PIG':DEV_ALMCT,7) THEN
335                                   BEGIN                                   ;be6
336     0199                          LET PSTAT = 0
337     0200                          LET FCLEAR = 1
338     0201                          GOSUB ACKK
339                                   END                                     ;en6
340     0202                       NXT = PSTAT
341     0204                       IF (PPASS) & (PSTAT = 2 ) THEN NXT = 0
342     0205                       LAST = OR(LAST,NXT)
343                                END                                        ;en5
344                             END                                           ;en2
345     0207                ELSE IF !AND('RC1-PIG':DEV_ALMCT,16)&AND('RC1-PIG':DEV_FLG,8)THEN LET PSTAT = 0
346     0209                IF PSTAT = 5 THEN PSTAT = 0
347                         END                                               ;en1
348             ;
349             ;       END DEVICE ALARM (RC1-PIG LOW FLOW ALARM)
350             ;
351     0210            WHITE = 0
352     0211            LET FTRIP4 = 0
353     0213            IF LAST > 3 THEN PRINT ADI," LAMP VARIABLE IS GREATER THAN 3!"
354     0214            CURENT = OR(LAST,WHITE)
355     0215            CURENT = OR(CURENT,ANALOG)
356     0217            IF AND(CURENT,1) THEN LMPVAR = 2
357     0219               ELSE IF AND(CURENT,2) THEN LMPVAR = 1
358     0220                       ELSE LMPVAR = 0
359     0221            LAMP ADI,SKDKEY,LMPVAR
360     0222            MARK STEP "BOTTOM",0,0
361     0223            NOCRIT
362             ;
363             ;       BEGIN LOGIC FOR GRAPHICS:
364             ;
365     0224            AT27 = AND(HVEA1,2048)                  ; $T27 = RSH-RC1-1 OFF
366     0225            AT28 = AND(HVEA1,4096)                  ; $T28 = RI-RC1-1  OFF
367     0226            AT29 = AND(HVEA1,8192)                  ; $T29 = RSH-EA1-1 OFF
368     0227            AT30 = AND(HVEA1,16384)                 ; $T30 = RI-EA1-1  OFF
369     0228            AT31 = AND(HVEA1,32769)                 ; $T31 = HV-EA1-2 INTERLOCK OFF
370     0229            SA0  = AND(HVEA1,2)                     ; $T17 = HV-EA1-2 INTERLOCK ON
371             ;       SA1  = 'ZS-EA1-2D':DIN_VAL              ; HV-EA1-2 DIVERTED = 1
372             ;       SA2  = 'ZS-EA12NM':DIN_VAL              ; HV-EA1-2 TO WEIR  = 1
373     0230            SA3  = AND('HV-RC1-3':DEV_TMP1,2)       ; $T17 = HV-RC1-3 INTERLOCK ON
374             ;       SA4  = 'ZS-RC1-3D':DIN_VAL              ; HV-RC1-3 DIVERTED = 1
375             ;       SA5  = 'ZS-RC13NM':DIN_VAL              ; HV-RC1-3 TO BASIN = 1
376     0231            AT0  = AND('HV-RC1-3':DEV_TMP1,32768)   ; $T31 = HV-RC1-3 INTERLOCK OFF
377             ;
378     0232            GT15 = AND('PB2GLOBAL':DEV_TMP0,32768)  ; RC1-SAMP TO RC1-PIG "SAMPLE"
379     0233            SA7  = 'YS-P-RC1':DIN_VAL               ; PUMP P-RC1 SC SAMPL PUMP  ON
380     0234            SB0  = AND(PIG1,4)                      ; $T20 = YS-RC1-6 BYPASS ON
381     0235            SB1  = AND(PIG0,1)                      ; $T0  = BTIMER KY-RC1-1 ON
382     0236            SB2  = AND(PIG0,2)                      ; $T1  = BTIMER KY-RC1-1 TIMEOUT
383     0237            AT2  = AND(PIG0,4)
384     0238            AT3  = AND(PIG0,8)
385     0239            AT4  = AND(PIG0,16)                     ; $T4  = RC1-PIG INTERLOCK OFF
```

```
386   0240          AT8  = AND(PIG0,256)                    ; $T8  = RC1-PIG DIVERT
387   0241          AT10 = AND(PIG0,1024)                   ; $T10 = RC1-PIG FLUSH
388   0242          AT15 = AND('RC1-SAMP':DEV_TMP0,32768)   ; $T15 = RC1-SAMP FAULT
389   0243          AT17 = AND(PIG1,2)
390   0244          AT20 = AND(PIG1,16)                     ; $T20 = RC1-PIG BYPASS ON
391   0245          AT21 = AND('RC1-SAMP':DEV_TMP1,32)      ; $T21 = PULSE SAMPLER RC1-SAMP
392   0246          AT23 = AND('RC1-SAMP':DEV_TMP1,128)     ; $T23 = SAMPLE RC1-SAMP
393   0247          BT5  = AND(PIG0,1)
394   0248          BT1  = AND(PIG0,2)                      ; $T1  = BTIMER KY-RC1-1 TIMEOUT
395   0249          BT0  = XOR(BT1,BT5)                     ; $BT0 = BTIMER KY-RC1-1 ON
396   0250          BT2 = AND('RC1-PIG':DEV_STAT,8)
397   0251          BT3 = AND('RC1-PIG':DEV_STAT,16)
398   0252          BT4 = AND('RC1-PIG':DEV_STAT,32)
399   0253          BT5 = AND('RC1-PIG':DEV_STAT,64)
400   0254          BT6 = AND('RC1-PIG':DEV_STAT,128)
401   0255          BT7 = AND('HV-EA1-2':DEV_STAT,4)
402   0256          BT8 = AND('HV-EA1-2':DEV_STAT,16)
403   0257          BT9 = AND('HV-EA1-2':DEV_STAT,32)
404   0258          BT10 = AND('HV-RC1-3':DEV_STAT,4)
405   0259          BT11 = AND('HV-RC1-3':DEV_STAT,16)
406   0260          BT12 = AND('HV-RC1-3':DEV_STAT,32)
407   0261          BT13 = AND('RC1-SAMP':DEV_STAT,8)       ; SAMPLE
408   0262          BT14 = AND('RC1-SAMP':DEV_STAT,16)      ; SHUTDOWN
409   0263          BT19 = AND('RC1-SAMP':DEV_STAT,32)      ; INTERLOCK
410   0264          BT31 = AND('RC1-SAMP':DEV_TMP1,32768)   ; $T31 = RC1-SAMP INTERLOCK OFF
411   0266          IF AND('SC-ALARM':DEV_FLG,64)THEN PUTMANL 'SC-ALARM'
412   0268          IF AND('HV-RC1-3':DEV_FLG,64)THEN PUTMANL 'HV-RC1-3'
413   0270          IF AND('RC1-SAMP':DEV_FLG,64)THEN PUTMANL 'RC1-SAMP'
414   0272          IF AND('RC1-PIG':DEV_FLG,64)THEN PUTMANL 'RC1-PIG'
415   0274          IF AND('HV-EA1-2':DEV_FLG,64)THEN PUTMANL 'HV-EA1-2'
416               ;
417               ;       BEGIN LOGIC FOR GRAPHICS
418               ;
419               ;       B E G I N   F Q I C - R E S E T :
420               ;
421               ;       Function:       For FQIC loops that have a CALC block which performs the
422               ;                       integration, set the output of the CALC block to zero
423               ;                       after the RC1-SAMP device is turned off.
424               ;
425   0276          IF AND(DEVON,2) THEN SDO = 1                    ; If on (2), set device stat flag
426   0277          IF AND(DEVON,1) & AND(SDO,1) THEN               ; For on-off transition
427                    BEGIN
428   0278             SDO = 0
429   0279             PRINT ADI,"... FQIC-RC1 LAST READING =  %f GAL",CALOUT
430   0280             CALOUT = 0.0
431                    END
432               ;
433               ;       E N D   F Q I C - R E S E T
434               ;
435   0281          GOTO AGAIN
436               ;
437               ;       E N D   P R O G R A M   L O O P
438               ;
439               ;       BEGIN CLEAR, ALARM, ACKNOWLEDGE, MESSAGE, ERROR, AND TERMINATE SUBROUTINES:
440               ;
```

```
441   0282   CLR:     LET PSTAT = 0
442                   ;
443                   ; If not white, then unack:
444                   ;
445   0285            IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
446   0286            LET FCLEAR = 1
447   0287            GOSUB ACKK
448   0288            RETURN
449           ;
450   0289   ALM:     LET PSTAT = 1
451                   ;
452                   ; If not white, then unack:
453                   ;
454   0292            IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
455                   ;
456                   ; Turn audible alarm on if white or yellow:
457                   ;
458   0293            PDEV:DEV_TMP0 = 2
459   0295            IF AND(PTYPE,3)THEN FALMON = 1
460   0296            LAMP ADI,SKDKEY,2
461   0297            CRIT
462   0298            GOSUB ACKK
463   0299            RETURN.
464           ;
465   0300   ACK1:    LET PSTAT = 2                                          ; Ack the alarm
466   0301            LET FACK = 1                                           ; Set ACK flag for C program
467           ;
468           ;       ALARM TYPER STRINGS:
469           ;
470   0303   ACKK:    IF ILOOP = 0 THEN STRNG0 = "REBOILER SC RADIATION HI/FAIL"
471   0305            IF ILOOP = 1 THEN STRNG0 = "WEIR BOX LEVEL LOW"
472   0307            IF ILOOP = 2 THEN STRNG0 = "WEIR BOX OVERFLOW TRAP LVL LO"
473   0309           IF ILOOP = 3 THEN STRNG0 = "PUMP RC1 SC SAMPLE PUMP ON   "
474   0311            IF ILOOP = 4 THEN STRNG0 = "RC1 RADIATIN BYPASS SWITCH ON"
475   0313            IF ILOOP = 5 THEN STRNG0 = "SC SAMPLER FLOW LOW"
476   0315            IF ILOOP = 6 THEN STRNG0 = "SC SAMPLER RADIATION HIGH"
477   0317            IF ILOOP = 7 THEN STRNG0 = "SC RADIATION SYSTEM FAIL"
478   0319            IF ILOOP = 8 THEN STRNG0 = "WEIR BOX SC-LINE TRAP LEVEL LO"
479   0321            IF ILOOP = 20 THEN STRNG0 = "RC1-PIG LOW FLOW"
480   0323            IF ILOOP = 24 THEN STRNG0 = "START SCBATCH PGM "
481   0325            IF ILOOP = 25 THEN STRNG0 = "SCBATCH WRONG UNIT"
482   0327            IF ILOOP = 26 THEN STRNG0 = "TERMINATE SCBATCH "
483           ;
484           ;  If acking alarm, then
485           ;
486   0328   AMSG:    IF PTYPE = 1 THEN
487                     BEGIN
488                     ;
489                     ;  If white alarm, print:
490                     ;
491   0330              IF (FACK = 0) & (FCLEAR = 0) & (FSTART = 0) THEN PRINT ADI,"WHITE ALARM %p",STRNG0
492   0331              IF (FACK) THEN
493                       BEGIN
494                       ;
495                       ;  If white, print it is ack and reset the flag:
```

```
496                          ;
497    0332                  PRINT ADI,"%p ALARM ACKNOWLEDGE",STRNG0
498    0333                  LET FACK = 0
499                          END
500                        END
501            ;
502            ;  If CLEAR, print it is clear and reset flag:
503            ;
504    0334              IF (FCLEAR) THEN
505                        BEGIN
506    0335                PRINT ADI,"%p ALARM CLEAR",STRNG0
507    0336                LET FCLEAR = 0
508                        END
509            ;
510            ;  If STARTUP, print startup message:
511            ;
512    0337              IF (FSTART) THEN
513                        BEGIN
514    0338                LET FSTART = 0
515    0339                PRINT ADI,"%p",STRNG0
516                        END
517    0340              RETURN
518            ;
519            ;  If ERROR - print message, start timer, and set flag:
520            ;
521    0341    ERRSUB:     IF ERRN != ERRNUM() THEN
522                          BEGIN
523    0342                  TIMERON STIME0
524    0343                  TIMERCLR STIME0
525    0344                  LET FSN0 = 1
526    0345                  LET ERRN = ERRNUM()
527    0346                  LET ERRL = ERRLIN()
528    0347                  PRINT ADI,"ERROR # %i AT LINE %i",errn,errl
529                          END
530    0348                ERSCLR
531    0349                GOTO AGAIN
532            ;
533            ;  If FABORT = 1, then abort program:
534            ;
535            ; TERM:  LET FABORT = 0
536    0350    TERM:     LET STRNG0 = "SCBATCH TERMINATED  . "
537    0351              LET ILOOP = STAR + 2
538    0352              LET FSTART = 1
539    0353              LET STRNG0 = "TERM"
540    0354              GOSUB ACKK
541                      END

      Total Errors Detected      0     Total Warnings      0
           Sequence Program Size = 7088 Bytes, 443 Pages
                14 Forward Reference(s)
```

BASINBAT

Sequence Instruction Compiler Listing 15:33:59 24-Feb-95 Page 1

```
;       B A S I N B A T . S E Q                                  VERSION 2
;
;       Function:       Sequence program for Lamp function,
;                       message display on Graphic number 022,
;                       and printer messages.
;
;       Written By      : R. N. Kersey
;       Date            : 05-31-92
;       Facility        : 242-A Evaporator Bldg.
;       Machine         : VAX 4104-A
;       Op. System      : VAX/VMS 5.5-2AH
;       D/3 Version     : 6.3-2 BASELINE
;       Compiler        : SIC
;       Build command   : @[320,20]BASINBAT
;
;       UNIT = 17, SKDKEY = 28
;
;       Assoc. UPF:     [300,1]_PRCESS.___      PRCESS.LST
;       Assoc. RPF:     [300,1]_242A.___        242A.LST
;       SDB:            [300,1]PCMOA.SDB        PCMOASDB.CMD
;
;       POINTER   SDB VAR.      FUNCTION
;       -------   ---------     ----------------------------------------
;       PSTAT     INTn          Current EPN alarm state is an integer tied to
;                               graphic flashing string of value:
;       .         (n=0-100)             0 = previously not alarmed
;       .                               1 = previously alarmed, not ack
;       .                               2 = previously alarmed, is ack
;       .                               3 = alarm is inhibited
;       .                               4 = input is off scan
;       .                               5 = input is bad
;       .         LEAKn         Special INT variable for multiple white alarms on an EPN
;       PDIN      'EPN'         ---
;       PTYPE     TYPEn         alarm type 0 = none, 1 = white, 2 = yellow,
;                                  3 = yellow and yellow, 4 = red,
;                                  6 = yellow and red
;       PREV      FRVn          Reverse acting (not fail safe) flag (1=set)
;       PSKID     SKIDn         Used for analog alarms
;       PPASS     FPASn         Turn lamp off when alarm is ACKnowledged
;       PALARM    FALMn         Analog alarm flag
;       ---       FNING         Unused error flag
;
; MODIFICATION LOG:
;
;        30-DEC-92      Fixed white alarm print message.          M. Teats
;
;-mct1+  17-MAY-93       Added P-207A-S yellow device alarm and          M. Teats
;                        removed white analog pump amp alarms to make
;                        consistent with documentation.
;
;       09-JUN-93        Added comments and baselined for          M. Teats
;                        D/3 version 3.3.  CCN# 93JUN01-01.
;
;       15-DEC-94        Revised header to indicate the new VMS    J. Berger
;                        system. Rebaselined document. Removed
```

```
56          ;                       obsolite Select() statements.
57          ;
58          ;  END OF MOD LOG
59          ;
60          UNIT    BASINS
61                  COMMON A350
62          ;
63    0000          NOP
64    0001          LET STAR = 14                   ; Start message
65    0002          LET SKDKEY = 28          ; SKID Key
66    0003          LET UNITT = 17                  ; SEQ Unit number
67    0004          ASSIGN PDEV = 'BASIN-ALM'       ; Alarm device
68    0005          ASSIGN PTYPE = TYPE14
69    0006          LET ADI = 0                     ; Alarm destination index
70    0007          LET FSNO = 0                    ; No alarm yet
71    0008          TIMEROFF STIME0          ; Alarm timer off
72    0009          TIMERCLR STIME0          ; Alarm timer clear
73    0010          LET STRNG0 = " "                ; Blank 30 character string
74    0011          LET ERRN = 0                    ; Set error number to zero
75    0012          LET ERRL = 0                    ; Set error line to zero
76    0013          LET FTRIP4 = 0                  ; FTRIP4 = 1 when audible horn is to be turned on
77    0014          GETUNIT TRIP                    ; Which sequence unit is program running in
78    0015          IF TRIP != UNITT THEN           ; Abort if running in wrong seq. unit
79                     BEGIN
80    0016               FSTART = 1
81    0017               ILOOP = STAR + 1
82    0018               PTYPE = 0
83    0019               GOSUB ACKK
84    0020               GOTO TERM
85                       END
86          ;         IF !SELECT() THEN GOTO AGAIN
87          ;
88          ;         B E G I N   I N I T I A L I Z I N G :
89          ;
90    0021          STRNG0 = "INIT"
91    0022    INIT: LET FABORT = 0                  ; If FABORT = 1 then abort program
92    0023          LET FSTART = 0                  ; Program start flag = 0
93    0024          LAMP ADI,SKDKEY,0               ; Turn off the annunciator key
94    0025          LET LSG0 = " "                  ; Clear the string variables
95    0026          LET LSG1 = " "
96    0027          LET LSG2 = " "
97    0028          LET LSG3 = " "
98    0029          LET LSG4 = " "
99    0030          PDEV:DEV_TMP0 = 1                                 ; Set alarm device to non-alarm state
100   0031          IF AND(PDEV:DEV_ALMCT,16) THEN                    ; If alarm device is inha, remove the inha
101   0032            PDEV:DEV_ALMCT=XOR(PDEV:DEV_ALMCT,16)
102   0033          IF AND(PDEV:DEV_ALMCT,32) THEN                    ; If alarm device is ack, clear it
103   0034             PDEV:DEV_ALMCT = XOR(PDEV:DEV_ALMCT,32)
104         ;
105         ;         First, set all flags and integers:
106         ;
107   0035          ASSIGN PDIN = 'YS-SCP3IL'
108   0036          ASSIGN PSTAT = INT0
109   0037          ASSIGN PTYPE = TYPE0
110   0038          ASSIGN PREV = FRV0
```

```
111  0039          ASSIGN PPASS = FPAS0
112          ;
113  0040          FOR ILOOP = 0 TO 10
114  0041          LET PSTAT = 0
115  0042          LET PTYPE = 2
116  0043          LET PREV = 0
117  0044          LET PPASS = 0
118  0045          IF ILOOP < 10 THEN                              ; For digitals.  -mct1+
119                 BEGIN                                          ;-mct1+
120  0046           IF AND(PDIN:DIN_DFLG,3) THEN                  ; Check for alarms (must be red or yellow)
121                   BEGIN
122  0047             PTYPE = 2                                    ; Default to yellow alarm
123  0049             IF AND(PDIN:DIN_DFLG,1) THEN LET PREV = 1          ; Reverse acting
124  0051             IF AND(PDIN:DIN_DFLG,4) THEN LET PTYPE = 4         ; Set to red critical
125                   END
126  0052           STEP PDIN BY 1                          ;-mct1+
127                 END                                             ;-mct1+
128  0053          STEP PSTAT BY 1
129  0054          STEP PTYPE BY 1
130  0055          STEP PREV BY 1
131  0056          STEP PPASS BY 1
132  0057          NEXT ILOOP
133          ;
134  0058          FRV0 = 0
135  0059          FRV1 = 0
136  0060          FRV2 = 1
137  0061          FRV3 = 0
138  0062          FRV4 = 0
139  0063          FRV5 = 0
140  0064          FRV6 = 0
141  0065          FRV7 = 0
142  0066          FRV8 = 0
143  0067          FRV9 = 0
144          ;
145  0068          TYPE2 = 0
146  0069          TYPE5 = 1
147  0070          TYPE7 = 1
148  0071          TYPE9 = 1
149  0072          TYPE10 = 2                                      ;-mct1+
150          ;
151  0073          INT10 = 0                                       ;-mct1+
152          ;
153          ;     E N D   I N I T I A L I Z I N G
154          ;
155          ;     S E N D   P R O G R A M   S T A R T   M E S S A G E
156          ;
157  0074          ILOOP = STAR
158  0075          FSTART = 1
159  0076          GOSUB ACKK
160  0077          ON ERROR GOSUB ERRSUB
161          ;
162          ;     B E G I N   P R O G R A M   L O O P :
163          ;
164  0078    AGAIN:MARK STEP "WTUNTL",0,0
165          ;     WAITUNTIL SELECT() ¦ FABORT      ; Wait until micro selected or program aborted
```

```
166             ;       IF (FABORT) THEN GOTO TERM      ; Terminate program if abort flag set
167     0079            MARK STEP "RNNING",0,0
168     0080            WAIT 1
169     0081            IF (FSNO) THEN                  ; If alarm flag is set
170     0082               IF STIMEO > 60 THEN          ; If alarm flag active for 1 minute
171                           BEGIN                     ; Reset timer and turn off flag
172     0083                  TIMEROFF STIMEO
173     0084                  TIMERCLR STIMEO
174     0085                  LET FSNO = 0
175     0086                  LET ERRN = 0
176     0087                  LET ERRL = 0
177                           END
178             ;
179             ;       If BASIN-ALM $T1 is alarmed and it's ALMCT indicates alarms are acknowledged:
180             ;
181     0088            IF (PDEV:DEV_TMPO = 2) & AND(PDEV:DEV_ALMCT,32) THEN
182                        BEGIN
183     0089               PDEV:DEV_TMPO = 1            ; Put BASIN-ALM in ack state
184     0090               LET FTRIP4 = 1               ; Set for unit alarm ack
185     0091               LET ACK = 1                  ; Set for unit alarm ack
186     0092               CRIT
187     0093               LAMP AD1,SKDKEY,1            ; Turn lamp on
188                        END
189             ;
190             ;       BEGIN DINs:
191             ;
192     0094            ASSIGN PDIN = 'YS-SCP3IL'
193     0095            ASSIGN PSTAT = INTO
194     0096            ASSIGN PTYPE = TYPEO
195     0097            ASSIGN PREV = FRVO
196     0098            ASSIGN PPASS = FPASO
197     0099            LET CURENT = 0                  ; Used to set lamp status
198     0100            LET LAST = 0
199     0101            LET ANALOG = 0
200     0102            LET NXT = 0
201     0103            LMPVAR = 0
202     0104            BITVAL = 0
203             ;
204     0105    LOP:    FOR ILOOP = 0 TO 9
205     0106            IF ( PTYPE > 0 ) THEN                                   ; If some type of alarm
206                        BEGIN                                             ;be1
207     0108               IF AND(PDIN:DIN_ALMCT,16) THEN LET PSTAT = 3    ; If EPN is inhibited
208     0110               IF AND(PDIN:DIN_ALMCT,4) THEN LET PSTAT = 5             ; If EPN is bad input
209     0112               IF !AND(PDIN:DIN_FLG,8) THEN LET PSTAT = 4              ; If EPN is disabled
210     0113               IF ( PSTAT < 3 ) THEN                            ; If not inhibited
211                              BEGIN                                      ;be2
212     0114                     IF PSTAT = 1 THEN
213                                 BEGIN                                   ;be3
214     0115                        IF PTYPE != 1 THEN
215     0117                           IF AND(PDIN:DIN_ALMCT,32) THEN GOSUB ACK1
216     0119                        IF (FTRIP4) THEN GOSUB ACK1
217                                 END                                     ;en3
218     0122                     IF PSTAT = 0 THEN IF(PDIN:DIN_VAL = PREV) THEN GOSUB ALM
219     0123                     IF PSTAT THEN
220                                 BEGIN                                   ;be4
```

```
221  0125                    IF PDIN:DIN_VAL != PREV THEN GOSUB CLR
222  0126                    NXT = PSTAT
223  0128                    IF (PPASS) & (PSTAT = 2 ) THEN NXT = 0
224  0129                    LAST = OR(LAST,NXT)
225                          END                                          ;en4
226                       END                                             ;en2
227  0131              ELSE IF !AND(PDIN:DIN_ALMCT,16)&!AND(PDIN:DIN_ALMCT,4)&AND(PDIN:DIN_FLG,8)THEN LET PSTAT = 0
228                       END                                             ;en1
229  0132           STEP PDIN BY 1
230  0133           STEP PSTAT BY 1
231  0134           STEP PTYPE BY 1
232  0135           STEP PREV BY 1
233  0136           STEP PPASS BY 1
234  0137           NEXT ILOOP
235          ;
236          ;       END DINs
237          ;
238          ;+mct1-  BEGIN P-207A-S DEVICE ALARM:
239          ;
240  0138    SMP:    ILOOP = 10                                                ; P-207A-S
241  0139            ASSIGN PSTAT = INT10
242  0140            ASSIGN PTYPE = TYPE10
243          ;       IF !SELECT() THEN GOTO AGAIN
244  0141            IF (TYPE10) THEN                                          ; If some type of alarm
245                     BEGIN                                          ;be1
246  0143               IF AND('P-207A-S':DEV_ALMCT,16) THEN PSTAT = 3         ; If inhibited
247  0144               IF ( PSTAT < 3 ) THEN                                  ; If not inhibited
248                        BEGIN                                       ;be2
249  0145                  IF PSTAT = 1 THEN
250                                                                            ; If previously not alarmed
                           BEGIN                                       ;be3
251  0146                     IF PTYPE != 1 THEN                               ; If not white
252  0148                        IF AND('P-207A-S':DEV_ALMCT,32) THEN GOSUB ACK1   ; If acknowledged
253  0150                     IF (FTRIP4) THEN GOSUB ACK1                      ; If system acknowledged
254                           END                                         ;en3
255                     ;
256                     ; If not alarmed but device status is alarming:
257                     ;
258  0152               IF PSTAT = 0 THEN IF AND('P-207A-S':DEV_STAT,64) THEN
259                        BEGIN                                       ;be4
260  0153                  LET PSTAT = 1                                    ; Update the alarm status
261  0156                  IF PTYPE != 1 THEN IF AND('P-207A-S':DEV_ALMCT,32)THEN 'P-207A-S':DEV_ALMCT = XOR('P-207A-S':DEV_ALMCT,32)
262  0157                  LET PDEV:DEV_TMP0 = 2                            ; Set BASIN-ALM to alarm status
263  0159                  IF AND(PTYPE,3) THEN FALMON = 1                  ; Turn on audible horn
264  0160                  CRIT
265  0161                  LAMP ADI,SKDKEY,2                                ; Blink the SKID key
266  0162                  GOSUB ACKK
267                        END                                         ;en4
268          ;
269  0163               IF (PSTAT) THEN                                     ; If previously alarmed
270                        BEGIN                                       ;be5
271  0164                  IF !AND('P-207A-S':DEV_STAT,64) THEN             ; If device status is not alarming
272                           BEGIN                                    ;be6
273  0165                     PSTAT = 0                                     ; Update the alarm status
274  0168                     IF PTYPE != 1 THEN IF AND('P-207A-S':DEV_ALMCT,32)THEN 'P-207A-S':DEV_ALMCT = XOR('P-207A-S':DEV_ALMCT,32)
275  0169                     LET FCLEAR = 1
```

```
276  0170                        GOSUB ACKK
277                              END                                          ;en6
278                           ;
279                           ; Manage The Lamp:
280                           ;
281  0171                     NXT = PSTAT
282  0173                     IF (PPASS) & (PSTAT = 2 ) THEN NXT = 0
283  0174                     LAST = OR(LAST,NXT)
284                           END                                             ;en5
285                        END                                                ;en2
286  0176                ELSE IF !AND('P-207A-S':DEV_ALMCT,16) THEN PSTAT = 0
287                        END                                                ;en1
288          ;
289          ;-mct1+  END P-207A-S DEVICE ALARM
290          ;
291  0177          MARK STEP "BELOW",0,0
292  0178          FTRIP4 = 0
293  0179          FTRIP3 = 0
294  0181          IF LAST > 3 THEN PRINT 0," LAMP VARIABLE IS GREATER THAN 3!"
295  0182          CURENT = OR(LAST,WHITE)
296  0184          IF AND(CURENT,1)THEN LMPVAR = 2
297  0186             ELSE IF AND(CURENT,2)THEN LMPVAR = 1
298  0187                     ELSE LMPVAR = 0
299  0188          LAMP ADI,SKDKEY,LMPVAR
300  0189          MARK STEP "BOTTOM",0,0
301  0190          NOCRIT
302          ;
303          ;        BEGIN LOGIC FOR DEVICES:
304          ;
305  0191          SA0 = AND(16,'207-A-P3':DEV_STAT)                          ; SHUTDOWN
306  0192          SA1 = AND(32,'207-A-P3':DEV_STAT)                          ; INTERLOCK
307  0193          AT24 = OR('LDS-350-2':DIN_VAL,'LSH-350-2':DIN_VAL)         ;
308  0194          AT25 = AND('207-A-P3':DEV_TMP1,512)
309  0195          AT26 = AND('207-A-P3':DEV_TMP1,1024)
310  0196          AT27 = AND('207-A-P3':DEV_TMP1,2048)
311  0197          AT28 = AND('207-A-P3':DEV_TMP1,4096)
312  0198          AT30 = AND('207-A-P3':DEV_TMP1,16384)
313  0199          AT31 = AND('207-A-P3':DEV_TMP1,32768)                      ; INTERLOCK
314          ;
315  0200          IF AT28 THEN
316  0201             IF !FLG0 THEN
317                         BEGIN
318  0202                   TIMERON OSTMR
319  0203                   TIMERCLR OSTMR
320  0204                   FLG0 = 1
321                         END
322          ;
323  0205          IF FLG0 THEN
324                   BEGIN
325  0206             IF OSTMR > 60 THEN
326                        BEGIN
327  0207                  TIMEROFF OSTMR
328  0208                  TIMERCLR OSTMR
329                        END
330  0210             IF AT28 = 0 THEN FLG0 = 0
```

```
331                     END
332             ;
333             ;       BEGIN LOGIC FOR DEVICES:
334             ;
335             ;       DEVICES TO LOCK IN MANUAL:
336             ;
337     0212            IF AND('BASIN-ALM':DEV_FLG,64) THEN PUTMANL 'BASIN-ALM'
338     0214            IF AND('P-207A-S':DEV_FLG,64) THEN PUTMANL 'P-207A-S'
339     0216            IF AND('207-A-P3':DEV_FLG,64) THEN PUTMANL '207-A-P3'
340     0218            IF AND('HV-SC-1A':DEV_FLG,64) THEN PUTMANL 'HV-SC-1A'
341     0220            IF AND('HV-SC-2A':DEV_FLG,64) THEN PUTMANL 'HV-SC-2A'
342     0222            IF AND('HV-SC-3A':DEV_FLG,64) THEN PUTMANL 'HV-SC-3A'
343     0224            IF AND('HV-SC-2B':DEV_FLG,64) THEN PUTMANL 'HV-SC-2B'
344     0226            IF AND('HV-SC-1B':DEV_FLG,64) THEN PUTMANL 'HV-SC-1B'
345     0228            IF AND('HV-SC-3B':DEV_FLG,64) THEN PUTMANL 'HV-SC-3B'
346             ;
347     0229            GOTO AGAIN
348             ;
349             ;       E N D   P R O G R A M   L O O P
350             ;
351             ;       BEGIN CLEAR, ALARM, ACKNOWLEDGE, MESSAGE, ERROR, AND TERMINATE SUBROUTINES:
352             ;
353     0230    CLR:    LET PSTAT = 0
354                     ;
355                     ; If not white, then unack:
356                     ;
357     0233            IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
358     0234            LET FCLEAR = 1
359     0235            GOSUB ACKK
360     0236            RETURN
361             ;
362     0237    ALM:    LET PSTAT = 1
363                     ;
364                     ; If not white, then unack:
365                     ;
366     0240            IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
367                     ;
368                     ; Turn audible ;alarm on if white or yellow:
369                     ;
370     0241            PDEV:DEV_TMP0 = 2
371     0242            LAMP ADI,SKDKEY,2
372     0244            IF !AND(PTYPE,4) THEN FALMON = 1
373     0245            CRIT
374     0246            LET FTRIP4 = 0
375     0247            GOSUB ACKK
376     0248            RETURN
377             ;
378     0249    ACK1:   LET PSTAT = 2                                   ; Ack the alarm
379     0250            LET FACK = 1                                    ; Set ACK flag for C program
380             ;
381             ;       ALARM TYPER STRINGS:
382             ;
383     0252    ACKK:   IF ILOOP =  0 THEN STRNG0 = "PUMP P3 INTERLOCK     "
384     0254            IF ILOOP =  1 THEN STRNG0 = "207-A SUMP LEVEL HIGH "
385     0256            IF ILOOP =  3 THEN STRNG0 = "SC BASINS LEVEL HIHI  "
```

```
386   0258            IF ILOOP =  4 THEN STRNGO = "SC BASIN-1 LEVEL HIGH "
387   0260            IF ILOOP =  5 THEN STRNGO = "SC BASIN-1 LEVEL LOW  "
388   0262            IF ILOOP =  6 THEN STRNGO = "SC BASIN-2 LEVEL HIGH "
389   0264            IF ILOOP =  7 THEN STRNGO = "SC BASIN-2 LEVEL LOW  "
390   0266            IF ILOOP =  8 THEN STRNGO = "SC BASIN-3 LEVEL HIGH "
391   0268            IF ILOOP =  9 THEN STRNGO = "SC BASIN-3 LEVEL LOW  "
392   0270            IF ILOOP = 10 THEN STRNGO = "207A SUMP PMP ALARM   "
393   0272            IF ILOOP = 14 THEN STRNGO = "BASINBAT.SEQ START    "
394   0274            IF ILOOP = 15 THEN STRNGO = "BASINBAT IN WRONG UNIT"
395   0276            IF ILOOP = 16 THEN STRNGO = "BASINBAT TERMINATING  "
396         ;
397   0277  AMSG:     IF PTYPE = 1 THEN
398                     BEGIN
399                     ;
400                     ; If white alarm, print:
401                     ;
402   0279              IF (FACK = 0) & (FCLEAR = 0) & (FSTART = 0) THEN PRINT ADI,"WHITE ALARM %p",STRNGO
403   0280              IF (FACK) THEN
404                       BEGIN
405                       ;
406                       ; If white, print it is ack and reset the flag:
407                       ;
408   0281                PRINT ADI,"%p ALARM ACKNOWLEDGE",STRNGO
409   0282                LET FACK = 0
410                       END
411                   END
412         ;
413         ;  If CLEAR, print it is clear and reset flag:
414         ;
415   0283            IF (FCLEAR) THEN
416                     BEGIN
417   0284              PRINT ADI,"%p ALARM CLEAR",STRNGO
418   0285              LET FCLEAR = 0
419                     END
420         ;
421         ;  If STARTUP, print startup message:
422         ;
423   0286            IF (FSTART) THEN
424                     BEGIN
425   0287              LET FSTART = 0
426   0288              PRINT ADI,"%p",STRNGO
427                     END
428   0289            RETURN
429         ;
430         ;  If ERROR - print message, start timer, and set flag:
431         ;
432   0290  ERRSUB:   IF ERRN != ERRNUM() THEN
433                     BEGIN
434   0291              TIMERON STIMEO
435   0292              TIMERCLR STIMEO
436   0293              LET FSNO = 1
437   0294              LET ERRN = ERRNUM()
438   0295              LET ERRL = ERRLIN()
439   0296              PRINT ADI,"ERROR # %i AT LINE %i",errn,errl
440                     END
```

```
441   0297              ERSCLR
442   0298            GOTO AGAIN
443           ;
444           ;  If FABORT = 1 then abort program:
445           ;
446           ; TERM:  LET FABORT = 0
447   0299    TERM:    LET STRNGO = "BASINBAT TERMINATED    "
448   0300             LET ILOOP = STAR + 2
449   0301             LET FSTART = 1
450   0302             LET STRNGO = "TERM"
451   0303             GOSUB ACKK
452                    END

      Total Errors Detected      0     Total Warnings      0
          Sequence Program Size = 5936 Bytes, 371 Pages
                14 Forward Reference(s)
```

UTILBATCH

Sequence Instruction Compiler Listing 15:37:09 24-Feb-95 Page 1

```
;       U T I L B A T C H . S E Q                                   VERSION 2
;
;       Function:       Sequence program for Lamp function (SKDKEY 46),
;                       message display on Graphic number 021, and
;                       printer messages.  Sequence Unit 18.
;
;       Written By      : R. N. Kersey
;       Date            : 05-31-92
;       Facility        : 242-A Evaporator Bldg.
;       Machine         : VAX 4104-A
;       Op. System      : VAX/VMS 5.5-2AH
;       D/3 Version     : 6.3-2 BASELINE
;       Compiler        : SIC
;       Build command   : @[320,20]UTILBATCH
;
;       UNIT = 18, SKDKEY = 46
;
;       Assoc. UPF:     [300,1]_PRCESS.___      PRCESS.LST
;       Assoc. RPF:     [300,1]_242A.___        242A.LST
;       SDB:            [300,1]PCM0A.SDB        PCM0ASDB.CMD
;
;       POINTER   SDB VAR.      FUNCTION
;       -------   ---------     ----------------------------------------
;       PSTAT     INTn          Current EPN alarm state is an integer tied to
;                               graphic flashing string of value:
;       .         (n=0-100)             0 = previously not alarmed
;       .                               1 = previously alarmed, not ack
;       .                               2 = previously alarmed, is ack
;       .                               3 = alarm is inhibited
;       .                               4 = input is off scan
;       .                               5 = input is bad
;       .         LEAKn         Special INT variable for multiple white alarms on an EPN
;       PDIN      'EPN'         ---
;       PTYPE     TYPEn         alarm type 0 = none, 1 = white, 2 = yellow,
;                                  3 = yellow and yellow, 4 = red,
;                                  6 = yellow and red
;       PREV      FRVn          Reverse acting (not fail safe) flag (1=set)
;       PSKID     SKIDn         Used for analog alarms
;       PPASS     FPASn         Turn lamp off when alarm is ACKnowledged
;       PALARM    FALMn         Analog alarm flag
;       ---       FNING         Unused error flag
;
; MODIFICATION LOG:
;
;       11-AUG-92       - Changed PI-IA-1 white low alarm 15 to 18          M. Teats
;                       - Changed PI-PA-1 white low alarm 80 to
;                         yellow low alarm 95.  Added section 'JOP:'
;                       - Fixed FTRIP4 to prevent auto-acknowledge
;                       - Fixed print message error
;                       - Improved to cover Red alarms
;                       - Added comments
;
;       30-NOV-92       - Added 3 new UPS DINS per ECN 168693               M. Teats
;                       - Fixed SKDKEY reflashing when acknowledge
```

```
56          ;       30-DEC-92       Fixed white alarm print message.        M. Teats
57          ;
58          ;       09-JUN-93       Added comments and baselined for        M. Teats
59          ;                       D/3 version 3.3.  CCN# 93JUN01-01.
60          ;
61          ;       15-DEC-94       Revised header to indicate the new VMS          J. Berger
62          ;                       system. Rebaselined document. Remove
63          ;                       obsolite Select() statments.
64          ;
65          ;  END MOD LOG
66          ;
67          UNIT    UTIL
68          ;
69    0000          NOP
70    0001          STAR = 26                       ; Start message
71    0002          LET SKDKEY = 46         ; SKID Key
72    0003          LET UNITT = 18                  ; SEQ Unit number
73    0004          ASSIGN PDEV = 'UTILITIES'       ; Alarm device
74    0005          ASSIGN PTYPE = TYPE26
75    0006          LET ADI = 0                     ; Alarm destination index
76    0007          LET FSNO = 0                    ; No alarm yet
77    0008          TIMEROFF STIME0         ; Alarm timer off
78    0009          TIMERCLR STIME0         ; Alarm timer clear
79    0010          STRNG0 = " "                    ; Blank 30 character string
80    0011          LET ERRN = 0                    ; Set error number to zero
81    0012          LET ERRL = 0                    ; Set error line to zero
82          ;old    DLEAK0 = 0                      ; Unused flag for inhibited alarms
83    0013          FTRIP4 = 0                      ; FTRIP4 = 1 when audible horn is to be turned on
84    0014          GETUNIT TRIP                    ; Which sequence unit is program running in
85    0015          IF TRIP != UNITT THEN           ; Abort if running in wrong seq. unit
86                     BEGIN
87    0016             FSTART = 1
88    0017             ILOOP = STAR + 1
89    0018             PTYPE = 0
90    0019             GOSUB ACKK
91    0020             GOTO TERM
92                     END
93          ;       IF !SELECT() THEN GOTO AGAIN
94          ;
95          ;       B E G I N   I N I T I A L I Z I N G :
96          ;
97          ;       ILOOP 0-10 are DINs, 11-25 are AINs
98          ;
99    0021          STRNG0 = "INIT"
100   0022    INIT: LET FABORT = 0                  ; If FABORT = 1 then abort program
101   0023          LET FSTART = 0                  ; Program start flag = 0
102   0024          LAMP ADI,SKDKEY,0               ; Turn off the annunciator key
103   0025          LSG0 = " "                      ; Clear the string variables
104   0026          LSG1 = " "
105   0027          LSG2 = " "
106   0028          LSG3 = " "
107   0029          LSG4 = " "
108   0030          PDEV:DEV_TMP0 = 1                               ; Set alarm device to non-alarm state
109   0031          IF AND(PDEV:DEV_ALMCT,16) THEN                  ; If alarm device is inha, remove the inha
110   0032            PDEV:DEV_ALMCT=XOR(PDEV:DEV_ALMCT,16)
```

```
111  0033          IF AND(PDEV:DEV_ALMCT,32) THEN                  ; If alarm device is ack, clear it
112  0034             PDEV:DEV_ALMCT = XOR(PDEV:DEV_ALMCT,32)
113            ;
114            ;     First, set all flags and integers:
115            ;
116  0035          ASSIGN PDIN = 'PSLCPE1/2'
117  0036          ASSIGN PSTAT = INTO
118  0037          ASSIGN PTYPE = TYPEO
119  0038          ASSIGN PREV = FRVO
120  0039          ASSIGN PPASS = FPASO
121  0040          ASSIGN PALARM = FALMO
122  0041          ASSIGN PSKID = SKIDO
123            ;
124  0042          FOR ILOOP = 0 TO 25
125  0043          LET PSKID = 0
126  0044          LET PPASS = 0
127  0045          LET PREV = 0
128  0046          LET PTYPE = 2                                   ; Default to Yellow alarm
129  0048          IF PSTAT > 5 THEN PSTAT = 0                     ; Reset current status if undefined
130  0049          PALARM = 0                                      ; Default previous status to no alarms
131  0050          IF ILOOP <= 10 THEN                             ; For DINs
132                BEGIN
133  0051          IF AND(PDIN:DIN_DFLG,3) THEN                    ; Check for alarms (must be red or yellow)
134                   BEGIN
135  0052             LET PTYPE = 2                                ; Default to Yellow alarm
136  0054             IF AND(PDIN:DIN_DFLG,1) THEN LET PREV = 1    ; Reverse acting
137  0056             IF AND(PDIN:DIN_DFLG,4) THEN LET PTYPE = 4   ; Set to red critical alarm
138                   END
139  0058          IF ILOOP < 10 THEN STEP PDIN BY 1               ; Point to next DIN
140                END
141  0059          ELSE PTYPE = 1                                  ; For AINs, set to White alarm
142  0060    SKP:  STEP PSTAT BY 1
143  0061          STEP PTYPE BY 1
144  0062          STEP PREV BY 1
145  0063          STEP PPASS BY 1
146  0064          STEP PALARM BY 1
147  0065          STEP PSKID BY 1
148  0066          NEXT ILOOP
149            ;
150  0067          TYPE2 = 1               ; YS-CPE1-1
151  0068          TYPE3 = 1               ; YS-CPE2-1
152  0069          TYPE6 = 4               ; YS-GEN-1
153  0070          TYPE7 = 4               ; ZS-PWR-1
154  0071          TYPE8 = 4               ; JXS-PWR-3
155  0072          TYPE12 = 2              ; PI-PA-1 LO
156            ;
157            ;     For White LOLO and LO Alarms:
158            ;
159  0073          FRV15 = 1               ; PI-IA-1 LO
160  0074          FRV18 = 1               ; TI-AMBS-1 LOLO
161  0075          FRV19 = 1               ; TI-AMBS-1 LO
162  0076          FRV22 = 1               ; TI-AMBS-2 LOLO
163  0077          FRV23 = 1               ; TI-AMBS-2 LO
164            ;
165  0078          AIO =  18.0             ; PI-IA-1 LO          LEAK4 = 2+          ILOOP = INT #:
                                                                                        15
```

```
166  0079          AI1 =  25.0            ; PI-IA-1 HI          LEAK4 = 4+        16
167  0080          AI2 = 120.0            ; PI-PA-1 HI          LEAK1 = 4+        17
168  0081          AI3 =  48.0            ; TI-AMBS-1 LOLO      LEAK2 = +         18
169  0082          AI4 =  55.0            ; TI-AMBS-1 LO        LEAK2 = 2+        19
170  0083          AI5 =  80.0            ; TI-AMBS-1 HI        LEAK2 = 4+        20
171  0084          AI6 =  90.0            ; TI-AMBS-1 HIHI      LEAK2 = 6+        21
172  0085          AI7 =  48.0            ; TI-AMBS-2 LOLO      LEAK3 = +         22
173  0086          AI8 =  55.0            ; TI-AMBS-2 LO        LEAK3 = 2         23
174  0087          AI9 =  80.0            ; TI-AMBS-2 HI        LEAK3 = 4+        24
175  0088          AI10 = 90.0            ; TI-AMBS-2 HIHI      LEAK3 = 6+        25
176          ;
177          ;     E N D   I N I T I A L I Z I N G
178          ;
179          ;     S E N D   P R O G R A M   S T A R T   M E S S A G E
180          ;
181  0089          ANALOG = 0
182  0090          ILOOP = STAR
183  0091          FSTART = 1
184  0092          GOSUB ACKK
185  0093          ON ERROR GOSUB ERRSUB
186          ;
187          ;     B E G I N   P R O G R A M   L O O P :
188          ;
189  0094    AGAIN:  MARK STEP "WTUNTL",0,0
190          ;       WAITUNTIL SELECT() ¦ FABORT      ; Wait until micro is selected or program aborted
191          ;       IF (FABORT) THEN GOTO TERM       ; Terminate program if FABORT flag set
192  0095            MARK STEP "RNNING",0,0
193  0096            WAIT 1
194  0097            IF (FSN0) THEN                   ; If alarm flag is set
195  0098               IF STIME0 > 60 THEN           ; If alarm flag active for 1 minute
196                        BEGIN                     ; Reset timer and turn off flag
197  0099                  TIMEROFF STIME0
198  0100                  TIMERCLR STIME0
199  0101                  FSN0 = 0
200  0102                  ERRN = 0
201  0103                  ERRL = 0
202                        END
203          ;
204          ;     If UTILITIES $T1 is alarmed and it's ALMCT indicates alarms are acknowledged:
205          ;
206  0104            IF (PDEV:DEV_TMP0 = 2) & AND(PDEV:DEV_ALMCT,32) THEN
207                     BEGIN
208  0105               PDEV:DEV_TMP0 = 1            ; Put UTILITIES in ACKNWLDG state
209  0106               FTRIP4 = 1                   ; Set for Unit alarm ack
210  0107               ACK = 1                      ; Set for Unit alarm ack
211  0108               CRIT
212  0109               LAMP ADI,SKDKEY,1            ; Turn LAMP on
213                     END
214          ;
215          ;     BEGIN DINs:
216          ;
217  0110            ASSIGN PDIN = 'PSLCPE1/2'
218  0111            ASSIGN PTYPE = TYPE0
219  0112            ASSIGN PREV = FRV0
220  0113            ASSIGN PSTAT = INT0
```

```
221   0114          ASSIGN PPASS = FPAS0
222   0115          ASSIGN PSKID = SKID0
223   0116          LET CURENT = 0                  ; Used to set lamp status
224   0117          LET LAST = 0
225   0118          LET ANALOG = 0
226   0119          LET NXT = 0
227   0120          LMPVAR = 0
228   0121          BITVAL = 0
229         ;
230   0122  LOP:    FOR ILOOP = 0 TO 10
231         ;       IF !SELECT() THEN GOTO AGAIN
232   0123          IF ( PTYPE > 0 ) THEN                                   ; If some type of alarm
233                   BEGIN                                               ;be1
234   0125            IF AND(PDIN:DIN_ALMCT,16) THEN LET PSTAT = 3        ; If EPN inhibited
235   0127            IF AND(PDIN:DIN_ALMCT,4) THEN LET PSTAT = 5                 ; If EPN is bad input
236   0129            IF !AND(PDIN:DIN_FLG,8) THEN LET PSTAT = 4                  ; If EPN is disabled
237   0130            IF (PSTAT < 3) THEN                                         ; If Not Inhibited
238                       BEGIN                                           ;be2
239   0131                IF (PSTAT = 1) THEN
240                         BEGIN                                         ;be3
241   0132                  IF (PTYPE != 1) THEN
242   0134                    IF AND(PDIN:DIN_ALMCT,32) THEN GOSUB ACK1
243   0136                  IF (FTRIP4) THEN GOSUB ACK1
244                         END                                           ;en3
245   0139                IF PSTAT = 0 THEN IF(PDIN:DIN_VAL = PREV) THEN GOSUB ALM
246   0140                IF (PSTAT) THEN
247                           BEGIN                                       ;be4
248   0142                    IF (PDIN:DIN_VAL != PREV) THEN GOSUB CLR
249   0143                    NXT = PSTAT
250   0145                    IF (PPASS) & (PSTAT = 2) THEN NXT = 0
251   0146                    LAST = OR(LAST,NXT)
252                           END                                         ;en4
253                     END                                               ;en2
254   0148            ELSE IF !AND(PDIN:DIN_ALMCT,16)&!AND(PDIN:DIN_ALMCT,4)&AND(PDIN:DIN_FLG,8)THEN LET PSTAT = 0
255                   END                                                 ;en1
256   0149          STEP PDIN BY 1
257   0150          STEP PTYPE BY 1
258   0151          STEP PREV BY 1
259   0152          STEP PSTAT BY 1
260   0153          STEP PPASS BY 1
261   0154          STEP PSKID BY 1
262   0155          NEXT ILOOP
263         ;
264         ;       END DINs
265         ;
266         ;       BEGIN ANALOG RED AND YELLOWS:
267         ;
268   0156          ASSIGN PAI = 'PI-IA-1'
269   0157          ASSIGN PSKID = SKID11
270   0158          ASSIGN PREV = FRV11
271   0159          ASSIGN PSTAT = INT11
272   0160          ASSIGN PTYPE = TYPE11
273         ;
274   0161  JOP:    FOR JLOOP = 11 TO 14
275         ;       IF !SELECT() THEN GOTO AGAIN
```

```
276   0162            IF (PTYPE) THEN                                  ; If some alarm
277                      BEGIN                                ;be1
278   0163               BITVAL = PAI:AI_ALMCT
279   0164               IF !AND(BITVAL,16) THEN                        ; If not inhibited
280                         BEGIN                             ;be2
281   0165                  IF AND(BITVAL,15) THEN                      ; If alarm exists and is current
282                            BEGIN                          ;be3
283   0167                     IF AND(BITVAL,32) THEN NXT = 2           ; If alarm is ack
284                            ELSE                                     ; If alarm is unack
285                               BEGIN                       ;be4
286   0168                        IF PSKID !=BITVAL THEN
287                                    BEGIN                  ;be5
288   0170                             IF BITVAL < 3 THEN FALMON = 1
289   0171                             LET PSKID = BITVAL
290   0172                             PDEV:DEV_TMPO = 2
291   0173                             NXT = 1
292                                    END                    ;en5
293   0174                        ELSE NXT = 1
294                               END                         ;en4
295                            END                            ;en3
296                         ELSE                                        ; Else there is no alarm
297                            BEGIN
298   0175                     NXT = 0
299   0176                     LET PSKID = 0
300                            END
301                         END                               ;en2
302                      ELSE                                           ; Else alarm is inhibited
303                         BEGIN
304   0177                  NXT = 0
305   0178                  LET PSKID = 0
306                         END
307   0179                ANALOG = OR(ANALOG,NXT)
308                      END                                  ;en1
309   0180            STEP PSKID BY 1
310   0181            STEP PAI BY 1
311   0182            STEP PREV BY 1
312   0183            STEP PSTAT BY 1
313   0184            STEP PTYPE BY 1
314   0185            NEXT JLOOP
315             ;
316             ;      END ANALOG RED AND YELLOWS
317             ;
318             ;      BEGIN ANALOG WHITE ALARMS:
319             ;
320   0186      WHT:  WHITE = 0
321   0187            ASSIGN PALARM = AI0
322   0188            ASSIGN PSTAT = INT15
323   0189            ASSIGN PREV = FRV15
324   0190            ASSIGN PPASS = FPAS15
325   0191            ASSIGN PTYPE = TYPE15
326   0192            ILOOP = 15
327   0193            ASSIGN PAI = 'PI-IA-1'
328   0194            GOSUB ANLG
329   0195            GOSUB ANLG
330   0196            ASSIGN PAI = 'PI-PA-1'
```

```
331  0197            GOSUB ANLG
332  0198            ASSIGN PAI = 'TI-AMBS-1'
333  0199            GOSUB ANLG
334  0200            GOSUB ANLG
335  0201            GOSUB ANLG
336  0202            GOSUB ANLG
337  0203            ASSIGN PAI = 'TI-AMBS-2'
338  0204            GOSUB ANLG
339  0205            GOSUB ANLG
340  0206            GOSUB ANLG
341  0207            GOSUB ANLG
342          ;
343          ;       END ANALOG WHITE ALARMS
344          ;
345  0208            MARK STEP "BELOW",0,0
346  0209            SORT = 0
347  0210            FTRIP4 = 0
348  0212            IF LAST > 3 THEN PRINT ADI," LAMP VARIABLE IS GREATER THAN 3!"
349  0213            CURENT = OR(LAST,WHITE)
350  0214            CURENT = OR(CURENT,ANALOG)
351  0216            IF AND(CURENT,1) THEN LMPVAR = 2
352  0218               ELSE IF AND(CURENT,2) THEN LMPVAR = 1
353  0219                       ELSE LMPVAR = 0
354  0220            LAMP ADI,SKDKEY,LMPVAR
355  0221            MARK STEP "BOTTOM",0,0
356  0222            NOCRIT
357  0224            IF AND(UTILITIES:DEV_FLG,64) THEN PUTMANL UTILITIES
358  0225            GOTO AGAIN
359          ;
360          ;       E N D   P R O G R A M   L O O P
361          ;
362          ;       BEGIN ANALOG WHITE SUBROUTINE:
363          ;
364          ; ANLG:  IF !SELECT() THEN GOTO AGAIN
365  0226    ANLG:   IF (PTYPE = 1) THEN                          ; If some alarm
366                  BEGIN                                ;be1
367  0227            BITVAL = PAI:AI_ALMCT
368          ;old    IF AND(BITVAL,16) THEN                       ; If inhibited
369          ;old       BEGIN
370          ;old       BLET PSTAT = 3
371          ;old       DLEAK0 = 1
372          ;old       GOSUB ACKK
373          ;old       END
374  0228            IF !(AND(BITVAL,16)) THEN                    ; If not inhibited
375                     BEGIN                             ;be2
376  0230               IF PSTAT = 3 THEN LET PSTAT = 0
377  0231               IF (PREV) THEN                            ; If rev acting, LOLO, or LO
378                        BEGIN                          ;be3
379  0232                  IF (PSTAT = 0) THEN                    ; If not previously alarmed
380                           BEGIN                       ;be4
381  0234                     IF (PAI:AI_INVL <= PALARM) THEN GOSUB ALM
382                           END                         ;en4
383                        ELSE
384                           BEGIN                       ;be5
385  0235                     IF (PSTAT = 1) THEN                 ; If previously alarmed
```

```
386                             BEGIN                                   ;be6
387     0237                    IF (FTRIP4) THEN GOSUB ACK1             ; If ACK flag set, then ack it
388     0240                    IF PTYPE !=1 THEN IF AND(PAI:AI_ALMCT,32) THEN GOSUB ACK1
389                             END                                     ;en6
390     0242                  IF (PAI:AI_INVL >= PALARM + PAI:AI_DEAD) THEN GOSUB CLR
391                           END                                       ;en5
392                         END                                         ;en3
393                       ELSE                                                ; If fail-safe, HIHI, or HI
394                         BEGIN                                       ;be10
395     0243                IF (PSTAT = 0) THEN
396                           BEGIN                                     ;be11
397     0245                  IF (PAI:AI_INVL >= PALARM) THEN GOSUB ALM
398                           END                                       ;en11
399                         ELSE
400                           BEGIN                                     ;be12
401     0246                  IF (PSTAT = 1) THEN
402                             BEGIN                                   ;be13
403     0248                    IF (FTRIP4) THEN GOSUB ACK1
404     0251                    IF PTYPE != 1 THEN IF AND(PAI:AI_ALMCT,32) THEN GOSUB ACK1
405                             END                                     ;en13
406     0253                  IF (PAI:AI_INVL <= PALARM - PAI:AI_DEAD) THEN GOSUB CLR
407                           END                                       ;en12
408                         END                                         ;en10
409     0254                NXT = PSTAT
410     0256                IF (PPASS) & (PSTAT = 2) THEN NXT = 0
411                         END                                         ;en2
412                       ELSE
413                         BEGIN
414     0257                LET PSTAT = 3
415     0258                NXT = 0
416                         END
417     0259                WHITE = OR(WHITE,NXT)
418                       END                                           ;en1
419     0260              ELSE LET PSTAT = 0
420     0261              STEP PALARM BY 1
421     0262              STEP PSTAT BY 1
422     0263              STEP PREV BY 1
423     0264              STEP PPASS BY 1
424     0265              STEP PTYPE BY 1
425     0266              ILOOP = ILOOP + 1
426     0267              RETURN
427             ;
428             ;         END ALALOG WHITE SUBROUTINE
429             ;
430             ;         BEGIN CLEAR, ALARM, ACKNOWLEDGE, MESSAGE, ERROR, AND TERMINATE SUBROUTINES:
431             ;
432     0268    CLR:      LET PSTAT = 0
433                       ;
434                       ;  If not white, then unack:
435                       ;
436     0271              IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
437     0272              LET FCLEAR = 1
438                       ;
439                       ;  To Set-Up For LOLO to LO or HIHI to HI Transition:
440                       ;
```

```
441   0274          IF ILOOP = 18 THEN LET INT19 = 0          ; If TI-AMBS-1 LOLO, Reset LO
442   0276          IF ILOOP = 21 THEN LET INT20 = 0          ; If TI-AMBS-1 HIHI, Reset HI
443   0278          IF ILOOP = 22 THEN LET INT23 = 0          ; If TI-AMBS-2 LOLO, Reset LO
444   0280          IF ILOOP = 25 THEN LET INT24 = 0          ; If TI-AMBS-2 HIHI, Reset HI
445   0281          GOSUB ACKK
446   0282          RETURN
447         ;
448   0283  ALM:    LET PSTAT = 1
449               ;
450               ;  If not white, then unack:
451               ;
452   0286          IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
453               ;
454               ;  Turn audible alarm on if white or yellow:
455               ;
456   0287          FALMON = 1
457   0289          IF AND(PTYPE,3) THEN FACK = 1
458   0290          CRIT
459   0291          PDEV:DEV_TMP0 = 2
460   0292          GOSUB ACKK
461   0293          RETURN
462         ;
463   0294  ACK1:   LET PSTAT = 2                                    ; Ack the alarm
464   0295          FACK = 1                                          ; Set ACK flag for C program
465         ;
466         ;               Begin Multiple AI White Alarms on One EPN:
467         ;                       PI-IA-1:
468   0297  ACKK:   IF(INT15 & INT15 < 3)THEN LET LEAK4 = INT15 + 2   ; LO
469   0299           ELSE IF(INT16 & INT16 < 3)THEN LET LEAK4 = INT16 + 4        ; HI
470   0301            ELSE IF INT15 = 3 THEN LET LEAK4 = 9
471   0302             ELSE LET LEAK4 = 0
472               ;
473               ;                       PI-PA-1:
474   0304           IF(INT17 & INT17 < 3)THEN LET LEAK1 = INT17 + 4   ; HI
475   0306            ELSE IF INT17 = 3 THEN LET LEAK1 = 9
476   0307             ELSE LET LEAK1 = 0
477               ;
478               ;                       TI-AMBS-1:
479   0309          IF(INT18 & INT18 < 3)THEN LET LEAK2 = INT18                 ; LOLO
480   0311           ELSE IF(INT19 & INT19 < 3)THEN LET LEAK2 = INT19 + 2       ; LO
481   0313            ELSE IF(INT21 & INT21 < 3)THEN LET LEAK2 = INT21 + 6      ; HIHI
482   0315             ELSE IF(INT20 & INT20 < 3)THEN LET LEAK2 = INT20 + 4     ; HI
483   0317              ELSE IF INT18 = 3 THEN LET LEAK2 = 9
484   0318               ELSE LET LEAK2 = 0
485               ;
486               ;                       TI-AMBS-2:
487   0320          IF(INT22 & INT22 < 3) THEN LET LEAK3 = INT22              ; LOLO
488   0322           ELSE IF (INT23 & INT23 < 3) THEN LET LEAK3 = INT23 + 2   ; LO
489   0324            ELSE IF(INT25 & INT25 < 3) THEN LET LEAK3 = INT25 + 6   ; HIHI
490   0326             ELSE IF( INT24 & INT24 < 3) THEN LET LEAK3 = INT24 + 4 ; HI
491   0328              ELSE IF INT22 = 3 THEN LET LEAK3 = 9
492   0329               ELSE LET LEAK3 = 0
493         ;
494         ;                   End Multiple AI White Alarms
495         ;
```

```
496           ;old     IF DLEAK0 = 1 THEN                                        ; If Inhibited
497           ;old              BEGIN
498           ;old              BLET DLEAK0 = 0
499           ;old              RETURN                                           ; Skip message routine
500           ;old              END
501           ;
502           ;         ALARM TYPER STRINGS:
503           ;
504    0331             IF ILOOP = 0 THEN STRNG0 = "PRCESS AIR PRSS LOW  "              ; PSLCPE1/2
505    0333             IF ILOOP = 1 THEN STRNG0 = "INSTUMENT AIR PRES LO"              ; PSL-IA-2
506    0335             IF ILOOP = 2 THEN STRNG0 = "AIR COMPRESSOR 1 OFF "              ; YS-CPE1-1
507    0337             IF ILOOP = 3 THEN STRNG0 = "AIR COMPRESSOR 2 OFF "              ; YS-CPE2-1
508    0339             IF ILOOP = 4 THEN STRNG0 = "24 VDC POWER FAIL    "              ; JXS-PWR-1
509    0341             IF ILOOP = 5 THEN STRNG0 = "DIESEL GENRATR TROUBL"              ; YS-DIEGEN
510    0343             IF ILOOP = 6 THEN STRNG0 = "EMRGNCY DIESELGEN FAIL"             ; YS-GEN-1
511    0345             IF ILOOP = 7 THEN STRNG0 = "EMERG PWR XFR SWITCH ON"            ; ZS-PWR-1
512    0347             IF ILOOP = 8 THEN STRNG0 = "UPS AC INPUT FAIL"                  ; JXS-PWR-3
513    0349             IF ILOOP = 9 THEN STRNG0 = "UPS BATTERY LOW"                    ; YSL-UPS-1
514    0351             IF ILOOP = 10 THEN STRNG0 = "UPS MBS BYPASS POSITION"           ; ZS-UPS-1
515    0353             IF ILOOP = 15 THEN STRNG0 = "INSTRUMENT AIR HDR PRESS LOW"      ; PI-IA-1
516    0355             IF ILOOP = 16 THEN STRNG0 = "INSTRUMENT AIR HDR PRESS HI" ; PI-IA-1
517    0357             IF ILOOP = 17 THEN STRNG0 = "PROCESS AIR HDR PRESS HIGH " ; PI-PA-1
518    0359             IF ILOOP = 18 THEN STRNG0 = "COLD JUNCTION SENSOR 1 LOLO" ; TI-AMBS-1
519    0361             IF ILOOP = 19 THEN STRNG0 = "COLD JUNCTION SENSOR 1 LO  " ; TI-AMBS-1
520    0363             IF ILOOP = 20 THEN STRNG0 = "COLD JUNCTION SENSOR 1 HI  " ; TI-AMBS-1
521    0365             IF ILOOP = 21 THEN STRNG0 = "COLD JUNCTION SENSOR 1 HIHI" ; TI-AMBS-1
522    0367             IF ILOOP = 22 THEN STRNG0 = "COLD JUNCTION SENSOR 2 LOLO" ; TI-AMBS-2
523    0369             IF ILOOP = 23 THEN STRNG0 = "COLD JUNCTION SENSOR 2 LO  " ; TI-AMBS-2
524    0371             IF ILOOP = 24 THEN STRNG0 = "COLD JUNCTION SENSOR 2 HI  " ; TI-AMBS-2
525    0373             IF ILOOP = 25 THEN STRNG0 = "COLD JUNCTION SENSOR 2 HIHI" ; TI-AMBS-2
526    0375             IF ILOOP = 26 THEN STRNG0 = "UTILBATCH.SEQ START        " ; STAR
527    0377             IF ILOOP = 27 THEN STRNG0 = "UTILBATCH IN WRONG UNIT    "
528    0379             IF ILOOP = 28 THEN STRNG0 = "UTILBATCH TERMINATING      "
529           ;
530    0380   AMSG:     IF PTYPE = 1 THEN
531                        BEGIN
532                        ;
533                        ;  If white alarm, print:
534                        ;
535    0382                IF (FACK = 0) & (FCLEAR = 0) & (FSTART = 0) THEN PRINT ADI,"WHITE ALARM %p",STRNG0
536    0383                IF(FACK) THEN
537                           BEGIN
538                           ;
539                           ;  If white, print it is ack and reset the flag:
540                           ;
541    0384                   PRINT ADI,"%p ALARM ACKNOWLEDGE",STRNG0
542    0385                   LET FACK = 0
543                           END
544                        END
545           ;
546           ;  If CLEAR, print it is clear and reset flag:
547           ;
548    0386            IF (FCLEAR) THEN
549                       BEGIN
550    0387               PRINT ADI,"%p ALARM CLEAR",STRNG0
```

```
551    0388              LET FCLEAR = 0
552                      END
553           ;
554           ;  If STARTUP, print startup message:
555           ;
556    0389            IF (FSTART) THEN
557                      BEGIN
558    0390              LET FSTART = 0
559    0391              PRINT ADI,"%p",STRNGO
560                      END
561    0392            RETURN
562           ;
563           ;  If ERROR - print message, start timer, and set flag:
564           ;
565    0393    ERRSUB:   IF ERRN != ERRNUM() THEN
566                      BEGIN
567    0394              TIMERON STIMEO
568    0395              TIMERCLR STIMEO
569    0396              LET FSNO = 1
570    0397              LET ERRN = ERRNUM()
571    0398              LET ERRL = ERRLIN()
572    0399              PRINT ADI,"ERROR # %i AT LINE %i",errn,errl
573                      END
574    0400            ERSCLR
575    0401            GOTO AGAIN
576           ;
577           ;  If FABORT = 1, then abort program:
578           ;
579           ; TERM:  LET FABORT = 0
580    0402    TERM:   LET STRNGO = "UTILBATCH TERMINATED    "
581    0403            LET ILOOP = STAR + 2
582    0404            LET FSTART = 1
583    0405            LET STRNGO = "TERM"
584    0406            GOSUB ACKK
585                    END

      Total Errors Detected      0     Total Warnings      0
          Sequence Program Size = 8032 Bytes, 502 Pages
               29 Forward Reference(s)
```

AMUBATCH

Sequence Instruction Compiler Listing 15:32:28 24-Feb-95 Page 1

```
1           ;       A M U B A T C H . S E Q                                 VERSION 2
2           ;
3           ;       Function:       Sequence program for Lamp function,
4           ;                       message display on Graphic number 019,
5           ;                       and printer messages.
6           ;
7           ;       Written By      : R. N. Kersey
8           ;       Date            : 05-31-92
9           ;       Facility        : 242-A Evaporator Bldg.
10          ;       Machine         : VAX 4104-A
11          ;       Op. System      : VAX/VMS 5.5-2AH
12          ;       D/3 Version     : 6.3-2 BASELINE
13          ;       Compiler        : SIC
14          ;       Build command   : @[320,20]AMUBATCH
15          ;
16          ;       UNIT = 20, SKDKEY = 45
17          ;
18          ;       Assoc. UPF:     [300,1]_PRCESS.___      PRCESS.LST
19          ;       Assoc. RPF:     [300,1]_242A.___        242A.LST
20          ;       SDB:            [300,1]PCM0A.SDB        PCM0ASDB.CMD
21          ;
22          ;       POINTER   SDB VAR.      FUNCTION
23          ;       -------   --------      ----------------------------------------
24          ;       PSTAT     INTn          Current EPN alarm state is an integer tied to
25          ;                               graphic flashing string of value:
26          ;       .         (n=0-100)             0 = previously not alarmed
27          ;       .                               1 = previously alarmed, not ack
28          ;       .                               2 = previously alarmed, is ack
29          ;       .                               3 = alarm is inhibited
30          ;       .                               4 = input is off scan
31          ;       .                               5 = input is bad
32          ;       .         LEAKn         Special INT variable for multiple white alarms on an EPN
33          ;       PDIN      'EPN'         ---
34          ;       PTYPE     TYPEn         alarm type 0 = none, 1 = white, 2 = yellow,
35          ;                                  3 = yellow and yellow, 4 = red,
36          ;                                  6 = yellow and red
37          ;       PREV      FRVn          Reverse acting (not fail safe) flag (1=set)
38          ;       PSKID     SKIDn         Used for analog alarms
39          ;       PPASS     FPASn         Turn lamp off when alarm is ACKnowledged
40          ;       PALARM    FALMn         Analog alarm flag
41          ;       ---       FNING         Unused error flag
42          ;
43          ; MODIFICATION LOG:
44          ;
45          ;       30-DEC-92       Fixed white alarm print message.        M. Teats
46          ;
47          ;       08-JUN-93       Added comments and baselined for        R. Persin
48          ;                       D/3 version 3.3.  CCN# 93JUN01-01.
49          ;
50          ;       15-DEC-94       Revised header to indicate the new VMS          J. Berger
51          ;                       system. Rebaselined document. Removed
52          ;                       obsolite Select() statment.
53          ;
54          ; END MOD LOG
55          ;
```

```
 56           UNIT     AMU
 57           ;
 58    0000            NOP
 59    0001            STAR = 33
 60    0002            ASSIGN PDEV = 'AMU-ALARM'                    ; Alarm device
 61           ;
 62    0003            LET FSNO = 0                                 ; Error message repeater
 63    0004            TIMEROFF STIMEO                            ; Alarm timer off
 64    0005            TIMERCLR STIMEO                            ; Alarm timer clear
 65    0006            LET STRNGO = " "                             ; Blank 30 character string
 66    0007            LET ERRN = 0                                 ; Set erron number to zero
 67    0008            LET ERRL = 0                                 ; Set error linbe to zero
 68           ;
 69    0009            LET FTRIP4 = 0                               ; FTRIP = 1 when audible horn is to be turned on
 70    0010            LET DLEAK3 = 0                               ; Flag for inhibited alarms
 71           ;
 72           ;                S E Q U E N C E   U N I T   A N D   P C M   C H E C K
 73           ;
 74    0011            GETUNIT TRIP                                 ; Which sequence unit is program running in
 75    0012            IF TRIP != 20 THEN                           ; Abort if runni9ng in wring seq. unit
 76                       BEGIN
 77    0013               LET FSTART = 1
 78    0014               LET ILOOP = STAR + 1
 79    0015               GOSUB ACKK
 80    0016               LET FSTART = 1
 81    0017               PRINT ADI," >>>>>  ATTEMPT TO RUN AMUBATCH IN WRONG UNIT"
 82    0018               GOTO TERM
 83                       END
 84           ;        IF !SELECT() THEN GOTO AGAIN
 85           ;
 86           ;             E N D   S E Q U E N C E   U N I T   A N D   P C M   C H E C K
 87           ;
 88           ;              B E G I N   I N I T I A L I Z A T I O N :
 89           ;
 90    0019   INIT:    LET SKDKEY = 45                            ; SKID Key
 91    0020            LET ADI = 0                                  ; Alarm destination index
 92    0021            LET FABORT = 0                               ; If FABORT = 1 then abort progream
 93    0022            LET FSTART = 0                               ; Program start flag = 0
 94    0023            LAMP ADI,SKDKEY,0                            ; Turn off the annunciator key
 95    0024            LET PDEV:DEV_TMPO = 1                        ; Set alarm device to non-alarm state
 96    0025            IF AND(PDEV:DEV_ALMCT,16) THEN               ; If alarm device is inhibited , remove the inhibit
 97    0026              LET PDEV:DEV_ALMCT=XOR(PDEV:DEV_ALMCT,16)
 98    0027            IF AND(PDEV:DEV_ALMCT,32) THEN               ; If alarm device is ack, clear it
 99    0028               LET PDEV:DEV_ALMCT = XOR(PDEV:DEV_ALMCT,32)
100           ;
101           ;               First, set all flags and integers
102           ;
103    0029            ASSIGN PDIN = 'YS-A-E101'                    ; Assign pointer to first DIN
104    0030            ASSIGN PPASS = FPASO
105    0031            ASSIGN PREV = FRVO                           ; Reverse acting flag
106    0032            ASSIGN PSTAT = INTO
107    0033            ASSIGN PTYPE = TYPEO
108    0034            ASSIGN PSKID = SKIDO                         ; stePer
109           ;
110    0035            FOR ILOOP = 0 TO 25
```

```
111  0036              LET PPASS = 0
112  0037              LET PREV = 0
113  0039              IF ILOOP < 6 THEN LET PREV = 1          ; For white DINs
114  0041              IF PSTAT > 5 THEN PSTAT = 0                     ; Reset current status if undefined
115  0042              LET PSKID = 0                           ; Default previous status to no alarm
116  0043              LET PTYPE = 0                           ; Default to no alarm
117  0044              IF ILOOP < 6 THEN
118                     BEGIN
119  0045               LET PTYPE = 2                                  ; Default to yellow alarm
120  0046               IF AND(PDIN:DIN_DFLG,3) THEN                   ; Check for alarms if defined (must be a red or yellow alarm)
121                      BEGIN
122  0047                LET PTYPE = 2                                         ; Default to yellow alarm
123  0049                IF AND(PDIN:DIN_DFLG,1) THEN LET PREV = 1     ; Reverse acting
124  0051                IF AND(PDIN:DIN_DFLG,4) THEN LET PTYPE = 4    ; Set to critical red alarm
125                      END
126                    END
127  0052.             ELSE PTYPE = 6                                 ; Red and Yellow, future use
128  0054  SKP:         IF ILOOP < 6 THEN STEP PDIN BY 1              ; Point to next DIN
129  0055              STEP PPASS BY 1
130  0056              STEP PREV BY 1
131  0057              STEP PSTAT BY 1
132  0058              STEP PSKID BY 1
133  0059              STEP PTYPE BY 1
134  0060            NEXT ILOOP
135            ;
136            ;                 Set alarm type if not red or yellow
137            ;
138  0061            LET TYPE0 = 1                                   ; YS-A-E101
139  0062            LET TYPE1 = 1                                   ; YS-P-E101
140  0063            LET TYPE2 = 1                                   ; YS-A-E102
141  0064            LET TYPE3 = 1                                   ; YS-P-E102
142  0065            LET TYPE4 = 1                                   ; YS-A-E104
143  0066            LET TYPE5 = 1                                   ; YS-P-E104
144  0067            LET TYPE6 = 1                                   ; TI-E101-2 Hi
145  0068            LET TYPE8 = 1                                   ; WFI-E101 Lo-Lo
146  0069            LET TYPE9 = 1                                   ; WFI-E101 Lo
147  0070            LET TYPE10 = 0                                  ; DI-E101-2
148  0071            LET TYPE12 = 0                                  ; FI-E102-1
149  0072            LET TYPE14 = 1                                  ; WFI-E102 Lo-Lo
150  0073            LET TYPE15 = 1                                  ; WFI-E102 Lo
151  0074            LET TYPE16 = 0                                  ; FQI-E102
152  0075            LET TYPE17 = 1                                  ; TI-E104-2 Hi
153  0076            LET TYPE19 = 1                                  ; WFI-E104 Lo-Lo
154  0077            LET TYPE20 = 1                                  ; WFI-E104 Lo
155  0078            LET TYPE21 = 0                                  ; DI-E104-2
156  0079            LET TYPE23 = 0                                  ; FQI-E104
157  0080            LET TYPE24 = 0                                  ; FI-E104-2
158  0081            LET TYPE25 = 0                                  ; FIC-E102
159            ;
160  0082            LET FRV8 = 1                                    ; WFI-E101 Lo-Lo
161  0083            LET FRV9 = 1                                    ; WFI-E101 Lo
162  0084            LET FRV14 = 1                                   ; WFI-E102 Lo-Lo
163. 0085            LET FRV15 = 1                                   ; WFI-E102 Lo
164  0086            LET FRV19 = 1                                   ; WFI-E104 Lo-Lo
165  0087            LET FRV20 = 1                                   ; WFI-E104 Lo
```

```
166             ;
167             ;               Analog white alarm limits
168             ;
169     0088            LET AI0 = 150                           ; TI-E101-2 Hi
170     0089            LET AI2 = 15                            ; WFI-E101 Lo-Lo
171     0090            LET AI3 = 25                            ; WFI-E101 Lo
172     0091            LET AI8 = 5                             ; WFI-E102 Lo-Lo
173     0092            LET AI9 = 12                            ; WFI-E102 Lo
174     0093            LET AI11 = 170                          ; TI-E104-2 Hi
175     0094            LET AI13 = 5                            ; WFI-E104 Lo-Lo
176     0095            LET AI14 = 12                           ; WFI-E104 Lo
177             ;
178             ;               E N D   I N I T I A L I Z I N G
179             ;
180             ;
181             ;       S E N D   P R O G R A M   S T A R T   M E S S A G E
182             ;
183             ;
184     0096            LET ANALOG = 0
185     0097            LET ILOOP = STAR
186     0098            LET FSTART = 1
187     0099            GOSUB ACKK
188             ;
189             ;
190             ;       B E G I N   P R O G R A M   L O O P
191             ;
192             ;
193     0100    AGAIN:  MARK STEP "AGAIN",0,0
194     0101            ON ERROR GOSUB ERRSUB
195             ;       IF (FABORT) THEN GOTO TERM              ; Terminate program if abort flag is set
196             ;       WAITUNTIL SELECT()
197     0102            WAIT 1
198             ;
199     0103            IF (FSNO) THEN                          ; If alarm flag is set
200     0104               IF STIMEO > 60 THEN                  ; If alarm flag active for 1 minute
201                           BEGIN                             ; Reset timer and turn off flag
202     0105                  TIMEROFF STIMEO
203     0106                  TIMERCLR STIMEO
204     0107                  LET FSNO = 0
205     0108                  LET ERRN = 0
206     0109                  LET ERRL = 0
207                           END
208             ;
209             ;
210             ;       If AMU ALARM $T1 is alarmed and it's ALMCT indicates alarms are acknowledged:
211             ;
212             ;
213     0110            IF (PDEV:DEV_TMPO = 2) & AND(PDEV:DEV_ALMCT,32) THEN
214                        BEGIN
215     0111               PDEV:DEV_TMPO = 1                    ; Put AMU - ALARM is ack state
216     0112               LET FTRIP4 = 1                       ; Set for unit alarm ack, system to graphic
217     0113               LET ACK = 1                          ; Set for unit alarm ack, graphic to system
218     0114               CRIT
219                        END
220             ;
```

```
221             ;
222             ;               B E G I N   D I G I T A L S
223             ;
224             ;               Function:       Recognize digital red, yellow, white alarms
225             ;                               and associate with graphic string.  Turn
226             ;                               on alarm ACK device, turn on Lamp, and
227             ;                               flash message on graphic.
228             ;
229     0115            ASSIGN PDIN = 'YS-A-E101'
230     0116            ASSIGN PTYPE = TYPE0                    ; ALARM TYPE
231     0117            ASSIGN PREV = FRV0                      ; REVERSE ACTING FLAG
232     0118            ASSIGN PSTAT = INT0                     ; INTEGERS TIED TO GRAPHIC FLASHING STRINGS
233     0119            ASSIGN PPASS = FPAS0
234     0120            LET LAST = 0
235     0121            LET NXT = 0
236     0122            LMPVAR = 0
237     0123            BITVAL = 0
238             ;
239     0124    LOP:    FOR ILOOP = 0 TO 5
240             ;       IF !SELECT() THEN GOTO AGAIN
241     0125            IF PTYPE > 0 THEN                               ; If some type of alarm
242                        BEGIN                                                ; b1
243     0127               IF AND(PDIN:DIN_ALMCT,16) THEN LET PSTAT = 3
244     0129               IF AND(PDIN:DIN_ALMCT,4) THEN LET PSTAT = 5
245     0131               IF !AND(PDIN:DIN_FLG,8) THEN LET PSTAT = 4
246     0132               IF PSTAT < 3 THEN                            ; If not inhibited
247                            BEGIN                                            ; b2
248     0133                   IF PSTAT = 1 THEN
249                                BEGIN                                        ; b3
250     0134                       IF PTYPE != 1 THEN
251     0136                          IF AND(PDIN:DIN_ALMCT,32) THEN GOSUB ACK1
252     0138                       IF (FTRIP4) THEN GOSUB ACK1
253                                END                                          ; e3
254     0141                   IF PSTAT = 0 THEN IF (PDIN:DIN_VAL = PREV) THEN GOSUB ALM
255     0142                   IF (PSTAT) THEN
256                                BEGIN                                        ; b4
257     0144                       IF (PDIN:DIN_VAL != PREV) THEN GOSUB CLR
258     0145                       NXT = PSTAT
259     0147                       IF (PPASS) & (PSTAT = 2) THEN NXT = 0
260     0148                       LAST = OR(LAST,NXT)
261                                END                                          ; e4
262                            END                                              ; b2
263     0149                ELSE IF !AND(PDIN:DIN_ALMCT,16)&!AND(PDIN:DIN_ALMCT,4)&AND(PDIN:DIN_FLG,8) THEN
264     0150                   LET PSTAT = 0
265                        END                                              ; e1
266     0151            STEP PDIN BY 1
267     0152            STEP PTYPE BY 1
268     0153            STEP PREV BY 1
269     0154            STEP PSTAT BY 1
270     0155            STEP PPASS BY 1
271             ;
272             ;                                       PSTAT NOW POINTING AT STAT6
273     0156            NEXT ILOOP
274             ;
275             ;               E N D   D I G I T A L S
```

```
276             ;
277             ;               B E G I N   A N A L O G S  R E D  A N D  Y E L L O W :
278             ;
279             ;               Function:       Recognize analog red and yellow alarms
280             ;                               and associate with graphic strings.  Turn
281             ;                               on alarm ACK device, turn on Lamp, and
282             ;                               flash message on graphic.
283             ;
284     0157            ANALOG = 0
285     0158            ASSIGN PSKID = SKID6
286     0159            ASSIGN PAI = 'TI-E101-2'
287     0160            FOR JLOOP = 6 TO 25
288             ;       IF !SELECT() THEN GOTO AGAIN                    ; If some alarm exists
289     0161            IF (PTYPE) THEN
290                        BEGIN                                                ; b1
291     0162               BITVAL = PAI:AI_ALMCT
292     0163               IF !AND(BITVAL,16) THEN                      ; If not inhibited
293                           BEGIN                                             ; b2
294     0164                  IF AND(BITVAL,15) THEN                    ; If alarm exists and is current
295                              BEGIN                                          ; b3
296     0166                     IF AND(BITVAL,32) THEN NXT = 2         ; If alarm is ack
297                              ELSE                                   ; If alarm is unack
298                                 BEGIN                                       ; b4
299     0167                        IF PSKID != BITVAL THEN
300                                     BEGIN                                   ; b5
301     0168                            LET PSKID = BITVAL
302     0170                            IF BITVAL < 3 THEN FALMON = 1
303     0171                            PDEV:DEV_TMP0 = 2
304     0172                            NXT = 1
305                                     END                                     ; e5
306     0173                        ELSE NXT = 1
307                                 END                                         ; e4
308                              END                                            ; e3
309                           ELSE                                      ; Else there is no alarm
310                              BEGIN                                          ; b6
311     0174                     NXT = 0
312     0175                     LET PSKID = 0
313                              END                                            ; e6
314                           END                                               ; e2
315                        ELSE                                         ; Else alarm is inhibited
316                           BEGIN                                             ; b7
317     0176                  NXT = 0
318     0177                  LET PSKID = 0
319                           END                                               ; e7
320     0178               ANALOG = OR(ANALOG,NXT)
321                        END                                                  ; e1
322     0179            STEP PREV BY 1
323     0180            STEP PSTAT BY 1
324     0181            STEP PSKID BY 1
325     0182            STEP PTYPE BY 1
326     0183            IF (JLOOP!=8)&(JLOOP!=9)&(JLOOP!=14)&(JLOOP!=15)&(JLOOP!=19)&(JLOOP!=20) THEN
327                        BEGIN
328     0184               STEP PAI BY 1                    ; Step PAI if JLOOP != 8, 9, 14, 15, 19, OR 20
329                        END
330     0185            NEXT JLOOP
```

```
331             ;
332             ;               E N D   A N A L O G S
333             ;
334             ;               B E G I N   A N A L O G   W H I T E   A L A R M S
335             ;
336             ;               Function:       Recognize AI White alarms, associate
337             ;                               with graphic string.  Turn on alarm
338             ;                               ACK device, turn on Lamp, and flash
339             ;                               message on graphic.
340             ;
341     0186            WHITE = 0
342     0187            ASSIGN PALARM = AI0
343     0188            ASSIGN PAI = 'TI-E101-2'
344     0189            ASSIGN PREV = FRV6
345     0190            ASSIGN PSTAT = INT6
346     0191            ASSIGN PTYPE = TYPE6
347     0192                    IN1 = 6
348     0193                    IN2 = 6
349     0194                    GOSUB ANLG
350     0195            ASSIGN PAI = 'WFI-E101'
351     0196                    IN1 = 7
352     0197                    IN2 = 9
353     0198                    GOSUB ANLG
354     0199            ASSIGN PAI = 'DI-E101-2'
355     0200                    IN1 = 10
356     0201                    IN2 = 12
357     0202                    GOSUB ANLG
358     0203            ASSIGN PAI = 'WFI-E102'
359     0204                    IN1 = 13
360     0205                    IN2 = 15
361     0206                    GOSUB ANLG
362     0207            ASSIGN PAI = 'FQI-E102'
363     0208                    IN1 = 16
364     0209                    IN2 = 17
365     0210                    GOSUB ANLG
366     0211            ASSIGN PAI = 'WFI-E104'
367     0212                    IN1 = 18
368     0213                    IN2 = 20
369     0214                    GOSUB ANLG
370     0215            ASSIGN PAI = 'DI-E104-2'
371     0216                    IN1 = 21
372     0217                    IN2 = 25
373     0218                    GOSUB ANLG
374             ;
375             ;               E N D   A N A L O G   W H I T E   A L A R M S
376             ;
377     0220            IF (INT6 & INT6 < 3) THEN LET LEAK3 = INT6 + 4                  ; TI-E101-2 Hi
378     0222               ELSE IF INT6 = 3 THEN LET LEAK3 = 9                          ; TI-E101-2 INHA
379     0223                  ELSE LET LEAK3 = 0                                ; No alarm
380
381     0225            IF (INT8 & INT8 < 3) THEN LET LEAK0 = INT8                      ; WFI-E101 Lo-Lo
382     0227               ELSE IF (INT9 & INT9 < 3) THEN LET LEAK0 = INT9 + 2          ; WFI-E101 Lo
383     0229                  ELSE IF INT8 = 3 THEN LET LEAK0 = 9                       ; WFI-E101 INHA
384     0230                     ELSE LET LEAK0 = 0                                     ; No alarm
385
```

```
386  0232          IF (INT14 & INT14 < 3) THEN LET LEAK1 = INT14                    ; WFI-E102 Lo-Lo
387  0234             ELSE IF (INT15 & INT15 < 3) THEN LET LEAK1 = INT15 + 2 ; WFI-E102 Lo
388  0236                ELSE IF INT14 = 3 THEN LET LEAK1 = 9                       ; WFI-E102 INHA
389  0237                   ELSE LET LEAK1 = 0                                      ; No alarm
390
391  0239          IF (INT17 & INT17 < 3) THEN LET LEAK4 = INT17 + 4         ; TI-E104-2 Hi
392  0241             ELSE IF INT17 = 3 THEN LET LEAK4 = 9                   ; TI-E104-2 INHA
393  0242                ELSE LET LEAK4 = 0                                  ; No alarm
394
395  0244          IF (INT19 & INT19 < 3) THEN LET LEAK2 = INT19                    ; WFI-E104 Lo-Lo
396  0246             ELSE IF (INT20 & INT20 < 3) THEN LET LEAK2 = INT20 + 2 ; WFI-E104 Lo
397  0248                ELSE IF INT19 = 3 THEN LET LEAK2 = 9                       ; WFI-E104 INHA
398  0249                   ELSE LET LEAK2 = 0                                      ; No alarm
399
400  0250          MARK STEP "BELOW",0,0
401  0251          SORT = 0
402  0253          IF LAST > 3 THEN PRINT ADI," LAMP VARIABLE IS GREATER THAN 3!"
403  0254          CURENT = OR(LAST,WHITE)
404  0255          CURENT = OR(CURENT,ANALOG)
405  0257          IF AND(CURENT,1) THEN LMPVAR = 2
406  0259             ELSE IF AND(CURENT,2) THEN LMPVAR = 1
407  0260                     ELSE LMPVAR = 0
408  0261          LAMP ADI,SKDKEY,LMPVAR
409  0262          MARK STEP "BOTTOM",0,0
410  0263          NOCRIT
411  0265          IF AND('AMU-ALARM':DEV_FLG,64)THEN PUTMANL 'AMU-ALARM'
412  0267          IF AND('P-E-101':DEV_FLG,64)THEN PUTMANL 'P-E-101'
413  0269          IF AND('A-E-101':DEV_FLG,64)THEN PUTMANL 'A-E-101'
414  0271          IF AND('P-E-102':DEV_FLG,64)THEN PUTMANL 'P-E-102'
415  0273          IF AND('A-E-102':DEV_FLG,64)THEN PUTMANL 'A-E-102'
416  0275          IF AND('P-E-104':DEV_FLG,64)THEN PUTMANL 'P-E-104'
417  0277          IF AND('A-E-104':DEV_FLG,64)THEN PUTMANL 'A-E-104'
418  0278          GOTO AGAIN
419          ;
420          ;              B E G I N   A N A L O G   W H I T E   S U B R O U T I N E
421          ;
422          ;              ILOOP     EPN           ALARM LEVEL
423          ;              -----   --------        -----------
424          ;                6     TI-E101-2          Hi
425          ;                8     WFI-E101           Lo-Lo
426          ;                9     WFI-E101           Lo
427          ;               14     WFI-E102           Lo-Lo
428          ;               15     WFI-E102           Lo
429          ;               17     TI-E104-2          Hi
430          ;               19     WFI-E104           Lo-Lo
431          ;               20     WFI-E104           Lo
432          ;
433  0279    ANLG:    FOR ILOOP = IN1 TO IN2
434          ;        IF !SELECT() THEN GOTO AGAIN
435  0280             IF PTYPE = 1 THEN
436                      BEGIN                                          ; b0
437  0281                BITVAL = PAI:AI_ALMCT
438          ;.
439  0282                IF AND(BITVAL,16) THEN                    ; If inhibited
440                          BEGIN
```

```
441  0283                    LET PSTAT = 3
442  0284                    DLEAK3 = 1
443  0285                    GOSUB ACKK
444                          END
445          ;
446  0286            IF !(AND(BITVAL,16)) THEN            ; If not inhibited and alarm enabled
447                    BEGIN                                                ; b1
448  0287              IF (PREV) THEN                     ; If rev acting, Lo-Lo, or Lo
449                      BEGIN                                              ; b2
450  0288                IF (PSTAT = 0) THEN
451                        BEGIN                                            ; b3
452  0290                    IF (PAI:AI_MEAS <= PALARM) THEN GOSUB ALM
453                        END                                              ; e3
454                      ELSE
455                        BEGIN                                            ; b4
456  0291                    IF PSTAT = 1 THEN
457                            BEGIN                                        ; b5
458  0293                      IF (FTRIP4) THEN GOSUB ACK1
459  0296                      IF PTYPE !=1 THEN IF AND(PAI:AI_ALMCT,32) THEN GOSUB ACK1
460                            END                                          ; e5
461  0298                    IF (PAI:AI_MEAS >= PALARM + PAI:AI_DEAD) THEN GOSUB CLR
462                          END                                            ; e4
463                      END                                              ; e2
464                    ELSE                               ; If fail safe, Hi-Hi, or Hi
465                      BEGIN                                              ; b2.1
466  0299                IF PSTAT = 0  THEN
467                        BEGIN                                            ; b6
468  0301                    IF (PAI:AI_MEAS >= PALARM) THEN GOSUB ALM
469                        END                                              ; e6
470                      ELSE
471                        BEGIN                                            ; b7
472  0302                    IF PSTAT = 1 THEN
473                            BEGIN                                        ; b8
474  0304                      IF (FTRIP4) THEN GOSUB ACK1
475  0307                      IF PTYPE != 1 THEN IF AND(PAI:AI_ALMCT,32) THEN GOSUB ACK1
476                            END                                          ; e8
477  0309                    IF (PAI:AI_MEAS <= PALARM - PAI:AI_DEAD) THEN GOSUB CLR
478                          END                                            ; e7
479                      END                                              ; e2.1
480  0310              NXT = PSTAT
481  0312              IF (PPASS) & (PSTAT = 2) THEN NXT = 0
482                    END                                                  ; e1
483                    ELSE
484                      BEGIN                                              ; b9
485  0313                LET PSTAT = 3
486  0314                NXT = 0
487                      END                                              ; e9
488  0315            WHITE = OR(WHITE,NXT)
489                  END                                                  ; e0
490  0316            ELSE LET PSTAT = 0
491  0317            STEP PALARM BY 1
492  0318            IF (ILOOP!=7)&(ILOOP!=8)&(ILOOP!=13)&(ILOOP!=14)&(ILOOP!=18)&(ILOOP!=19) THEN
493                    BEGIN
494  0319              STEP PAI BY 1                      ; Step PAI if ILOOP != 7, 8, 13, 14, 18, OR 19
495                    END
```

```
496   0320          STEP PSTAT BY 1
497   0321          STEP PTYPE BY 1
498   0322          STEP PREV BY 1
499   0323          STEP PPASS BY 1
500   0324          NEXT ILOOP
501   0325          RETURN
502         ;
503         ;               E N D   A N A L O G   W H I T E   S U B R O U T I N E
504         ;
505         ;     BEGIN CLEAR, ALARM, ACKNOWLEDGE, MESSAGE, ERROR, AND TERMINATE SUBROUTINES:
506         ;
507         ;
508   0326  CLR:  LET PSTAT = 0
509   0328        IF ILOOP = 8 THEN LET INT9 = 0          ; Clear Hi alarm for Hi-Hi to Hi transition
510   0330        IF ILOOP = 14 THEN LET INT15 = 0        ; Clear Hi alarm for Hi-Hi to Hi transition
511   0332        IF ILOOP = 19 THEN LET INT20 = 0        ; Clear Hi alarm for Hi-Hi to Hi transition
512               ;
513               ; If not white, then unack:
514               ;
515   0334        IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32) THEN
516   0335              PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
517   0336        LET FCLEAR = 1                          ; Clear flag
518   0337        GOSUB ACKK                              ; Print message
519   0338        RETURN
520         ;
521   0339  ALM:  LET PSTAT = 1                           ; Int set to one
522               ;
523               ; If not white, then unack:
524               ;
525   0341        IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32) THEN
526   0342              PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
527               ;
528               ; Turn audible alarm on if white or yellow
529               ;
530   0343        PDEV:DEV_TMP0 = 2
531   0345        IF AND(PTYPE,3)THEN FALMON = 1
532   0346        LAMP AD1,SKDKEY,2
533   0347        CRIT
534   0348        GOSUB ACKK                              ; Print Message
535   0349        RETURN
536         ;
537   0350  ACK1: LET PSTAT = 2                           ; Ack the alarm
538   0351        LET FACK = 1                            ; Set ack flag for C program
539         ;
540         ;           For multiple AI white alarms on one EPN (use with string packet 78):
541         ;
542         ;
543   0352  ACKK: IF DLEAK3 = 1 THEN                                  ; If inhibited
544   .                 BEGIN
545   0353              LET DLEAK3 = 0
546   0354              RETURN                                ; Skip message routine
547               END
548         ;
549         ;           End multiple AI white alarms on one EPN
550         ;
```

```
551          ;              A L A R M   T Y P E R   S T R I N G S
552          ;
553          ;              For DIN white (there are no DIN R or Y), AI white,
554          ;              and program status.
555          ;
556  0356            IF ILOOP = 0 THEN STRNGO = "ELUANT AGITATOR ON       "    ; YS-A-E101
557  0358            IF ILOOP = 1 THEN STRNGO = "ELUANT PUMP ON           "    ; YS-P-E101
558  0360            IF ILOOP = 2 THEN STRNGO = "ANTIFOAM AGITATOR ON     "    ; YS-A-E102
559  0362            IF ILOOP = 3 THEN STRNGO = "ANTIFOAM PUMP ON         "    ; YS-P-E102
560  0364            IF ILOOP = 4 THEN STRNGO = "DECON AGITATOR ON        "    ; YS-A-E104
561  0366            IF ILOOP = 5 THEN STRNGO = "DECON PUMP ON            "    ; YS-P-E104
562  0368            IF ILOOP = 6 THEN STRNGO = "TK-E-101 LIQUID TEMP HI  "    ; TI-E101-2 Hi
563  0370            IF ILOOP = 8 THEN STRNGO = "TK-E-101 WT FACTOR LO-LO "    ; WFI-E101 Lo-Lo
564  0372            IF ILOOP = 9 THEN STRNGO = "TK-E-101 WT FACTOR LO    "    ; WFI-E101 Lo
565  0374            IF ILOOP = 14 THEN STRNGO = "TK-E-102 WT FACTOR LO-LO "   ; WFI-E102 Lo-Lo
566  0376            IF ILOOP = 15 THEN STRNGO = "TK-E-102 WT FACTOR LO    "   ; WFI-E102 Lo
567  0378            IF ILOOP = 17 THEN STRNGO = "TK-E-104 LIQUID TEMP HI  "   ; TI-E104-2 Hi
568  0380            IF ILOOP = 19 THEN STRNGO = "TK-E-104 WT FACTOR LO-LO "   ; WFI-E104 Lo-Lo
569  0382            IF ILOOP = 20 THEN STRNGO = "TK-E-104 WT FACTOR LO    "   ; WFI-E104 Lo
570  0384            IF ILOOP = 33 THEN STRNGO = "START AMUBATCH PGM "
571  0386            IF ILOOP = 34 THEN STRNGO = "AMUBATCH WRONG UNIT"
572  0388            IF ILOOP = 35 THEN STRNGO = "TERMINATE AMUBATCH "
573          ;
574          ; If acknowledging alarm, then:
575          ;
576  0389    AMSG:     IF PTYPE = 1 THEN
577                       BEGIN
578                    ;
579                    ;  If white alarm, print:
580                    ;
581  0391                 IF (FACK = 0) & (FCLEAR = 0) & (FSTART = 0) THEN PRINT ADI,"WHITE ALARM %p",STRNGO
582  0392                 IF (FACK) THEN
583                          BEGIN
584  0393                    PRINT ADI,"%p ALARM ACKNOWLEDGE",STRNGO
585  0394                    LET FACK = 0
586                          END
587                       END
588          ;
589          ; If CLEAR, print it is clear and reset flag:
590          ;
591  0395            IF (FCLEAR) THEN
592                       BEGIN
593  0396                 PRINT ADI,"%p ALARM CLEAR",STRNGO
594  0397                 LET FCLEAR = 0
595                       END
596          ;
597          ; If STARTUP, print startup message:
598          ;
599  0398            IF (FSTART) THEN
600                       BEGIN
601  0399                 LET FSTART = 0
602  0400                 PRINT ADI,"%p",STRNGO
603                       END
604  0401             RETURN
605          ;
```

```
606             ; if ERROR - print message, start timer, and set flag
607             ;
608     0402    ERRSUB:   IF ERRN != ERRNUM() THEN
609                         BEGIN
610     0403                TIMERON STIME0
611     0404                TIMERCLR STIME0
612     0405                LET FSN0 = 1
613     0406                LET ERRN = ERRNUM()
614     0407                LET ERRL = ERRLIN()
615     0408                PRINT ADI,"ERROR # %i AT LINE %i",errn,errl
616                         END
617     0409              ERSCLR
618     0410              GOTO AGAIN
619             ;
620             ; if FABORT = 1, then abort program
621             ;
622             ; TERM:  LET FABORT = 0
623     0411    TERM:     LET ILOOP = STAR + 2
624     0412              LET FSTART = 1
625     0413              GOSUB ACKK
626                       END
```

Total Errors Detected 0 Total Warnings 0
Sequence Program Size = 8016 Bytes, 501 Pages
26 Forward Reference(s)

LERFBATCH

Sequence Instruction Compiler Listing 15:35:20 24-Feb-95 Page 1

```
1           ;       L E R F B A T C H . S E Q                                VERSION  2
2           ;
3           ;       Function:       Sequence program for Lamp function,
4           ;                       message display on Graphic number 024,
5           ;                       and printer messages.
6           ;
7           ;       Written By      : R. N. Kersey
8           ;       Date            : 05-31-92
9           ;       Facility        : 242-A Evaporator Bldg.
10          ;       Machine         : VAX 4104-A
11          ;       Op. System      : VAX/VMS 5.5-2AH
12          ;       D/3 Version     : 6.3-2 BASELINE
13          ;       Compiler        : SIC
14          ;       Build command   : @[320,20]LERFBATCH
15          ;
16          ;       UNIT = 21, SKDKEY = 40
17          ;
18          ;       Assoc. UPF:     [300,1]_PRCESS.___      PRCESS.LST
19          ;       Assoc. RPF:     [300,1]_242A.___        242A.LST
20          ;       SDB:            [300,1]PCM0A.SDB        PCM0ASDB.CMD
21          ;
22          ;       POINTER   SDB VAR.      FUNCTION
23          ;       -------   ---------     ----------------------------------------
24          ;       PSTAT     INTn          Current EPN alarm state is an integer tied to
25          ;                               graphic flashing string of value:
26          ;       .         (n=0-100)             0 = previously not alarmed
27          ;       .                               1 = previously alarmed, not ack
28          ;       .                               2 = previously alarmed, is ack
29          ;       .                               3 = alarm is inhibited
30          ;       .                               4 = input is off scan
31          ;       .                               5 = input is bad
32          ;       .         LEAKn         Special INT variable for multiple white alarms on an EPN
33          ;       PDIN      'EPN'         ---
34          ;       PTYPE     TYPEn         alarm.type 0 = none, 1 = white, 2 = yellow,
35          ;                                  3 = yellow and yellow, 4 = red,
36          ;                                  6 = yellow and red
37          ;       PREV      FRVn          Reverse acting (not fail safe) flag (1=set)
38          ;       PSKID     SKIDn         Used for analog alarms
39          ;       PPASS     FPASn         Turn lamp off when alarm is ACKnowledged
40          ;       PALARM    FALMn         Analog alarm flag
41          ;       ---       FNING         Unused error flag
42          ;
43          ; MODIFICATION LOG:
44          ;
45          ;       05-NOV-92       Deleted LERF valve MOV-43-1 associated          M. Teats
46          ;                       logic and DINs (search for WAS).
47          ;
48          ;       30-NOV-92       Added SA5, SA6, SA7 for line leaks;             M. Teats
49          ;                       renumbered SA0 to SA7; revised leak logic.
50          ;
51          ;       16-DEC-92       Added analog alarm code, and comments.          M. Teats
52          ;
53          ;       30-DEC-92       Fixed white alarm print message.        M. Teats
54          ;
55          ;       29-JAN-93       Added YS-PLC-1.                                 M. Teats
```

```
56              ;
57              ;       01-MAR-93       Change LDS-BSN42, 43, 44 to fail-safe           M. Teats
58              ;                       (alarms on voltage absent in MUX,
59              ;                       change FRV_ from 1 to 0).
60              ;
61              ;       11-MAR-93       Modified SA_ (SA0...SA8) variable code for      M. Teats
62              ;                       as-found.
63              ;
64              ;       08-JUN-93       Added comments and baselined for        R. Persin
65              ;                       D/3 version 3.3.  CCN# 93JUN01-01.
66              ;
67              ;       11-JUL-94       Added LERF DINs LSH60A12, LSH60A13              R. Kersey
68              ;                       YSETFCOM, and YS60A56
69              ;
70              ;       15-DEC-94       Revised header to indicate the new VMS          J. Berger
71              ;                       system. Rebaselined document. Removed
72              ;                       obsolite Select() document.
73              ;
74              ;  END MOD LOG
75              ;
76              UNIT    LERF
77      0000            NOP
78      0001            STAR = 26                       ; Start message
79      0002            LET SKDKEY = 40         ; SKID Key
80      0003            LET UNITT = 21                  ; SEQ Unit
81      0004            ASSIGN PDEV = 'LERF-ALM' ; Alarm device
82      0005            ASSIGN PTYPE = TYPE25
83      0006            LET ADI = 0                     ; Alarm destination index
84      0007            LET FSN0 = 0                    ; No alarm yet
85      0008            TIMEROFF STIME0         ; Alarm timer off
86      0009            TIMERCLR STIME0         ; Alarm timer clear
87      0010            STRNG0 = " "                    ; Blank 30 character string
88      0011            LET ERRN = 0                    ; Set error number to zero
89      0012            LET ERRL = 0                    ; Set error line to zero
90      0013            LET FTRIP4 = 0                  ; FTRIP = 1 when audible horn is to be turned on
91      0014            GETUNIT TRIP                    ; Which sequence unit is program running in
92      0015            IF TRIP != UNITT THEN           ; Abort if running in wrong seq.  unit
93                         BEGIN
94      0016               FSTART = 1
95      0017               ILOOP = STAR + 1
96      0018               PTYPE = 0
97      0019               GOSUB ACKK
98      0020               GOTO TERM
99                         END
100             ;       IF !SELECT() THEN GOTO AGAIN
101             ;
102             ;       B E G I N   I N I T I A L I Z I N G
103             ;
104             ;       DEVICES:
105             ;               LERF-ALM
106             ;               LERFILK
107             ;       DIGITALS:               ILOOP#
108             ;               YS-P-43         0
109             ;               LDS-A1          1
110             ;               Y-P-44          2
```

```
111             ;               ZS-105PS1       3
112             ;               ZS-105PS2       4
113             ;               YS-P-42         5
114             ;               LDS-BSN42       6
115             ;               LDS-BSN43       7
116             ;               LDS-BSN44       8
117             ;               LXS-LCH42       9
118             ;               LXS-BSN42       10
119             ;               LXS-LCH43       11
120             ;               LXS-BSN43       12
121             ;               LXS-LCH44       13
122             ;               LXS-BSN44       14
123             ;               YS-PLC-1        15
124             ;               LSH60A12        16
125             ;               LSH60A13        17
126             ;               YSETFCOM        18
127             ;               YS60A56         19
128             ;       ANALOGS:
129             ;               LDI-A1          20
130             ;               LI-LCH42        21
131             ;               LI-BSN42        22
132             ;               LI-LCH43        23
133             ;               LI-BSN43        24
134             ;               LI-LCH44        25
135             ;               LI-BSN44        26
136             ;
137     0021            STRNG0 = "INIT"
138     0022    INIT:   LET FABORT = 0                  ; if FABORT = 1 then abort program
139     0023            LET FSTART = 0                  ; Program start flag = 0
140     0024            LAMP ADI,SKDKEY,0               ; Turn off the annunciator key
141     0025            LET LSG0 = " "                  ; Clear the string variables
142     0026            LET LSG1 = " "
143     0027            LET LSG2 = " "
144     0028            LET LSG3 = " "
145     0029            LET LSG4 = " "
146     0030            LET PDEV:DEV_TMP0 = 1                           ; Set alarm device to non-alarm state
147     0031            IF AND(PDEV:DEV_ALMCT,16) THEN                  ; If alarm device is inhibited, remove the inhibit
148     0032              PDEV:DEV_ALMCT=XOR(PDEV:DEV_ALMCT,16)
149     0033            IF AND(PDEV:DEV_ALMCT,32) THEN                  ; If alarm device is ack, clear it
150     0034               PDEV:DEV_ALMCT = XOR(PDEV:DEV_ALMCT,32)
151             ;
152             ;       First, set all flags and integers:
153             ;
154     0035            ASSIGN PDIN = 'YS-P-43'         ; Assign pointer to first DIN
155     0036            ASSIGN PSTAT = INT0
156     0037            ASSIGN PTYPE = TYPE0
157     0038            ASSIGN PREV = FRV0
158     0039            ASSIGN PPASS = FPAS0
159     0040            ASSIGN PALARM = FALM0
160     0041            ASSIGN PSKID = SKID0
161             ;
162     0042            FOR ILOOP = 0 TO 26
163     0043            LET PSKID = 0
164     0044            LET PPASS = 0
165     0045            LET PREV = 0
```

```
166  0046            LET PTYPE = 0                                          ; Default to no alarm
167  0048            IF PSTAT > 5 THEN PSTAT = 0                            ; Reset current status if undefined
168  0049            PALARM = 0                                             ; Default previous status to no alarms
169  0050            IF ILOOP <= 19 THEN                                    ; For DINs
170                     BEGIN
171  0051               IF AND(PDIN:DIN_DFLG,3)THEN                         ; Check for alarms (must be a red or yellow)
172                       BEGIN
173  0052                 LET PTYPE = 2                                     ; Default to yellow
174  0054                 IF AND(PDIN:DIN_DFLG,1) THEN LET PREV = 1    ; reverse acting
175  0056                 IF AND(PDIN:DIN_DFLG,4) THEN LET PTYPE = 4   ; Set to red critical alarm
176                       END
177  0058               IF ILOOP < 19 THEN STEP PDIN BY 1                   ; Point to next DIN
178                     END
179  0059            ELSE PTYPE = 0                                         ; For AINs, set to no alarm
180  0060    SKP:    STEP PSTAT BY 1
181  0061            STEP PTYPE BY 1
182  0062            STEP PREV BY 1
183  0063            STEP PPASS BY 1
184  0064            STEP PALARM BY 1
185  0065            STEP PSKID BY 1
186  0066            NEXT ILOOP
187          ;
188  0067            TYPE0 = 1       ; YS-P-43
189  0068            TYPE1 = 2       ; LDS-A1
190  0069            TYPE2 = 1       ; YS-P-44
191  0070            TYPE3 = 1       ; ZS-105PS1
192  0071            TYPE4 = 1       ; ZS-105PS2
193  0072            TYPE5 = 1       ; YS-P-42
194  0073            TYPE6 = 2       ; LDS-BSN42
195  0074            TYPE7 = 2       ; LDS-BSN43
196  0075            TYPE8 = 2       ; LDS-BSN44
197  0076            TYPE9 = 4       ; LXS-LCH42
198  0077            TYPE10 = 4      ; LXS-BSN42
199  0078            TYPE11 = 4      ; LXS-LCH43
200  0079            TYPE12 = 4      ; LXS-BSN43
201  0080            TYPE13 = 4      ; LXS-LCH44
202  0081            TYPE14 = 4      ; LXS-BSN44
203  0082            TYPE15 = 4      ; YS-PLC-1
204  0083            TYPE16 = 2      ; LSH60A12
205  0084            TYPE17 = 4      ; LAH60A13
206  0085            TYPE18 = 4      ; YSETFCOM
207  0086            TYPE19 = 2      ; YS60A56
208          ;
209  0087            TYPE20 = 2      ; LDI-A1
210          ;
211  0088            FRV0 = 1        ; YS-P-43
212  0089            FRV2 = 1        ; YS-P-44
213  0090            FRV5 = 1        ; YS-P-42
214  0091            FRV19 = 0       ; YS60A56
215          ;
216  0092            ANALOG = 0
217          ;
218          ;       E N D   I N I T I A L I Z I N G
219          ;
220          ;       S E N D   P R O G R A M   S T A R T   M E S S A G E
```

```
221             ;
222     0093            ILOOP = STAR
223     0094            FSTART = 1
224     0095            PTYPE = 0
225     0096            GOSUB ACKK
226     0097            ON ERROR GOSUB ERRSUB
227             ;
228             ;       B E G I N   P R O G R A M   L O O P
229             ;
230     0098    AGAIN:  MARK STEP "WTUNTL",0,0
231     0099            STRNG0 = "WAIT"
232             ;       WAITUNTIL SELECT() ¦ FABORT      ; Wait until micro is selected or program aborted
233             ;       IF (FABORT) THEN GOTO TERM       ; Terminate program if FABORT flag is set
234     0100            MARK STEP "RNNING",0,0
235     0101            WAIT 1
236     0102            FTRIP4 = 0
237             ;
238     0103            IF (FSNO) THEN                   ; If alarm flag is set
239     0104               IF STIMEO > 60 THEN           ; If alarm flag active for 1 minute
240                           BEGIN                     ; Reset timer and turn off flag
241     0105                  TIMEROFF STIMEO
242     0106                  TIMERCLR STIMEO
243     0107                  LET FSNO = 0
244     0108                  LET ERRN = 0
245     0109                  LET ERRL = 0
246                           END
247             ;
248             ;       If LERF-ALM $T1 is alarmed and it's ALMCT indicates alarms are acknowledged:
249             ;
250     0110            IF (PDEV:DEV_TMP0 = 2) & AND(PDEV:DEV_ALMCT,32) THEN
251                        BEGIN
252     0111               'LERF-ALM':DEV_TMP0 = 1       ; Put LERF-ALM in ACKNWLDG state
253     0112               LET FTRIP4 = 1                ; Set for Unit alarm acknowledgement, system to graphic
254     0113               ACK = 1                       ; Set for Unit alarm acknowledgement, graphic to system
255     0114               CRIT
256     0115               LAMP AD1,SKDKEY,1
257                        END
258             ;
259             ;       BEGIN DINs:
260             ;
261     0116            ASSIGN PDIN = 'YS-P-43'
262     0117            ASSIGN PTYPE = TYPE0
263     0118            ASSIGN PREV = FRV0
264     0119            ASSIGN PSTAT = INT0
265     0120            ASSIGN PPASS = FPAS0
266     0121            LET CURENT = 0                   ; Used to set lamp status
267     0122            LET LAST = 0
268     0123            LET ANALOG = 0
269     0124            LET NXT = 0
270     0125            LMPVAR = 0
271     0126            BITVAL = 0
272             ;
273     0127    LOP:    FOR ILOOP = 0 TO 19
274             ;       IF !SELECT() THEN GOTO AGAIN
275     0128            IF ( PTYPE > 0 ) THEN                          ; If some type of alarm
```

```
276                     BEGIN                                           ;be1
277     0130            IF AND(PDIN:DIN_ALMCT,16) THEN LET PSTAT = 3    ; If EPN Inhibited
278     0132            IF AND(PDIN:DIN_ALMCT,4) THEN LET PSTAT = 5             ; If EPN is bad input
279     0134            IF !AND(PDIN:DIN_FLG,8) THEN LET PSTAT = 4              ; If EPN is disabled
280     0135            IF ( PSTAT < 3 ) THEN                           ; If not inhibited
281                          BEGIN                                      ;be2
282     0136                 IF PSTAT = 1 THEN
283                             BEGIN                                   ;be3
284     0137                    IF PTYPE != 1 THEN
285     0139                       IF AND(PDIN:DIN_ALMCT,32) THEN GOSUB ACK1
286     0141                    IF (FTRIP4) THEN GOSUB ACK1
287                             END                                     ;en3
288     0144                 IF PSTAT = 0 THEN IF(PDIN:DIN_VAL = PREV) THEN GOSUB ALM
289     0145                 IF (PSTAT) THEN
290                                BEGIN                                ;be4
291     0147                       IF PDIN:DIN_VAL != PREV THEN GOSUB CLR
292     0148                       NXT = PSTAT
293     0150                       IF (PPASS) & (PSTAT = 2) THEN NXT = 0
294     0151                       LAST = OR(LAST,NXT)
295                                END                                  ;en4
296                          END                                        ;en2
297     0153                ELSE IF !AND(PDIN:DIN_ALMCT,16)&!AND(PDIN:DIN_ALMCT,4)&AND(PDIN:DIN_FLG,8) THEN LET PSTAT = 0
298                         END                                         ;en1
299     0155            IF ILOOP < 19 THEN STEP PDIN BY 1
300     0156            STEP PTYPE BY 1
301     0157            STEP PREV BY 1
302     0158            STEP PSTAT BY 1
303     0159            STEP PPASS BY 1
304     0160            NEXT ILOOP
305             ;
306             ;       END DINs:
307             ;
308             ;       BEGIN ANALOG RED AND YELLOWS:
309             ;
310     0161            ANALOG = 0
311     0162            LET STRNG0 = "RUN"
312     0163            ASSIGN PAI = 'LDI-A1'
313     0164            ASSIGN PSKID = SKID20
314     0165            ASSIGN PTYPE = TYPE20
315             ;
316     0166            FOR JLOOP = 20 TO 26
317             ;       IF !SELECT() THEN GOTO AGAIN
318     0167            IF (PTYPE) THEN                                         ;If some alarm exists
319                        BEGIN                                        ;be1
320     0168               BITVAL = PAI:AI_ALMCT
321     0169               IF !AND(BITVAL,16)THEN
322                           BEGIN                                     ;be2
323     0170                  IF AND(BITVAL,15)THEN
324                              BEGIN                                  ;be3
325     0172                     IF AND(BITVAL,32)THEN NXT = 2
326                              ELSE
327                                 BEGIN                               ;be4
328     0173                        IF PSKID !=BITVAL THEN
329                                    BEGIN                            ;be5
330     0175                           IF BITVAL < 3 THEN FALMON = 1
```

```
331   0176                          LET PSKID = BITVAL
332   0177                          LET PDEV:DEV_TMP0 = 2
333   0178                          NXT = 1
334                                 END                                    ;en5
335   0179                      ELSE NXT = 1
336                               END                                      ;en4
337                           END                                          ;en3
338                       ELSE
339                           BEGIN
340   0180                    NXT = 0
341   0181                    LET PSKID = 0
342                           END
343                       END                                              ;en2
344                   ELSE
345                       BEGIN
346   0182                NXT = 0
347   0183                LET PSKID = 0
348                       END
349   0184                ANALOG = OR(ANALOG,NXT)
350                       END                                              ;en1
351   0185            STEP PSKID BY 1
352   0186            STEP PAI BY 1
353   0187            STEP PTYPE BY 1
354   0188            NEXT JLOOP
355             ;
356             ;       END ANALOG RED AND YELLOWS
357             ;
358   0189            MARK STEP "BELOW",0,0
359   0190            LET FTRIP4 = 0
360   0192            IF LAST > 3 THEN PRINT ADI," LAMP VAR IS GREATER THAN 3"
361   0193            CURENT = OR(LAST,ANALOG)
362   0195            IF AND(CURENT,1) THEN LMPVAR = 2
363   0197               ELSE IF AND(CURENT,2) THEN LMPVAR = 1
364   0198                    ELSE LMPVAR = 0
365   0199            LAMP ADI,SKDKEY,LMPVAR                              ; LMPVAR 0=off, 1=on, 2=blink
366   0200            MARK STEP "BOTTOM",0,0
367   0201            NOCRIT
368   0203            IF AND('LERF-ALM':DEV_FLG,64) THEN PUTMANL 'LERF-ALM'
369             ;
370             ;       B E G I N   L E R F   L E A K
371             ;
372             ;       Each LDE represents 0.32 mA of the 4-20 mA signal, or 2% with the
373             ;       following exception.  During ATPs, the percent increment between
374             ;       leak detector zones was found to be 2.25 to 2.5%.  The following
375             ;       code corresponds to the following ATP findings:
376             ;
377             ;                                 % Reading
378             ;               Sensor            on LDI-A1       mA reading
379             ;               ------            ---------       ----------
380             ;               LDE-01              2.29            4.35
381             ;               LDE-02              4.61            4.72
382             ;               LDE-03              6.88            5.09
383             ;               LDE-04              9.15            5.45
384             ;               LDE-05             11.43            5.81
385             ;               LDE-06             13.95            6.22
```

```
386           ;                LDE-07          16.50          6.62
387           ;                LDE-08          18.76          6.98
388           ;
389           ;
390           ;        Allow 1 % error for LDE trip.  The SA_ variables, where
391           ;        the underscore is a number from 0-7, are tied to graphic
392           ;        indications of the alarmed leak detector zone.
393           ;
394    0204            AI10 = 'LDI-A1':AI_INVL
395    0205            IF AI10 >= 20 THEN
396                       BEGIN
397    0206               SA0 = 1       ; LDE-01
398    0207               SA1 = 1       ; LDE-02
399    0208               SA2 = 1       ; LDE-03
400    0209               SA3 = 1       ; LDE-04
401    0210               SA4 = 1       ; LDE-05
402    0211               SA5 = 1       ; LDE-06
403    0212               SA6 = 1       ; LDE-07
404    0213               SA7 = 1       ; LDE-08
405                       END
406                    ELSE
407                       BEGIN
408    0215                    IF (AI10 >= 1.0) & (AI10 < 3.0) THEN SA0 = 1      ; LDE-01
409    0216                       ELSE SA0 = 0
410    0218                    IF (AI10 >= 3.0) & (AI10 < 5.5) THEN SA1 = 1      ; LDE-02
411    0219                       ELSE SA1 = 0
412    0221                    IF (AI10 >= 5.5) & (AI10 < 8.0) THEN SA2 = 1      ; LDE-03
413    0222                       ELSE SA2 = 0
414    0224                    IF (AI10 >= 8.0) & (AI10 < 10.5) THEN SA3 = 1     ; LDE-04
415    0225                       ELSE SA3 = 0
416    0227                    IF (AI10 >= 10.5) & (AI10 < 13.0) THEN SA4 = 1    ; LDE-05
417    0228                       ELSE SA4 = 0
418    0230                    IF (AI10 >= 13.0) & (AI10 < 15.5) THEN SA5 = 1    ; LDE-06
419    0231                       ELSE SA5 = 0
420    0233                    IF (AI10 >= 15.5) & (AI10 < 17.5) THEN SA6 = 1    ; LDE-07
421    0234                       ELSE SA6 = 0
422    0236                    IF (AI10 >= 17.5) & (AI10 < 20.0) THEN SA7 = 1    ; LDE-08
423    0237                       ELSE SA7 = 0
424                       END
425           ;
426           ;        E N D   L E R F   L E A K
427           ;
428    0238            GOTO AGAIN
429           ;
430           ;        E N D   P R O G R A M   L O O P
431           ;
432           ;        BEGIN CLEAR, ALARM, ACKNOWLEDGE, MESSAGE, ERROR, AND TERMINATE SUBROUTINES:
433           ;
434    0239    CLR:    LET PSTAT = 0                                          ; Int set to zero
435                    ;
436                    ; If not white, then unack:
437                    ;
438    0242            IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
439    0243            LET FCLEAR = 1                                         ; Clear flag
440    0244            GOSUB ACKK                                             ; Print message
```

```
441   0245            RETURN
442           ;
443   0246    ALM:    LET PSTAT = 1                                          ; Int set to one
444                   ;
445                   ; If not white, then unack:
446                   ;
447   0249            IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
448   0250            LET PDEV:DEV_TMP0 = 2
449   0252            IF AND(PTYPE,3) THEN FALMON = 1
450   0253.           CRIT
451   0254            LAMP ADI,SKDKEY,2
452   0255            GOSUB ACKK                                             ; Print message
453   0256            RETURN
454           ;
455   0257    ACK1:   LET PSTAT = 2                                          ; Ack the alarm
456   0258            LET FACK = 1                                           ; Set ACK flag for C program
457           ;
458           ;       ALARM TYPER STRINGS:
459           ;
460   0260    ACKK:   IF ILOOP = 0 THEN STRNGO = "LERF LECHATE PUMP #43 ON"
461   0262            IF ILOOP = 1 THEN STRNGO = "EVAPORATOR TO LERF LEAK"
462   0264            IF ILOOP = 2 THEN STRNGO = "LERF LECHATE PUMP #44 ON"
463   0266            IF ILOOP = 3 THEN STRNGO = "LERF MAIN GATE (NORTH) OPEN"
464   0268            IF ILOOP = 4 THEN STRNGO = "LERF REAR GATE (SOUTH) OPEN"
465   0270            IF ILOOP = 5 THEN STRNGO = "LERF LECHATE PUMP #42 ON"
466   0272            IF ILOOP = 6 THEN STRNGO = "CATCH BASIN 42 LEAK DETECT "
467   0274            IF ILOOP = 7 THEN STRNGO = "CATCH BASIN 43 LEAK DETECT "
468   0276            IF ILOOP = 8 THEN STRNGO = "CATCH BASIN 44 LEAK DETECT "
469   0278            IF ILOOP = 9 THEN STRNGO = "BASIN 42 LEACHATE DIAGNOSTIC"
470   0280            IF ILOOP = 10 THEN STRNGO = "BASIN 42 LIQUID DIAGNOSTIC  "
471   0282            IF ILOOP = 11 THEN STRNGO = "BASIN 43 LEACHATE DIAGNOSTIC"
472   0284            IF ILOOP = 12 THEN STRNGO = "BASIN 43 LIQUID DIAGNOSTIC  "
473   0286            IF ILOOP = 13 THEN STRNGO = "BASIN 44 LEACHATE DIAGNOSTIC"
474   0288            IF ILOOP = 14 THEN STRNGO = "BASIN 44 LIQUID DIAGNOSTIC  "
475   0290            IF ILOOP = 15 THEN STRNGO = "LERF MODEM COMM STATUS "
476   0292            IF ILOOP = 16 THEN STRNGO = "ETF SURGE TANK HI"
477   0294            IF ILOOP = 17 THEN STRNGO = "ETF SURGE TANK HIHI"
478   0296            IF ILOOP = 18 THEN STRNGO = "ETF COMMUNICATION FAIURE"
479   0298            IF ILOOP = 19 THEN STRNGO = "ETF AOV CLOSED"
480   0300            IF ILOOP = 27 THEN STRNGO = "LERFBATCH STARTING "
481   0302            IF ILOOP = 28 THEN STRNGO = "LERFBATCH TERMINATING "
482           ;
483           ; If acknowledging alarms, then:
484           ;
485   0303    AMSG:   IF PTYPE = 1 THEN
486.                    BEGIN
487                   ;
488                   ; If white alarm, print:
489                   ;
490   0305              IF (FACK = 0) & (FCLEAR = 0) & (FSTART = 0) THEN PRINT ADI,"WHITE ALARM %p",STRNGO
491   0306              IF (FACK) THEN
492                       BEGIN
493                   ;
494                   ; If white alarm, print:
495                   ;
```

```
496    0307                PRINT ADI,"%p ALARM ACKNOWLEDGE",STRNGO
497    0308                LET FACK = 0
498                        END
499                      END
500           ;
501           ; If CLEAR, print it is ack and reset the flag:
502           ;
503    0309            IF (FCLEAR) THEN
504                       BEGIN
505    0310               PRINT ADI,"%p ALARM CLEAR",STRNGO
506    0311               LET FCLEAR = 0
507                       END
508           ;
509           ; If STARTUP, print startup message:
510           ;
511    0312            IF (FSTART) THEN
512                       BEGIN
513    0313               LET FSTART = 0
514    0314               PRINT ADI,"%p",STRNGO
515                       END
516    0315            RETURN
517           ;
518           ; If ERROR - print message, start timer, and set flag:
519           ;
520    0316    ERRSUB:  IF ERRN != ERRNUM() THEN
521                       BEGIN
522    0317               TIMERON STIMEO
523    0318               TIMERCLR STIMEO
524    0319               LET FSNO = 1
525    0320               LET ERRN = ERRNUM()
526    0321               LET ERRL = ERRLIN()
527    0322               PRINT ADI,"ERROR # %i AT LINE %i",errn,errl
528                       END
529    0323               ERSCLR
530    0324            GOTO AGAIN
531           ;
532           ; If FABORT = 1, then abort program
533           ;
534           ; TERM:  LET FABORT = 0
535    0325    TERM:    LET STRNGO = "LERFBATCH TERMINATED    "
536    0326             LET ILOOP = STAR + 2
537    0327             LET FSTART = 1
538    0328             LET STRNGO = "TERM"
539    0329             GOSUB ACKK
540                      END
```

```
Total Errors Detected      0     Total Warnings      0
     Sequence Program Size = 6656 Bytes, 416 Pages
          10 Forward Reference(s)
```

CASSBATCH

Sequence Instruction Compiler Listing 15:34:24 24-Feb-95 Page 1

```
                ;       C A S S B A T C H . S E Q                                VERSION 2
                ;
                ;       Written By      : R. N. Kersey
                ;       Date            : 05-31-92
                ;       Facility        : 242-A Evaporator Bldg.
                ;       Machine         : VAX 4104-A
                ;       Op. System      : VAX/VMS 5.5-2AH
                ;       D/3 Version     : 6.3-2 BASELINE
                ;       Compiler        : SIC
                ;       Build command   : @[320,20]CASSBATCH
                ;
                ;       UNIT = 22, SKDKEY = 44
                ;
                ;       Assoc. UPF:     [300,1]_OVRVEW.___      OVRVEW.LST
                ;       Assoc. RPF:     [300,1]_242A.___        242A.LST
                ;       SDB:            [300,1]PCM0ASDB.CMD     PCM0ASDB.LST
                ;
                ; MODIFICATION LOG:
                ;
                ;       03-AUG-93       Added the following:                    M. Teats
                ;                       - comments
                ;                       - error handling
                ;                       - unit checking
                ;                       - start message
                ;                       - micro select, abort
                ;                       - redundant SABL
                ;                       - ACKK, ERRSUB, TERM subroutines
                ;                       and baselined for D/3 version 3.3.
                ;
                ;       15-DEC-94       Revised header to indicate the new VMS  J. Berger
                ;                       system. Rebaselined document. Removed
                ;                       obsolite Select() statements.
                ;
                ; END MOD LOG
                ;
                UNIT    EVAP
                        COMMON WHOLE0,WHOLE1
                ;
0000                    NOP
0001                    STA100 = 20                     ; Start message
0002                    LET STA101 = 22         ; SEQ Unit number
0003                    LET ADI = 0                     ; Alarm Destination Index
0004                    LET FSPAR0 = 0                  ; No Error yet
0005                    TIMEROFF STIME0         ; Alarm timer off
0006                    TIMERCLR STIME0         ; Alarm timer clear
0007                    MSSG0 = " "                     ; Blank 30 character string
0008                    LET ERRN = 0                    ; Set error number to zero
0009                    LET ERRL = 0                    ; Set error line to zero
0010                    GETUNIT TRIP                    ; Which sequence unit is program running in
0011                    IF TRIP != STA101 THEN          ; Abort if running in wrong seq. unit
                           BEGIN
0012                       FSTART = 1
0013                       ILOOP = STA100 + 1           ; ILoop = 21
0014                       GOSUB ACKK
0015                       GOTO TERM
```

```
                        END
            ;       IF !SELECT() THEN GOTO AGAIN
            ;
            ;       S E N D   P R O G R A M   S T A R T   M E S S A G E
            ;
0016                ILOOP = STA100                  ; ILoop = 20
0017                FSTART = 1
0018                GOSUB ACKK
0019                ON ERROR GOSUB ERRSUB
            ;
            ;       B E G I N   P R O G R A M   L O O P
            ;
0020        AGAIN:  MARK STEP "WTUNTL",0,0
0021                MSSG0 = "WAIT"
            ;       WAITUNTIL SELECT() | FABORT     ; Wait until micro is selected or program aborted
            ;       IF (FABORT) THEN GOTO TERM      ; Terminate program if FABORT flag is set
0022                MARK STEP "RNNING",0,0
0023                WAIT 1
0024                IF (FSPAR0) THEN                ; If Error flag set
0025                   IF STIME0 > 60 THEN          ; If Error flag active for 1 minute
                          BEGIN                     ; Reset timer and turn off flag
0026                      TIMEROFF STIME0
0027                      TIMERCLR STIME0
0028                      LET FSPAR0 = 0
0029                      LET ERRN = 0
0030                      LET ERRL = 0
                          END
                    ;
0031                INDEX = CASS1AL1A:DEV_TMP0
0032                ASSIGN PDSABL = FAL0
0033                GOSUB FETCH
                    ;
0034                INDEX = CASS0AL1A:DEV_TMP0
0035                ASSIGN PDSABL = FAL20
0036                GOSUB FETCH
                    ;
0037                INDEX = CASS0AL2A:DEV_TMP0
0038                ASSIGN PDSABL = FAL40
0039                GOSUB FETCH
                    ;
0040                INDEX = CASS0AL2B:DEV_TMP0
0041                ASSIGN PDSABL = FAL60
0042                GOSUB FETCH
                    ;
0043                INDEX = CASS1AL2A:DEV_TMP0
0044                ASSIGN PDSABL = FAL80
0045                GOSUB FETCH
                    ;
0046                WAIT 2
0047                GOTO AGAIN
            ;
            ;       SUBROUTINES:
            ;
0048        FETCH:  BITVAL = 1
0049                FOR JLOOP = 0 TO 13
```

```
111  0050            PDSABL = AND(INDEX,BITVAL)
112  0051            BITVAL = BITVAL ( 1
113  0052            STEP PDSABL BY 1
114  0053            NEXT JLOOP
115  0054            RETURN
116          ;
117          ;       ALARM TYPER STRINGS:
118          ;
119  0056    ACKK:   IF ILOOP = 20 THEN MSSG0 = "START CASSBATCH PGM "
120  0058            IF ILOOP = 21 THEN MSSG0 = "CASSBATCH WRONG UNIT"
121  0060            IF ILOOP = 22 THEN MSSG0 = "TERMINATE CASSBATCH "
122                  ;
123                  ; If Startup, then print startup message:
124                  ;
125  0061            IF (FSTART) THEN
126                     BEGIN
127  0062               LET FSTART = 0
128  0063               PRINT ADI,"%p",MSSG0
129                     END
130  0064            RETURN
131          ;
132          ; If Error - print message, start timer, and set Error flag:
133          ;
134  0065    ERRSUB: IF ERRN != ERRNUM() THEN
135                     BEGIN
136  0066               TIMERON STIME0
137  0067               TIMERCLR STIME0
138  0068               LET FSPAR0 = 1
139  0069               LET ERRN = ERRNUM()
140  0070               LET ERRL = ERRLIN()
141  0071               PRINT ADI,"ERROR # %i AT LINE %i",errn,errl
142                     END
143  0072            WAIT 5
144  0073            ERSCLR
145  0074            GOTO AGAIN
146          ;
147          ; If FABORT = 1, then abort program:
148          ;
149          ; TERM:  LET FABORT = 0
150  0075    TERM:   LET MSSG0 = "CASSBATCH TERMINATED  "
151  0076            LET ILOOP = STA100 + 2                  ; ILoop = 22
152  0077            LET FSTART = 1
153  0078            LET MSSG0 = "TERM"
154  0079            GOSUB ACKK
155                  END

Total Errors Detected      0     Total Warnings      0
     Sequence Program Size = 1328 Bytes, 83 Pages
          9 Forward Reference(s)
```

TAG

```
Sequence Instruction Compiler Listing       15:36:54  24-Feb-95          Page  1

 1          ;       T A G . S E Q                                       VERSION 2
 2          ;
 3          ;
 4          ;       Function:       EPN ON/OFF for Graphics.
 5          ;
 6          ;       Written By      : R. N. Kersey
 7          ;       Date            : 05-31-92
 8          ;       Facility        : 242-A Evaporator Bldg.
 9          ;       Machine         : VAX 4104-A
10          ;       Op. System      : BAX/VMS 5.5-2AH
11          ;       D/3 Version     : 6.3-2 BASELINE
12          ;       Compiler        : SIC
13          ;       Build command   : @[320,20]TAG
14          ;
15          ;       UNIT = 0, SKDKEY = 62
16          ;
17          ;       Assoc. UPF:     [300,1]_TOTAL0.___      TOTAL0.LST
18          ;       Assoc. RPF:     [300,1]_ALL0.___        ALL0.LST
19          ;       SDB:            [300,1]PCM0ASDB.CMD     PCM0ASDB.LST
20          ;
21          ; MODIFICATION LOG:
22          ;
23          ;       29-JUL-93       Added comments and baselined for        M. Teats
24          ;                       D/3 version 3.3.
25          ;
26          ;       15-DEC-94       Revised header to indicate the new VMS          J. Berger
27          ;                       system. Rebase line document.
28          ;
29          ; END MOD LOG
30          ;
31          UNIT    WHOLE0
32    0001          IF(FDISP)THEN FDISP = 0
33    0002          ELSE FDISP = 1
34                  END

     Total Errors Detected      0     Total Warnings      0
          Sequence Program Size = 80 Bytes, 5 Pages
               0 Forward Reference(s)
```

TIMEINHA

```
Sequence Instruction Compiler Listing       15:36:55  24-Feb-95          Page  1

  1           ;       T I M E I N H A . S E Q                                 VERSION 2
  2           ;
  3           ;
  4           ;       Function:  Provide time for INHIBIT program.
  5           ;
  6           ;       Written By      : R. N. Kersey
  7           ;       Date            : 05-31-92
  8           ;       Facility        : 242-A Evaporator Bldg.
  9           ;       Machine         : VAX 4104-A
 10           ;       Op. System      : VAX/VMS 5.5-2AH
 11           ;       D/3 Version     : 6.3-2 BASELINE
 12           ;       Compiler        : SIC
 13           ;       Build command   : @[320,20]TIMEINHA
 14           ;
 15           ;       UNIT = 0, SKDKEY = 53
 16           ;
 17           ;       Assoc. UPF:     [300,1]_TOTAL0.___      TOTAL0.LST
 18           ;       Assoc. RPF:     [300,1]_ALL0.___        ALL0.LST
 19           ;       SDB:            [300,1]PCM0ASDB.CMD     PCM0ASDB.LST
 20           ;
 21           ;  MODIFICATION LOG:
 22           ;
 23           ;       29-JUL-93       Added comments and baselined for        M. Teats
 24           ;                       D/3 version 3.3.
 25           ;
 26           ;       15-DEC-94       Revised header to indicate the new VMS          J. Berger
 27           ;                       system. Rebase line document.
 28           ;
 29           ;  END MOD LOG
 30           ;
 31          UNIT    WHOLE0
 32    0000          FTRIP0 = 0
 33    0001          FTRIP1 = 0
 34    0002          FTRIP2 = 0
 35    0003          FTRIP3 = 0
 36    0004          FSTOP = 0
 37    0005          INT0 = TIMSTR(TIME0)
 38    0006          INT0 = DATSTR(DATE0)
 39                  END

      Total Errors Detected      0     Total Warnings      0
           Sequence Program Size = 144 Bytes, 9 Pages
                0 Forward Reference(s)
```

PGMOA

```
1           ;       P G M 0 A . S E Q                                       VERSION 2
2           ;
3           ;
4           ;       Program to monitor sequence programs running in PCM0
5           ;       PGMSTAT sets bits in an integer to give the status of a program.
6           ;       It indicates if the program is running, is critical, trapped etc.
7           ;       This program fetches this value then, by using a bit mask, it
8           ;       checks specific statuses and converts these to a single integer value
9           ;       used by the string editor.  SED translates the integer to strings
10          ;       in graphic 043, titled "PROGRAM STATUS".  SED = String Editor for graphics
11          ;
12          ;       Written By      : R. N. Kersey
13          ;       Date            : 05-31-92
14          ;       Facility        : 242-A Evaporator Bldg.
15          ;       Machine         : VAX 4104-A
16          ;       Op. System      : VAX/VMS 5.5-2AH
17          ;       D/3 Version     : 6.3 BASELINE
18          ;       Compiler        : SIC
19          ;       Build command   : @[320,20]PGM0A
20          ;
21          ;       Assoc. UPF:     [300,1]_BLANKK.___      BLANKK.LST
22          ;       Assoc. RPF:     [300,1]_EMTY.___        EMTY.LST
23          ;       SDB:            [300,1]PCM0ASDB.CMD     PCM0ASDB.LST
24          ;
25          ; MODIFICATION LOG:
26          ;
27          ;       29-JUL-93       Added comments and baselined for        M. Teats
28          ;                       D/3 version 3.3.
29          ;
30          ;       15-DEC-94       Revised header to indicate the new VMS          J. Berger
31          ;                       system. Rebaselined document. Removed
32          ;                       obsolite Select() statements.
33          ;
34          ; END MOD LOG
35          ;
36          UNIT    BLANK0
37                  COMMON ROOMS
38          ;
39              .   PSTAT == PALARM
40                  PGRAPHIC == PSPAR0
41  0000            FABORT = 0
42          ;
43  0001            ASSIGN PGRAPHIC = INT0
44  0002            FOR ILOOP = 0 TO 39
45  0003            PGRAPHIC = 0
46  0004            NEXT ILOOP
47          ;
48  0005            TIMERON STMR1
49  0006            ON ERROR GOSUB ERRSUB
50          ;
51  0007    AGAIN:  MARK STEP "WTUNTL",0,0
52  0008            LSG6 = "WAIT"
53  0009            WAIT 1
54          ;       IF FABORT THEN GOTO TERM
55  0010            LSG6 = "RUNNING"
```

```
 56   0011            WAIT 1
 57   0012            IF FSPAR5 THEN
 58                      BEGIN
 59   0013               IF STMR0 > 60 THEN
 60                         BEGIN
 61   0014                  FSPAR5 = 0
 62   0015                  TIMERCLR STMR0
 63   0016                  TIMEROFF STMR0
 64                         END
 65                      END
 66   0017            TIMERCLR STMR1
 67            ;
 68            ;       SEQUENCE PROGRAM STATUS FOLLOW:
 69            ;
 70   0018            INT20 = PGMSTAT("244BATCH","A244")
 71   0019            INT21 = PGMSTAT("244HVAC","HVC244")
 72   0020            INT22 = PGMSTAT("WSTXFR","WASTE")
 73   0021            INT23 = PGMSTAT("350BATCH","A350")
 74   0022            INT24 = PGMSTAT("FEEDBATCH","FEED")
 75   0023            INT25 = PGMSTAT("CA1BATCH","CA1")
 76   0024            INT26 = PGMSTAT("PB1BATCH","PB1")
 77   0025            INT27 = PGMSTAT("PB2BATCH","PB2")
 78   0026            INT28 = PGMSTAT("EA1BATCH","EA1")
 79   0027            INT29 = PGMSTAT("EC1BATCH","EC1")
 80   0028            INT30 = PGMSTAT("RWBATCH","RWATER")
 81   0029            INT31 = PGMSTAT("URWBATCH","URW")
 82   0030            INT32 = PGMSTAT("VENTBATCH","VVENT")
 83   0031            INT33 = PGMSTAT("PCBATCH","PC")
 84   0032            INT34 = PGMSTAT("IXBATCH","IX")
 85   0033            INT35 = PGMSTAT("SCBATCH","SC")
 86   0034            INT36 = PGMSTAT("BASINBAT","BASINS")
 87   0035            INT37 = PGMSTAT("UTILBATCH","UTIL")
 88   0036            INT38 = PGMSTAT("AMUBATCH","AMU")
 89   0037            INT39 = PGMSTAT("LERFBATCH","LERF")
 90            ;
 91            ;                  PGRAPHIC                    PSTAT
 92            ;                  --------------              --------------------
 93            ;                  0 = _____
 94            ;                  1 = RUNNING                 1       000000000001
 95            ;                  2 = TRAPPED                 2       000000000010
 96            ;                  3 = CRITICAL                4       000000000100
 97            ;                  4 = WAIT                    8       000000001000
 98            ;                  5 = WAITUNTIL               4096    100000000000
 99            ;                  6 = NOT RUNNING             0       000000000000
100            ;                  7 = NO STATUS
101            ;
102   0038            ASSIGN PGRAPHIC = INT0
103   0039            ASSIGN PSTAT = INT20
104   0040            TRIP = 0                                        ; Initialize TRIP
105   0041            FTRIP1 = 0
106            ;
107   0042            FOR ILOOP = 0 TO 19
108   0043            BVAL = 1                                        ; 1 = 1 = RUNNING
109   0044            REMAIN = 0
110            ;
```

```
111   0045          FOR INDEX = 1 TO 4                          ; 2 = 2 = TRAPPED
112   0047          IF AND(BVAL,PSTAT)THEN REMAIN = INDEX       ; 4 = 3 = CRITICAL
113   0048          BVAL = BVAL ( 1                             ; 8 = 4 = WAIT
114   0049          NEXT INDEX
115           ;
116   0050          IF PSTAT = -1 THEN                          ; -1 = 6 = NOTRUNNING
117                    BEGIN
118   0051             FTRIP1 = 1
119   0052             PGRAPHIC = 6
120                    END
121                 ELSE
122                    BEGIN
123   0053             PGRAPHIC = REMAIN                        ; set SED value
124   0055             IF AND(PSTAT,4096)THEN PGRAPHIC = 5      ; 4096 = 5 = WAITUNTIL
125                    END
126   0056          TRIP = OR(TRIP,PSTAT)                       ; Accumulate for all pgms
127   0057          STEP PGRAPHIC BY 1
128   0058          STEP PSTAT BY 1
129   0059          NEXT ILOOP
130           ;
131           ;
132           ;              TRIP   = TRAPPED
133           ;              FTRIP1 = NOTRUNNING
134           ;              FTRIP2 = FLAG FROM DOGO
135           ;              FTRIP4 = FAULT
136           ;
137
138                   BEGIN
139   0061            IF AND(AUDALARM0:DEV_DFLG,1)THEN FTRIP4 = 1   ; IF AUDALARM0 IN FAULT
140   0062              ELSE FTRIP4 = 0
141   0063            FTRIP0 = OR(FTRIP1,FTRIP4)                    ; IF FAULT OR NOTRUNNING
142   0064            FSPAR0 = AND(TRIP,2)                          ; IF ONE PGM TRAPPED
143   0065            FTRIP0 = OR(FTRIP0,FSPAR0)                    ; TOTAL ALL
144   0067            IF OR(FTRIP0,FTRIP2)THEN LAMP 0,48,3
145   0068               ELSE LAMP 0,48,0
146                   END
147   0069          GOTO AGAIN
148           ;
149           ;
150           ;
151           ;=========================================================================
152           ;              SUBROUTINES
153           ;
154           ERRSUB:
155           ;
156           ; errors are handled differently depending on their severity
157           ;
158           ;  19    PROGRAM ALREADY IN KICK SUBROUTINE (rekick)
159           ;  32    UNDEFINED COMMUNICATIONS PACKET TYPE (ignore)
160           ;  41   PCM DB HEADER NOT LOADED  (ignore)
161           ;  69   IPN NOT DEFINED (ignore)
162           ;  162  NO SUCH PROGRAM IN THIS PCM (ignore)
163           ;  165   SEQUENCE PROGRAM ALREADY RUNNING (ignore)
164           ;  170   SERIAL NUMBER MISMATCH (wait forever)
165           ;  179  ILLEGAL SUBROUTINE STACK CLEAR (ignore)
```

```
166             ;  187   BACKUP PCM IS DOWN (ignore)
167             ;  193   NO SUCH PROGRAM IN THIS PCM (ignore)
168             ;  198   ILLEGAL SUBROUTINE STACK NESTING ERROR (ignore)
169             ;  199   ILLEGAL RDF NAME (wait forever)
170             ;  200   RDF NOT LOADED IN MEMORY (wait forever)
171             ;  201  RECIPE LOAD NOT ENABLED (wait forever)
172             ;  222  BOARDS DEAD (ignore)
173             ;  226   BOARD ERROR E.G. broken skid panel etc...  (ignore)
174             ;  245   BOTH PCMS DOWN, SENT REDUNDANT HIGHWAY ACCESS (ignore)
175             ;
176     0070            ERRN = ABS(ERRNUM())     ; get error number
177     0071            ERRL = ERRLIN()          ; get error line number
178     0072            TIMERON STMR0
179     0073            TIMERCLR STMR0
180     0074             FSPAR5 = 1
181             ;
182             ; if error is not serious, clear and return to program
183             ;
184     0075            IF (ERRN=187) ¦ (ERRN=32) ¦ (ERRN=69) ¦ (ERRN=222) THEN
185     0076                ERSRET
186
187     0077            IF (ERRN=41) ¦ (ERRN=165) ¦ (ERRN=162) ¦ (ERRN=179) THEN
188     0078                ERSRET
189
190     0079            IF (ERRN=226) ¦ (ERRN=198) ¦ (ERRN=245) THEN
191     0080                ERSRET
192
193     0081            IF (ERRN=193) ¦ (ERRN=19) THEN
194     0082                ERSRET
195
196             ; else report the error, and wait for corrective action ....
197     0083            PRINT 0,"ERROR NUMBER %i AT LINE %i ***",ERRN,ERRL
198
199     0084            ERSCLR
200     0085            GOTO AGAIN                       ; when corrective action taken,
201                                                      ; goto done and cleanup
202             ;
203             ;=============================================================================
204             ;
205             ;
206     0086    TERM:   ASSIGN PGRAPHIC = INT1
207     0087            FOR ILOOP = 1 TO 20
208     0088            PGRAPHIC = 7
209     0089            STEP PGRAPHIC BY 1
210     0090            NEXT ILOOP
211                     END

      Total Errors Detected      0     Total Warnings      0
          Sequence Program Size = 2032 Bytes, 127 Pages
                1 Forward Reference(s)
```

WHITE1

```
1           ;       W H I T E 1 . S E Q                                     VERSION 8
2           ;
3           ;       Function:       Drives WHITE ALARM SUMMARY PAGES,
4           ;                       Graphic numbers 100 - 102.
5           ;
6           ;       Written By      : R. N. Kersey
7           ;       Date            : 05-31-92
8           ;       Facility        : 242-A Evaporator Bldg.
9           ;       Machine         : VAX 4104-A
10          ;       Op. System      : VAX/VMS 5.5-2AH
11          ;       D/3 Version     : 6.3-2 BASELINE
12          ;       Compiler        : SIC
13          ;       Build command   : @[320,20]WHITE1
14          ;
15          ;       UNIT = 0, SKDKEY = 60
16          ;
17          ;       Assoc. UPF:     [300,1]_TOTAL1.___      TOTAL1.LST
18          ;       Assoc. RPF:     [300,1]_ALL1.___        ALL1.LST
19          ;       SDB:            [300,1]PCM1ASDB.___     PCM1ASDB.CMD
20          ;
21          ;       POINTER   SDB VAR.      FUNCTION
22          ;       -------   ---------     ------------------------------------------
23          ;       PTYPE     FDSn          Alarm inhibit status
24          ;       PREV      FRVn          Alarm condition or setpoint
25          ;       PSTAT     FXn           Current signal status
26          ;       PALARM    FALn          Previous alarm condition
27          ;
28          ; MODIFICATION LOG:
29          ;
30          ;       12-SEP-92       Changed LI-C100-2 to LI-C100-1 and alarm  M. Teats
31          ;                       level from 0.95 to 0.98
32          ;
33          ;       16-SEP-92       Deleted DDI-CA1-1 alarms (96+97)          M. Teats
34          ;
35          ;       02-DEC-92       - Corrected AMU strings                         M. Teats
36          ;                       - Modified PI-RW-2 alarms
37          ;                       - Added comments
38          ;
39          ;       09-DEC-92       Changed FIC-EC3-1 LO alarm from 140 to 50       M. Teats
40          ;
41          ;       29-DEC-92       Corrected white alarm level on II-AS-1          M. Teats
42          ;
43          ;       13-FEB-93       Added LERF gate white alarms:                   M. Teats
44          ;                               ZS-105PS1       (171)
45          ;                               ZS-105PS2       (172)
46          ;
47          ;       01-APR-93       Changed FRV128 (YS-A-C100) and FRV129           M. Teats
48          ;                       (YS-P-C100) from 1 to 0 so the alarm occurs
49          ;                       on shutdown.  Also changed strings for labels
50          ;                       S128: and S129: from "...CONFIRM ON " to
51          ;                       "...SHUTDOWN ".
52          ;
53          ;       04-APR-93       Reinstated white alarms on AW and AN pump       M. Teats
54          ;                       pump shutdown (YS-) signals, and on AP pump
55          ;                       MCC status (YS-) signals.
```

```
56              ;
57              ;-mct1+  13-MAY-93        Remove AW pump shutdown DEV white alarms,        M. Teats
58              ;                         which are based on the analog signals,
59              ;                         since they are not needed due to the 4/4/93
60              ;                         mods.  See CCN# 93MAY13-01.
61              ;
62              ;-mct2+  13-MAY-93        Remove AN pump shutdown DEV white alarms,        M. Teats
63              ;                         which are based on the analog signals,
64              ;                         since they are not needed due to the 4/4/93
65              ;                         mods.  See CCN# 93MAY13-01.
66              ;
67              ;-mct3+  14-MAY-93        Corrected typo on FDS36 and FX36 from            M. Teats
68              ;                         PSH-APFL1 to FXS-DR712.  Added FRV11, 17,
69              ;                         and 18 = 1 to get "MCC OFF" alarm for AP
70              ;                         pumps 103, 108, and 102M.  See CCN# 93MAY14-01.
71              ;
72              ;                 Summary To Date Of Deletions:
73              ;                    Deleted Points:  47-50, 80-83
74              ;                    Deleted alarms:  85-89, 93, 95, 96-97, 147, 118,
75              ;                         120, 122-125, 127, 134, 137, 141-142
76              ;                    Deleted DEV white alarms:  53-59 (AN pumps) -mct2-
77              ;                    Deleted DEV white alarms:  70-75 (AW pumps) -mct1-
78              ;
79              ;-mct4+ 19-MAY-93         Added YS-P-42, YS-P-43, YS-P-44 (173-175)        M. Teats
80              ;                         per CCN# 93MAY19-02.
81              ;
82              ;        10-JUN-93        Added comments and baselined for         M. Teats
83              ;                         D/3 version 3.3.  CCN# 93JUN01-01.
84              ;
85              ;-mct5+ 12-JAN-94         Deleted YS-AW102 per CCN# 94JAN12-01.            M. Teats
86              ;                         and deleted II-P-C100
87              ;
88              ;        01-AUG-94        Added White Alarm for PDI-CA1-3 per              J.F. Berger
89              ;                         EE-94-00019.
90              ;
91              ;        15-DEC-94        Revised header to indicate the new VMS           J. Berger
92              ;                         system. Rebaselined document. Removed
93              ;                         obsolite Select() statement.
94              ;
95              ;- END MOD LOG
96              ;
97              UNIT     WHOLE1
98                       COMMON WHOLE0
99              ;
100                      NXTGRA == PDATE
101                      NXTMSG == PTIME
102                      NXTINT == PEPN
103                      NXTLSG == PALM
104                      PLRGSG == PTRIP
105                      STPPER == WHITE
106             ;
107     0000             ASSIGN PGRAFX = GRAFX0
108     0001             ASSIGN PMESSG = MSSG0
109     0002             ASSIGN PDSABL = LSG0
110     0003             ASSIGN PLINE = LINE0
```

```
111   0004          FOR ILOOP = 0 TO 46
112   0005           PGRAFX = 0
113   0006           PMESSG = "  "
114   0007           PDSABL = "  "
115   0008           PLINE =  0
116   0009          STEP PGRAFX BY 1
117   0010          STEP PMESSG BY 1
118   0011          STEP PDSABL BY 1
119   0012          STEP PLINE  BY 1
120   0013          NEXT ILOOP
121         ;
122   0014          ERRN = 0
123   0015          ERRL = 0
124         ;
125   0016          LSG47 = " "
126   0017          MSSG47 = " "
127   0018          INT47 = 0
128   0019          GRAF47 = 0
129         ;
130         ;       B E G I N   I N I T I A L I Z I N G :
131         ;
132   0020          ASSIGN PSTAT = FX1
133   0021          ASSIGN PALARM = FAL1
134   0022          ASSIGN PREV = FRV1
135   0023          ASSIGN PTYPE = FDS1
136   0024          ASSIGN PLINE = INT1
137   0025          FOR ILOOP = 1 TO 175            ; +mct4-
138   0026           PSTAT =  0
139   0027           PALARM = 0
140   0028           PREV =   0
141   0029           PTYPE =  1
142   0030           PLINE = 0
143   0031          STEP PSTAT BY 1
144   0032          STEP PALARM BY 1
145   0033          STEP PREV BY 1
146   0034          STEP PTYPE BY 1
147   0035          STEP PLINE BY 1
148   0036          NEXT ILOOP
149         ;
150   0037          NOP
151         ;
152         ;       FRV8  = 1
153   0038          FRV9  = 1       ; YS-AP-101
154   0039          FRV10 = 1       ; YS-AP-102
155   0040          FRV11 = 1       ; YS-AP-103 -mct3+
156   0041          FRV12 = 0
157   0042          FRV13 = 1       ; YS-AP-104
158   0043          FRV14 = 1       ; YS-AP-105
159   0044          FRV15 = 1       ; YS-AP-106
160   0045          FRV16 = 1       ; YS-AP-107
161   0046          FRV17 = 1       ; YS-AP-108 -mct3+
162   0047          FRV18 = 1       ; YS-AP102M -mct3+
163   0048          FRV39 = 0
164                 ;
165                 ;  deleted YS-AY-101, -102, and YS-AZ-101, -102: 47-50
```

```
166                     ;
167             ;-mct2-  FRV53 = 1
168             ;        FRV54 = 1
169             ;        FRV55 = 1
170             ;        FRV56 = 1
171             ;        FRV57 = 1
172             ;        FRV58 = 1
173             ;-mct2-  FRV59 = 1
174     0049             FRV60 = 1
175     0050             FRV61 = 1
176     0051             FRV62 = 1
177     0052             FRV63 = 1
178     0053             FRV64 = 1
179     0054             FRV65 = 1
180     0055             FRV66 = 1
181     0056             FRV67 = 1
182     0057             FRV68 = 1
183     0058             FRV69 = 1
184             ;-mct1-  FRV70 = 1
185             ;        FRV71 = 1
186             ;        FRV72 = 1
187             ;        FRV73 = 1
188             ;        FRV74 = 1
189             ;-mct1-  FRV75 = 1
190     0059             FRV76 = 1
191     0060             FRV77 = 1
192     0061             FRV78 = 1
193     0062             FRV79 = 1
194     0063             ASSIGN PREV = FRV84
195     0064             FOR ILOOP = 84 TO 126
196     0065             PREV = 1
197     0066             STEP PREV BY 1
198     0067             NEXT ILOOP
199     0068             FRV100 = 1
200     0069             FRV102 = 0
201     0070             FRV103 = 1
202     0071             FRV104 = 1
203     0072             FRV112 = 0
204     0073             FRV113 = 0
205     0074             FRV114 = 0
206     0075             FRV117 = 1
207     0076             FRV119 = 1
208     0077             FRV121 = 1
209     0078             FRV128 = 0
210     0079             FRV129 = 0
211     0080             FRV130 = 1
212     0081             FRV133 = 0
213     0082             FRV135 = 1
214             ;        FRV136 = 1              ; commented out, -mct5+
215     0083             FRV137 = 1
216             ;        FRV140 = 1
217     0084             ASSIGN PREV = FRV145
218     0085             FOR ILOOP = 145 TO 155
219     0086             PREV = 1
220     0087             STEP PREV BY 1
```

```
221   0088          NEXT ILOOP
222   0089          FRV143 = 0              ; YS-CPE1-1
223   0090          FRV144 = 0              ; YS-CPE2-1
224   0091          FRV156 = 1              ; YS-A-E101
225   0092          FRV157 = 1              ; YS-P-E101
226   0093          FRV158 = 1              ; YS-A-E102
227   0094          FRV159 = 1              ; YS-P-E102
228   0095          FRV160 = 1              ; YS-A-E104
229   0096          FRV161 = 1              ; YS-P-E104
230   0097          FRV162 = 1
231   0098          FRV163 = 1              ; TI-E101-2
232   0099          FRV164 = 1              ; WFI-E101 LOLO
233   0100          FRV165 = 1              ; WFI-E101 LO
234   0101          FRV166 = 1              ; WFI-E102 LOLO
235   0102          FRV167 = 1              ; WFI-E102 LO
236   0103          FRV168 = 1              ; TI-E104-2 HI
237   0104          FRV169 = 1              ; WFI-E104 LOLO
238   0105          FRV170 = 1              ; WFI-E104 LO
239   0106          FRV173 = 1              ; YS-P-42  +mct4-
240   0107          FRV174 = 1              ; YS-P-43  +mct4-
241   0108          FRV175 = 1              ; YS-P-44  +mct4-
242   0109          FTRIP0 = 0
243   0110          ON ERROR GOSUB ERRSUB
244   0111          IN4 = 0
245   0112          ASSIGN PGRAFX = FSPAR0
246   0113          ASSIGN PMESSG = FSPAR1
247   0114          ASSIGN PDSABL = FSPAR2
248   0115          ASSIGN PLINE =  FSPAR3
249               ;
250               ;       E N D   I N I T I A L I Z I N G
251               ;
252               ;       B E G I N   P R O G R A M   L O O P :
253               ;
254   0116    AGAIN:  WAIT 3                  ; Go get alarm status for all
255               ;   WAITUNTIL SELECT()      ; Wait until micro selected
256   0117          IF (FSPAR0) THEN          ; If alarm flag is set
257   0118             IF STMR0 > 60 THEN     ; If alarm flag active for 1 minute
258                      BEGIN                ; Reset timer and turn off flag
259   0119               TIMEROFF STMR0
260   0120               TIMERCLR STMR0
261   0121               FSPAR0 = 0
262   0122               ERRN = 0
263   0123               ERRL = 0
264                      END
265   0124          SORT = PGMSTAT("PGM1A","BLANK1")
266   0125          REMAIN = 0
267   0126          BITVAL = 1
268   0127           FOR INDEX = 1 TO 4
269   0129           IF AND(BITVAL,SORT) THEN REMAIN = INDEX
270   0130           BITVAL = BITVAL ( 1
271   0131           NEXT INDEX
272   0133          IF SORT = -1 THEN IN0 = 6
273                  ELSE
274                    BEGIN
275   0134             IN0 = REMAIN
```

```
276   0136              IF AND(SORT,4096) THEN INO = 5
277                     END
278   0137            IF FTRIP0 THEN
279                     BEGIN
280   0138              LAMP ADI,60,0
281   0139              FTRIP0 = 0
282                     END
283                   ;
284                   ;  DETERMINE INHIBITED STATUS, (PTYPE = FDSn = 1 IF INHA):
285                   ;
286   0140            FDS1 = AND('YS-AN-101':DIN_ALMCT,16)      ; 1
287   0141            FDS2 = AND('YS-AN-102':DIN_ALMCT,16)      ; 2
288   0142            FDS3 = AND('YS-AN-103':DIN_ALMCT,16)      ; 3
289   0143            FDS4 = AND('YS-AN-104':DIN_ALMCT,16)      ; 4
290   0144            FDS5 = AND('YS-AN-105':DIN_ALMCT,16)      ; 5
291   0145            FDS6 = AND('YS-AN-106':DIN_ALMCT,16)      ; 6
292   0146            FDS7 = AND('YS-AN-107':DIN_ALMCT,16)      ; 7
293   0147            FDS8 = AND('TXS-AN-2':DIN_ALMCT,16)       ; 8
294   0148            FDS9 = AND('YS-AP-101':DIN_ALMCT,16)      ; 9
295   0149            FDS10 = AND('YS-AP-102':DIN_ALMCT,16)     ; 10
296   0150            FDS11 = AND('YS-AP-103':DIN_ALMCT,16)     ; 11
297   0151            FDS12 = AND('LDS-AP03D':DIN_ALMCT,16)     ; 12
298   0152            FDS13 = AND('YS-AP-104':DIN_ALMCT,16)     ; 13
299   0153            FDS14 = AND('YS-AP-105':DIN_ALMCT,16)     ; 14
300   0154            FDS15 = AND('YS-AP-106':DIN_ALMCT,16)     ; 15
301   0155            FDS16 = AND('YS-AP-107':DIN_ALMCT,16)     ; 16
302   0156            FDS17 = AND('YS-AP-108':DIN_ALMCT,16)     ; 17
303   0157            FDS18 = AND('YS-AP102M':DIN_ALMCT,16)     ; 18
304   0158            FDS19 = AND('PSHAP102M':DIN_ALMCT,16)     ; 19
305   0159            FDS20 = AND('PSLAP102M':DIN_ALMCT,16)     ; 20
306   0160            FDS21 = AND('YS-AP102R':DIN_ALMCT,16)     ; 21
307   0161            FDS22 = AND('WSHAP102M':DIN_ALMCT,16)     ; 22
308   0162            FDS23 = AND('TSHAP102M':DIN_ALMCT,16)     ; 23
309   0163            FDS24 = AND('VSHAP102M':DIN_ALMCT,16)     ; 24
310   0164            FDS25 = AND('TXS-AP-2':DIN_ALMCT,16)      ; 25
311   0165            FDS26 = AND('FSLAP2711':DIN_ALMCT,16)     ; 26
312   0166            FDS27 = AND('PSH-APFP1':DIN_ALMCT,16)     ; 27
313   0167            FDS28 = AND('ZS-AP-FP1':DIN_ALMCT,16)     ; 28
314   0168            FDS29 = AND('FSL-APK13':DIN_ALMCT,16)     ; 29
315   0169            FDS30 = AND('FSL-APK12':DIN_ALMCT,16)     ; 30
316   0170            FDS31 = AND('FSL-APK11':DIN_ALMCT,16)     ; 31
317   0171            FDS32 = AND('FSL-APK23':DIN_ALMCT,16)     ; 32
318   0172            FDS33 = AND('FSL-APK22':DIN_ALMCT,16)     ; 33
319   0173            FDS34 = AND('FSL-APK21':DIN_ALMCT,16)     ; 34
320   0174            FDS35 = AND('FSH-DR712':DIN_ALMCT,16)     ; 35
321   0175            FDS36 = AND('FXS-DR712':DIN_ALMCT,16)   ; 36 -mct3+
322   0176            FDS37 = AND('PSH-APFL1':DIN_ALMCT,16)     ; 37
323   0177            FDS38 = AND('ZS-AP-DR1':DIN_ALMCT,16)     ; 38
324   0178            FDS39 = AND('ZS-AP-FL1':DIN_ALMCT,16)     ; 39
325   0179            FDS40 = AND('YS-AW-101':DIN_ALMCT,16)     ; 40
326   0180            FDS41 = AND('YS-AW102':DIN_ALMCT,16)      ; 41
327   0181            FDS42 = AND('YS-AW-103':DIN_ALMCT,16)     ; 42
328   0182            FDS43 = AND('YS-AW-104':DIN_ALMCT,16)     ; 43
329   0183            FDS44 = AND('YS-AW-105':DIN_ALMCT,16)     ; 44
330   0184            FDS45 = AND('YS-AW-106':DIN_ALMCT,16)     ; 45
```

```
331   0185            FDS46 = AND('TXS-AW-2':DIN_ALMCT,16)      ; 46
332                   ;
333                   ;  deleted YS-AY-101, -102, and YS-AZ-101, -102: 47-50
334                   ;
335   0186            FDS51  = AND('LSH-AX-1':DIN_ALMCT,16)     ; 51
336   0187            FDS52  = AND('LSL-AX-1':DIN_ALMCT,16)     ; 52
337           ;-mct2- FDS53  = AND('P-AN-101':DEV_ALMCT,16)     ; 53
338           ;       FDS54  = AND('P-AN-102':DEV_ALMCT,16)     ; 54
339           ;       FDS55  = AND('P-AN-103':DEV_ALMCT,16)     ; 55
340           ;       FDS56  = AND('P-AN-104':DEV_ALMCT,16)     ; 56
341           ;       FDS57  = AND('P-AN-105':DEV_ALMCT,16)     ; 57
342           ;       FDS58  = AND('P-AN-106':DEV_ALMCT,16)     ; 58
343           ;-mct2- FDS59  = AND('P-AN-107':DEV_ALMCT,16)     ; 59
344   0188            FDS60  = AND('P-AP-101':DEV_ALMCT,16)     ; 60
345   0189            FDS61  = AND('P-AP-102':DEV_ALMCT,16)     ; 61
346   0190            FDS62  = AND('P-AP-103':DEV_ALMCT,16)     ; 62
347   0191            FDS63  = AND('P-AP-104':DEV_ALMCT,16)     ; 63
348   0192            FDS64  = AND('P-AP-105':DEV_ALMCT,16)     ; 64
349   0193            FDS65  = AND('P-AP-106':DEV_ALMCT,16)     ; 65
350   0194            FDS66  = AND('P-AP-107':DEV_ALMCT,16)     ; 66
351   0195            FDS67  = AND('P-AP-108':DEV_ALMCT,16)     ; 67
352   0196            FDS68  = AND('P-AP-102M':DEV_ALMCT,16)    ; 68
353   0197            FDS69  = AND('P-AP-102R':DEV_ALMCT,16)    ; 69
354           ;-mct1- FDS70  = AND('P-AW-101':DEV_ALMCT,16)     ; 70
355           ;       FDS71  = AND('P-AW-102':DEV_ALMCT,16)     ; 71
356           ;       FDS72  = AND('P-AW-103':DEV_ALMCT,16)     ; 72
357           ;       FDS73  = AND('P-AW-104':DEV_ALMCT,16)     ; 73
358           ;       FDS74  = AND('P-AW-105':DEV_ALMCT,16)     ; 74
359           ;-mct1- FDS75  = AND('P-AW-106':DEV_ALMCT,16)     ; 75
360   0198            FDS76  = AND('P-AY-101':DEV_ALMCT,16)     ; 76
361   0199            FDS77  = AND('P-AY-102':DEV_ALMCT,16)     ; 77
362   0200            FDS78  = AND('P-AZ-101':DEV_ALMCT,16)     ; 78
363   0201            FDS79  = AND('P-AZ-102':DEV_ALMCT,16)     ; 79
364   0202            FDS84  = AND('PSH-FP':DIN_ALMCT,16)       ; 84
365   0203            FDS90  = AND('TI-244-TK':AI_ALMCT,16)     ; 90
366   0204            FDS91  = AND('FI-244EXH':AI_ALMCT,16)     ; 91
367   0205            FDS92  = AND('FI-244EXH':AI_ALMCT,16)     ; 92
368   0206            FDS94  = AND('II-AW-102':AI_ALMCT,16)     ; 94
369           ;       FDS96  = AND('DDI-CA1-1':AI_ALMCT,16)     ; 96
370           ;       FDS97  = AND('DDI-CA1-1':AI_ALMCT,16)     ; 97
371   0207            FDS98  = AND('PDI-FH1-1':AI_ALMCT,16)     ; 98
372   0208            FDS99  = AND('PDI-FH2-1':AI_ALMCT,16)     ; 99
373   0209            FDS100 = AND('PDI-FH3-1':AI_ALMCT,16)     ; 100
374   0210            FDS101 = AND('SIC-PB2-1':AI_ALMCT,16)     ; 101
375   0211            FDS102 = AND('PSL-EA1-1':DIN_ALMCT,16)    ; 102
376   0212            FDS103 = AND('PI-EA1-13':AI_ALMCT,16)     ; 103
377   0213            FDS104 = AND('PI-EA1-13':AI_ALMCT,16)     ; 104
378   0214            FDS105 = AND('FIC-EA1-1':AI_ALMCT,16)     ; 105
379   0215            FDS106 = AND('PI-EA1-1':AI_ALMCT,16)      ; 106
380   0216            FDS107 = AND('PI-EA1-1':AI_ALMCT,16)      ; 107
381   0217            FDS108 = AND('TI-EA1-5':AI_ALMCT,16)      ; 108
382   0218            FDS109 = AND('PI-EC1-5':AI_ALMCT,16)      ; 109
383   0219            FDS110 = AND('FIC-EC3-1':AI_ALMCT,16)     ; 110
384   0220            FDS111 = AND('TIC-EC1-2':AI_ALMCT,16)     ; 111
385   0221            FDS112 = AND('LDS-BFP-1':DIN_ALMCT,16)    ; 112
```

```
386   0222          FDS113 = AND('LSL-RWSMP':DIN_ALMCT,16)    ; 113
387   0223          FDS114 = AND('LDS-BFP-2':DIN_ALMCT,16)    ; 114
388   0224          FDS115 = AND('PDI-RW-1':AI_ALMCT,16)      ; 115
389   0225          FDS116 = AND('PDI-RW-2':AI_ALMCT,16)      ; 116
390   0226          FDS117 = AND('PI-RW-2':AI_ALMCT,16)       ; 117
391         ;       FDS118 = AND('PI-RW-2':AI_ALMCT,16)       ; 118
392   0227          FDS119 = AND('FIC-EC1-1':AI_ALMCT,16)     ; 119
393         ;       FDS120 = AND('FIC-EC1-1':AI_ALMCT,16)     ; 120
394   0228          FDS121 = AND('II-EXC-1':AI_ALMCT,16)      ; 121
395   0229          FDS126 = AND('II-AS-1':AI_ALMCT,16)       ; 126
396   0230          FDS128 = AND('YS-A-C100':DIN_ALMCT,16)    ; 128
397   0231          FDS129 = AND('YS-P-C100':DIN_ALMCT,16)    ; 129
398   0232          FDS130 = AND('ZS-IDX-2':DIN_ALMCT,16)     ; 130
399   0233          FDS131 = AND('PSH-ILF-1':DIN_ALMCT,16)    ; 131
400   0234          FDS132 = AND('ZS-RC3-4':DIN_ALMCT,16)     ; 132
401   0235          FDS133 = AND('ZS-RC3-5':DIN_ALMCT,16)     ; 133
402   0236          FDS135 = AND('LI-C100-1':AI_ALMCT,16)     ; 135
403         ;       FDS136 = AND('II-P-C100':AI_ALMCT,16)     ; 136, commented out, -mct5+
404   0237          FDS137 = AND('PDI-CA1-3':AI_ALMCT,16)     ; 137
405   0238          FDS138 = AND('LSL-SC-1':DIN_ALMCT,16)     ; 138
406   0239          FDS139 = AND('LSL-SC-2':DIN_ALMCT,16)     ; 139
407   0240          FDS140 = AND('LSL-SC-3':DIN_ALMCT,16)     ; 140
408   0241          FDS143 = AND('YS-CPE1-1':DIN_ALMCT,16)    ; 143
409   0242          FDS144 = AND('YS-CPE2-1':DIN_ALMCT,16)    ; 144
410   0243          FDS145 = AND('PI-IA-1':AI_ALMCT,16)       ; 145
411   0244          FDS146 = AND('PI-IA-1':AI_ALMCT,16)       ; 146
412         ;       FDS147 = AND('PI-PA-1':AI_ALMCT,16)       ; 147
413   0245          FDS148 = AND('PI-PA-1':AI_ALMCT,16)       ; 148
414   0246          FDS149 = AND('TI-AMBS-1':AI_ALMCT,16)     ; 149
415   0247          FDS150 = AND('TI-AMBS-1':AI_ALMCT,16)     ; 150
416   0248          FDS151 = AND('TI-AMBS-1':AI_ALMCT,16)     ; 151
417   0249          FDS152 = AND('TI-AMBS-1':AI_ALMCT,16)     ; 152
418   0250          FDS153 = AND('TI-AMBS-2':AI_ALMCT,16)     ; 153
419   0251          FDS154 = AND('TI-AMBS-2':AI_ALMCT,16)     ; 154
420   0252          FDS155 = AND('TI-AMBS-2':AI_ALMCT,16)     ; 155
421   0253          FDS156 = AND('TI-AMBS-2':AI_ALMCT,16)     ; 156
422   0254          FDS157 = AND('YS-A-E101':DIN_ALMCT,16)    ; 157
423   0255          FDS158 = AND('YS-P-E101':DIN_ALMCT,16)    ; 158
424   0256          FDS159 = AND('YS-A-E102':DIN_ALMCT,16)    ; 159
425   0257          FDS160 = AND('YS-P-E102':DIN_ALMCT,16)    ; 160
426   0258          FDS161 = AND('YS-A-E104':DIN_ALMCT,16)    ; 161
427   0259          FDS162 = AND('YS-P-E104':DIN_ALMCT,16)    ; 162
428   0260          FDS163 = AND('TI-E101-2':AI_ALMCT,16)     ; 163
429   0261          FDS164 = AND('WFI-E101':AI_ALMCT,16)      ; 164
430   0262          FDS165 = AND('WFI-E101':AI_ALMCT,16)      ; 165
431   0263          FDS166 = AND('WFI-E102':AI_ALMCT,16)      ; 166
432   0264          FDS167 = AND('WFI-E102':AI_ALMCT,16)      ; 167
433   0265          FDS168 = AND('TI-E104-2':AI_ALMCT,16)     ; 168
434   0266          FDS169 = AND('WFI-E104':AI_ALMCT,16)      ; 169
435   0267          FDS170 = AND('WFI-E104':AI_ALMCT,16)      ; 170
436   0268          FDS171 = AND('ZS-105PS1':DIN_ALMCT,16)    ; 171
437   0269          FDS172 = AND('ZS-105PS2':DIN_ALMCT,16)    ; 172
438   0270          FDS173 = AND('YS-P-42':DIN_ALMCT,16)      ; 173  +mct4-
439   0271          FDS174 = AND('YS-P-43':DIN_ALMCT,16)      ; 174  +mct4-
440   0272          FDS175 = AND('YS-P-44':DIN_ALMCT,16)      ; 175  +mct4-
```

```
441                     ;
442                     ;  DETERMINE SIGNAL STATUS, PSTAT (FXn):
443                     ;
444     0273            FX1 = 'YS-AN-101':DIN_VAL          ; 1
445     0274            FX2 = 'YS-AN-102':DIN_VAL          ; 2
446     0275            FX3 = 'YS-AN-103':DIN_VAL          ; 3
447     0276            FX4 = 'YS-AN-104':DIN_VAL          ; 4
448     0277            FX5 = 'YS-AN-105':DIN_VAL          ; 5
449     0278            FX6 = 'YS-AN-106':DIN_VAL          ; 6
450     0279            FX7 = 'YS-AN-107':DIN_VAL          ; 7
451     0280            FX8 = 'TXS-AN-2':DIN_VAL ; 8
452     0281            FX9 = 'YS-AP-101':DIN_VAL          ; 9
453     0282            FX10 = 'YS-AP-102':DIN_VAL         ; 10
454     0283            FX11 = 'YS-AP-103':DIN_VAL         ; 11
455     0284            FX12 = 'LDS-AP03D':DIN_VAL         ; 12
456     0285            FX13 = 'YS-AP-104':DIN_VAL         ; 13
457     0286            FX14 = 'YS-AP-105':DIN_VAL         ; 14
458     0287            FX15 = 'YS-AP-106':DIN_VAL         ; 15
459     0288            FX16 = 'YS-AP-107':DIN_VAL         ; 16
460     0289            FX17 = 'YS-AP-108':DIN_VAL         ; 17
461     0290            FX18 = 'YS-AP102M':DIN_VAL         ; 18
462     0291            FX19 = 'PSHAP102M':DIN_VAL         ; 19
463     0292            FX20 = 'PSLAP102M':DIN_VAL         ; 20
464     0293            FX21 = 'YS-AP102R':DIN_VAL         ; 21
465     0294            FX22 = 'WSHAP102M':DIN_VAL         ; 22
466     0295            FX23 = 'TSHAP102M':DIN_VAL         ; 23
467     0296            FX24 = 'VSHAP102M':DIN_VAL         ; 24
468     0297            FX25 = 'TXS-AP-2':DIN_VAL          ; 25
469     0298            FX26 = 'FSLAP2711':DIN_VAL         ; 26
470     0299            FX27 = 'PSH-APFP1':DIN_VAL         ; 27
471     0300            FX28 = 'ZS-AP-FP1':DIN_VAL         ; 28
472     0301            FX29 = 'FSL-APK13':DIN_VAL         ; 29
473     0302            FX30 = 'FSL-APK12':DIN_VAL         ; 30
474     0303            FX31 = 'FSL-APK11':DIN_VAL         ; 31
475     0304            FX32 = 'FSL-APK23':DIN_VAL         ; 32
476     0305            FX33 = 'FSL-APK22':DIN_VAL         ; 33
477     0306            FX34 = 'FSL-APK21':DIN_VAL         ; 34
478     0307            FX35 = 'FSH-DR712':DIN_VAL         ; 35
479     0308            FX36 = 'FXS-DR712':DIN_VAL         ; 36 -mct3+
480     0309            FX37 = 'PSH-APFL1':DIN_VAL         ; 37
481     0310            FX38 = 'ZS-AP-DR1':DIN_VAL         ; 38
482     0311            FX39 = 'ZS-AP-FL1':DIN_VAL         ; 39
483     0312            FX40 = 'YS-AW-101':DIN_VAL         ; 40
484     0313            FX41 = 1                           ; 41 -mct5+
485     0314            FX42 = 'YS-AW-103':DIN_VAL         ; 42
486     0315            FX43 = 'YS-AW-104':DIN_VAL         ; 43
487     0316            FX44 = 'YS-AW-105':DIN_VAL         ; 44
488     0317            FX45 = 'YS-AW-106':DIN_VAL         ; 45
489     0318            FX46 = 'TXS-AW-2':DIN_VAL          ; 46
490                     ;
491                     ;  deleted YS-AY-101, -102, and YS-AZ-101, -102: 47-50
492                     ;
493     0319            FX51  = 'LSH-AX-1':DIN_VAL                ; 51
494     0320            FX52  = 'LSL-AX-1':DIN_VAL                ; 52
495             ;-mct2- FX53  = AND('P-AN-101':DEV_STAT,16)        ; 53
```

```
496             ;       FX54  = AND('P-AN-102':DEV_STAT,16)      ; 54
497             ;       FX55  = AND('P-AN-103':DEV_STAT,16)      ; 55
498             ;       FX56  = AND('P-AN-104':DEV_STAT,16)      ; 56
499             ;       FX57  = AND('P-AN-105':DEV_STAT,16)      ; 57
500             ;       FX58  = AND('P-AN-106':DEV_STAT,16)      ; 58
501             ;-mct2- FX59  = AND('P-AN-107':DEV_STAT,16)      ; 59
502     0321            FX60  = AND('P-AP-101':DEV_STAT,16)      ; 60
503     0322            FX61  = AND('P-AP-102':DEV_STAT,16)      ; 61
504     0323            FX62  = AND('P-AP-103':DEV_STAT,16)      ; 62
505     0324            FX63  = AND('P-AP-104':DEV_STAT,16)      ; 63
506     0325            FX64  = AND('P-AP-105':DEV_STAT,16)      ; 64
507     0326            FX65  = AND('P-AP-106':DEV_STAT,16)      ; 65
508     0327            FX66  = AND('P-AP-107':DEV_STAT,16)      ; 66
509     0328            FX67  = AND('P-AP-108':DEV_STAT,16)      ; 67
510     0329            FX68  = AND('P-AP-102M':DEV_STAT,16)     ; 68
511     0330            FX69  = AND('P-AP-102R':DEV_STAT,16)     ; 69
512             ;-mct1- FX70  = AND('P-AW-101':DEV_STAT,16)      ; 70
513             ;       FX71  = AND('P-AW-102':DEV_STAT,16)      ; 71
514             ;       FX72  = AND('P-AW-103':DEV_STAT,16)      ; 72
515             ;       FX73  = AND('P-AW-104':DEV_STAT,16)      ; 73
516             ;       FX74  = AND('P-AW-105':DEV_STAT,16)      ; 74
517             ;-mct1- FX75  = AND('P-AW-106':DEV_STAT,16)      ; 75
518     0331            FX76  = AND('P-AY-101':DEV_STAT,16)      ; 76
519     0332            FX77  = AND('P-AY-102':DEV_STAT,16)      ; 77
520     0333            FX78  = AND('P-AZ-101':DEV_STAT,16)      ; 78
521     0334            FX79  = AND('P-AZ-102':DEV_STAT,16)      ; 79
522     0335            FX84  = 'PSH-FP':DIN_VAL          ; 84
523             ;
524     0336            IF FX90  = 0 THEN
525     0338            IF 'TI-244-TK':AI_MEAS >= 200.0 THEN FX90  =  1          ; 90
526     0339            ELSE FX90  = 0
527     0341            ELSE IF 'TI-244-TK':AI_MEAS <=  200.0 - 'TI-244-TK':AI_DEAD THEN FX90  =  0
528             ;
529     0342            IF FX91  = 0 THEN
530     0344            IF 'FI-244EXH':AI_MEAS <= 100.0 THEN FX91  =  1          ; 91
531     0345            ELSE FX91  = 0
532     0347            ELSE IF 'FI-244EXH':AI_MEAS >=  100.0 + 'FI-244EXH':AI_DEAD THEN FX91  =  0
533             ;
534     0348            IF FX92  = 0 THEN
535     0350            IF 'FI-244EXH':AI_MEAS >= 200.0 THEN FX92  =  1          ; 92
536     0351            ELSE FX92  = 0
537     0353            ELSE IF 'FI-244EXH':AI_MEAS <=  200.0 - 'FI-244EXH':AI_DEAD THEN FX92  =  0
538             ;
539     0354            IF FX94  = 0 THEN
540     0356            IF 'II-AW-102':AI_MEAS <= 12.0 THEN FX94  =  1           ; 94
541     0357            ELSE FX94  = 0
542     0359            ELSE IF 'II-AW-102':AI_MEAS >=  12.0 + 'II-AW-102':AI_DEAD THEN FX94  =  0
543             ;
544             ;       IF FX95  = 0 THEN
545             ;       IF 'II-AW-102':AI_MEAS >= 25.0 THEN FX95  =  1           ; 95
546             ;       ELSE FX95  = 0
547             ;       ELSE IF 'II-AW-102':AI_MEAS <=  25.0 - 'II-AW-102':AI_DEAD THEN FX95  =  0
548             ;
549             ;       IF FX96  = 0 THEN
550             ;       IF 'DDI-CA1-1':AI_MEAS <= -0.05 THEN FX96  =  1          ; 96
```

```
551             ;       ELSE FX96  = 0
552             ;       ELSE IF 'DDI-CA1-1':AI_MEAS >=  -0.05 + 'DDI-CA1-1':AI_DEAD THEN FX96  =  0
553             ;
554             ;       IF FX97  = 0 THEN
555             ;       IF 'DDI-CA1-1':AI_MEAS >= +0.05 THEN FX97  =  1          ; 97
556             ;       ELSE FX97  = 0
557             ;       ELSE IF 'DDI-CA1-1':AI_MEAS <=  +0.05 - 'DDI-CA1-1':AI_DEAD THEN FX97  =  0
558             ;
559     0360            IF FX98  = 0 THEN
560     0362            IF 'PDI-FH1-1':AI_MEAS >= 15.0 THEN FX98  =  1           ; 98
561     0363            ELSE FX98  = 0
562     0365            ELSE IF 'PDI-FH1-1':AI_MEAS <=  15.0 - 'PDI-FH1-1':AI_DEAD THEN FX98  =  0
563             ;
564     0366            IF FX99  = 0 THEN
565     0368            IF 'PDI-FH2-1':AI_MEAS >= 15.0 THEN FX99  =  1           ; 99
566     0369            ELSE FX99  = 0
567     0371            ELSE IF 'PDI-FH2-1':AI_MEAS <=  15.0 - 'PDI-FH2-1':AI_DEAD THEN FX99  =  0
568             ;
569     0372            IF FX100  = 0 THEN
570     0374            IF 'PDI-FH3-1':AI_MEAS >= 5.0 THEN FX100  =  1           ; 100
571     0375            ELSE FX100  = 0
572     0377            ELSE IF 'PDI-FH3-1':AI_MEAS <=  5.0 - 'PDI-FH3-1':AI_DEAD THEN FX100  =  0
573             ;
574     0378            IF FX101  = 0 THEN
575     0380            IF 'SIC-PB2-1':AI_MEAS >= 95.0 THEN FX101  =  1          ; 101
576     0381            ELSE FX101  = 0
577     0383            ELSE IF 'SIC-PB2-1':AI_MEAS <=  95.0 - 'SIC-PB2-1':AI_DEAD THEN FX101  =  0
578             ;
579     0384            FX102 = 'PSL-EA1-1':DIN_VAL      ; 102
580             ;
581     0385            IF FX103  = 0 THEN
582     0387            IF 'PI-EA1-13':AI_MEAS <= 75.0 THEN FX103  =  1          ; 103
583     0388            ELSE FX103  = 0
584     0390            ELSE IF 'PI-EA1-13':AI_MEAS >=  75.0 + 'PI-EA1-13':AI_DEAD THEN FX103  =  0
585             ;
586     0391            IF FX104  = 0 THEN
587     0393            IF 'PI-EA1-13':AI_MEAS <= 80.0 THEN FX104  =  1          ; 104
588     0394            ELSE FX104  = 0
589     0396            ELSE IF 'PI-EA1-13':AI_MEAS >=  80.0 + 'PI-EA1-13':AI_DEAD THEN FX104  =  0
590             ;
591     0397            IF FX105  = 0 THEN
592     0399            IF 'FIC-EA1-1':AI_MEAS <= 3000.0 THEN FX105  =  1 ; 105
593     0400            ELSE FX105  = 0
594     0402            ELSE IF 'FIC-EA1-1':AI_MEAS >=  3000.0 + 'FIC-EA1-14':AI_DEAD THEN FX105  =  0
595             ;
596     0403            IF FX106  = 0 THEN
597     0405            IF 'PI-EA1-1':AI_MEAS <= 17.7 THEN FX106  =  1           ; 106
598     0406            ELSE FX106  = 0
599     0408            ELSE IF 'PI-EA1-1':AI_MEAS >=  17.7 + 'PI-EA1-1':AI_DEAD THEN FX106  =  0
600             ;
601     0409            IF FX107  = 0 THEN
602     0411            IF 'PI-EA1-1':AI_MEAS >= 29.7 THEN FX107  =  1           ; 107
603     0412            ELSE FX107  = 0
604     0414            ELSE IF 'PI-EA1-1':AI_MEAS <=  29.7 - 'PI-EA1-1':AI_DEAD THEN FX107  =  0
605             ;
```

```
606   0415          IF FX108  = 0 THEN
607   0417          IF 'TI-EA1-5':AI_MEAS >= 190.0 THEN FX108  =  1          ; 108
608   0418          ELSE FX108  = 0
609   0420          ELSE IF 'TI-EA1-5':AI_MEAS <=  190.0 - 'TI-EA1-5':AI_DEAD THEN FX108  =  0
610         ;
611   0421          IF FX109  = 0 THEN
612   0423          IF 'PI-EC1-5':AI_MEAS >= 100.0 THEN FX109  =  1          ; 109
613   0424          ELSE FX109  = 0
614   0426          ELSE IF 'PI-EC1-5':AI_MEAS <=  100.0 - 'PI-EC1-5':AI_DEAD THEN FX109  =  0
615         ;
616   0427          IF FX110  = 0 THEN
617   0429          IF 'FIC-EC3-1':AI_MEAS <= 50.0 THEN FX110  =  1          ; 110
618   0430          ELSE FX110  = 0
619   0432          ELSE IF 'FIC-EC3-1':AI_MEAS >=  50.0 + 'FIC-EC3-1':AI_DEAD THEN FX110  =  0
620         ;
621   0433          IF FX111  = 0 THEN
622   0435          IF 'TIC-EC1-2':AI_MEAS >= 110.0 THEN FX111  =  1          ; 111
623   0436          ELSE FX111  = 0
624   0438          ELSE IF 'TIC-EC1-2':AI_MEAS <=  110.0 - 'TIC-EC1-2':AI_DEAD THEN FX111  =  0
625         ;
626   0439          FX112 = 'LDS-BFP-1':DIN_VAL      ; 112
627   0440          FX113 = 'LSL-RWSMP':DIN_VAL      ; 113
628   0441          FX114 = 'LDS-BFP-2':DIN_VAL      ; 114
629         ;
630   0442          IF FX115  = 0 THEN
631   0444          IF 'PDI-RW-1':AI_MEAS >= 13.0 THEN FX115  =  1           ; 115
632   0445          ELSE FX115  = 0
633   0447          ELSE IF 'PDI-RW-1':AI_MEAS <=  13.0 - 'PDI-RW-1':AI_DEAD THEN FX115  =  0
634         ;
635   0448          IF FX116  = 0 THEN
636   0450          IF 'PDI-RW-2':AI_MEAS >= 13.0 THEN FX116  =  1           ; 116
637   0451          ELSE FX116  = 0
638   0453          ELSE IF 'PDI-RW-2':AI_MEAS <=  13.0 - 'PDI-RW-2':AI_DEAD THEN FX116  =  0
639         ;
640   0454          IF FX117  = 0 THEN
641   0456          IF 'PI-RW-2':AI_MEAS <= 60.0 THEN FX117  =  1            ; 117
642   0457          ELSE FX117  = 0
643   0459          ELSE IF 'PI-RW-2':AI_MEAS >=  60.0 + 'PI-RW-2':AI_DEAD THEN FX117  =  0
644         ;
645         ;       IF FX118  = 0 THEN
646         ;       IF 'PI-RW-2':AI_MEAS >= 160.0 THEN FX118  =  1           ; 118
647         ;       ELSE FX118  = 0
648         ;       ELSE IF 'PI-RW-2':AI_MEAS <=  160.0 - 'PI-RW-2':AI_DEAD THEN FX118  =  0
649         ;
650   0460          IF FX119  = 0 THEN
651   0462          IF 'FIC-EC1-1':AI_MEAS <= 1200.0 THEN FX119  =  1 ; 119
652   0463          ELSE FX119  = 0
653   0465          ELSE IF 'FIC-EC1-1':AI_MEAS >=  1200.0 + 'FIC-EC1-1':AI_DEAD THEN FX119  =  0
654         ;
655         ;       IF FX120  = 0 THEN
656         ;       IF 'FIC-EC1-1':AI_MEAS >= 2800.0 THEN FX120  =  1 ; 120
657         ;       ELSE FX120  = 0
658         ;       ELSE IF 'FIC-EC1-1':AI_MEAS <=  2800.0 - 'FIC-EC1-1':AI_DEAD THEN FX120  =  0
659         ;
660   0466          IF FX121  = 0 THEN
```

```
661   0468          IF 'II-EXC-1':AI_MEAS <= 1.0 THEN FX121  =  1            ; 121
662   0469        ELSE FX121  = 0
663   0471          ELSE IF 'II-EXC-1':AI_MEAS >=  1.0 + 'II-EXC-1':AI_DEAD THEN FX121  =  0
664         ;
665   0472          IF FX126  = 0 THEN
666   0474          IF 'II-AS-1':AI_MEAS <= 2.0 THEN FX126  =  1             ; 126
667   0475          ELSE FX126  = 0
668   0477          ELSE IF 'II-AS-1':AI_MEAS >=  2.0 + 'II-AS-1':AI_DEAD THEN FX126  =  0
669         ;
670   0478          FX128 = 'YS-A-C100':DIN_VAL      ; 128
671   0479          FX129 = 'YS-P-C100':DIN_VAL      ; 129
672   0480          FX130 = 'ZS-IDX-2':DIN_VAL       ; 130
673   0481          FX131 = 'PSH-ILF-1':DIN_VAL      ; 131
674   0482          FX132 = 'ZS-RC3-4':DIN_VAL       ; 132
675   0483          FX133 = 'ZS-RC3-5':DIN_VAL       ; 133
676         ;
677   0484          IF FX135  = 0 THEN
678   0486          IF 'LI-C100-1':AI_MEAS <= 0.98 THEN FX135  =  1          ; 135
679   0487          ELSE FX135  = 0
680   0489          ELSE IF 'LI-C100-1':AI_MEAS >=  0.98 + 'LI-C100-1':AI_DEAD THEN FX135  =  0
681       ;
682       ;         IF FX136  = 0 THEN                                       ;commented out, -mct5+
683       ;         IF 'II-P-C100':AI_MEAS <= 5.0 THEN FX136  =  1           ; 136
684       ;         ELSE FX136  = 0
685       ;         ELSE IF 'II-P-C100':AI_MEAS >=  5.0 + 'II-P-C100':AI_DEAD THEN FX136  =  0
686       ;
687   0490          IF FX137 = 0 THEN
688   0492          IF 'PDI-CA1-3':AI_MEAS >= 15.0 THEN FX137 = 1
689   0493          ELSE FX137 = 0
690   0495          ELSE IF 'PDI-CA1-3':AI_MEAS <= 15.0 - 'PDI-CA1-3':AI_DEAD THEN FX137 = 0
691         ;
692   0496          FX138 = 'LSL-SC-1':DIN_VAL       ; 138
693   0497          FX139 = 'LSL-SC-2':DIN_VAL       ; 139
694   0498          FX140 = 'LSL-SC-3':DIN_VAL       ; 140
695   0499          FX143 = 'YS-CPE1-1':DIN_VAL      ; 143
696   0500          FX144 = 'YS-CPE2-1':DIN_VAL      ; 144
697         ;
698   0501          IF FX145  = 0 THEN
699   0503          IF 'PI-IA-1':AI_MEAS <= 18.0 THEN FX145  =  1            ; 145
700   0504          ELSE FX145  = 0
701   0506          ELSE IF 'PI-IA-1':AI_MEAS >=  18.0 + 'PI-IA-1':AI_DEAD THEN FX145  =  0
702         ;
703   0507          IF FX146  = 0 THEN
704   0509          IF 'PI-IA-1':AI_MEAS >= 25.0 THEN FX146  =  1            ; 146
705   0510          ELSE FX146  = 0
706   0512          ELSE IF 'PI-IA-1':AI_MEAS <=  25.0 - 'PI-IA-1':AI_DEAD THEN FX146  =  0
707       ;
708       ;         IF FX147  = 0 THEN
709       ;         IF 'PI-PA-1':AI_MEAS <= 80.0 THEN FX147  =  1            ; 147
710       ;         ELSE FX147  = 0
711       ;         ELSE IF 'PI-PA-1':AI_MEAS >=  80.0 + 'PI-PA-1':AI_DEAD THEN FX147  =  0
712       ;
713   0513          IF FX148  = 0 THEN
714   0515          IF 'PI-PA-1':AI_MEAS >= 120.0 THEN FX148  =  1           ; 148
715   0516          ELSE FX148  = 0
```

```
716  0518          ELSE IF 'PI-PA-1':AI_MEAS <=  120.0 - 'PI-PA-1':AI_DEAD THEN FX148  =  0
717          ;
718  0519          IF FX149  = 0 THEN
719  0521          IF 'TI-AMBS-1':AI_MEAS <= 48.0 THEN FX149  =  1          ; 149
720  0522          ELSE FX149  = 0
721  0524          ELSE IF 'TI-AMBS-1':AI_MEAS >=  48.0 + 'TI-AMBS-1':AI_DEAD THEN FX149  =  0
722          ;
723  0525          IF FX150  = 0 THEN
724  0527          IF 'TI-AMBS-1':AI_MEAS <= 55.0 THEN FX150  =  1          ; 150
725  0528          ELSE FX150  = 0
726  0530          ELSE IF 'TI-AMBS-1':AI_MEAS >=  55.0 + 'TI-AMBS-1':AI_DEAD THEN FX150  =  0
727          ;
728  0531          IF FX151  = 0 THEN
729  0533          IF 'TI-AMBS-1':AI_MEAS >= 80.0 THEN FX151  =  1          ; 151
730  0534          ELSE FX151  = 0
731  0536          ELSE IF 'TI-AMBS-1':AI_MEAS <=  80.0 - 'TI-AMBS-1':AI_DEAD THEN FX151  =  0
732          ;
733  0537          IF FX152  = 0 THEN
734  0539          IF 'TI-AMBS-1':AI_MEAS >= 90.0 THEN FX152  =  1          ; 152
735  0540          ELSE FX152  = 0
736  0542          ELSE IF 'TI-AMBS-1':AI_MEAS <=  90.0 - 'TI-AMBS-1':AI_DEAD THEN FX152  =  0
737          ;
738  0543          IF FX153  = 0 THEN
739  0545          IF 'TI-AMBS-2':AI_MEAS <= 48.0 THEN FX153  =  1          ; 153
740  0546          ELSE FX153  = 0
741  0548          ELSE IF 'TI-AMBS-2':AI_MEAS >=  48.0 + 'TI-AMBS-2':AI_DEAD THEN FX153  =  0
742          ;
743  0549          IF FX154  = 0 THEN
744  0551          IF 'TI-AMBS-2':AI_MEAS <= 55.0 THEN FX154  =  1          ; 154
745  0552          ELSE FX154  = 0
746  0554          ELSE IF 'TI-AMBS-2':AI_MEAS >=  55.0 + 'TI-AMBS-2':AI_DEAD THEN FX154  =  0
747          ;
748  0555          IF FX155  = 0 THEN
749  0557          IF 'TI-AMBS-2':AI_MEAS >= 80.0 THEN FX155  =  1          ; 155
750  0558          ELSE FX155  = 0
751  0560          ELSE IF 'TI-AMBS-2':AI_MEAS <=  80.0 - 'TI-AMBS-2':AI_DEAD THEN FX155  =  0
752          ;
753  0561          IF FX156  = 0 THEN
754  0563          IF 'TI-AMBS-2':AI_MEAS >= 90.0 THEN FX156  =  1          ; 156
755  0564          ELSE FX156  = 0
756  0566          ELSE IF 'TI-AMBS-2':AI_MEAS <=  90.0 - 'TI-AMBS-2':AI_DEAD THEN FX156  =  0
757          ;
758  0567          FX157 = 'YS-A-E101':DIN_VAL      ; 157
759  0568          FX158 = 'YS-P-E101':DIN_VAL      ; 158
760  0569          FX159 = 'YS-A-E102':DIN_VAL      ; 159
761  0570          FX160 = 'YS-P-E102':DIN_VAL      ; 160
762  0571          FX161 = 'YS-A-E104':DIN_VAL      ; 161
763  0572          FX162 = 'YS-P-E104':DIN_VAL      ; 162
764          ;
765  0573          IF FX163  = 0 THEN
766  0575          IF 'TI-E101-2':AI_MEAS >= 150.0 THEN FX163  =  1         ; 163
767  0576          ELSE FX163  = 0
768  0578          ELSE IF 'TI-E101-2':AI_MEAS <=  150.0 - 'TI-E101-2':AI_DEAD THEN FX163  =  0
769          ;
770  0579          IF FX164  = 0 THEN
```

```
771   0581          IF 'WFI-E101':AI_MEAS <= 15.0 THEN FX164  =  1                    ; 164
772   0582          ELSE FX164  = 0
773   0584          ELSE IF 'WFI-E101':AI_MEAS >=  15.0 + 'WFI-E101':AI_DEAD THEN FX164  =  0
774           ;
775   0585          IF FX165  = 0 THEN
776   0587          IF 'WFI-E101':AI_MEAS <= 25.0 THEN FX165  =  1                    ; 165
777   0588          ELSE FX165  = 0
778   0590          ELSE IF 'WFI-E101':AI_MEAS >=  25.0 + 'WFI-E101':AI_DEAD THEN FX165  =  0
779           ;
780   0591          IF FX166  = 0 THEN
781   0593          IF 'WFI-E102':AI_MEAS <= 5.0 THEN FX166  =  1                     ; 166
782   0594          ELSE FX166  = 0
783   0596          ELSE IF 'WFI-E102':AI_MEAS >=  5.0 + 'WFI-E102':AI_DEAD THEN FX166  =  0
784           ;
785   0597          IF FX167  = 0 THEN
786   0599          IF 'WFI-E102':AI_MEAS <= 12.0 THEN FX167  =  1                    ; 167
787   0600          ELSE FX167  = 0
788   0602          ELSE IF 'WFI-E102':AI_MEAS >=  12.0 + 'WFI-E102':AI_DEAD THEN FX167  =  0
789           ;
790   0603          IF FX168  = 0 THEN
791   0605          IF 'TI-E104-2':AI_MEAS >= 170.0 THEN FX168  =  1          ; 168
792   0606          ELSE FX168  = 0
793   0608          ELSE IF 'TI-E104-2':AI_MEAS <=  170.0 - 'TI-E104-2':AI_DEAD THEN FX168  =  0
794           ;
795   0609          IF FX169  = 0 THEN
796   0611          IF 'WFI-E104':AI_MEAS <= 5 THEN FX169  =  1                       ; 169
797   0612          ELSE FX169  = 0
798   0614          ELSE IF 'WFI-E104':AI_MEAS >=  5 + 'WFI-E104':AI_DEAD THEN FX169  =  0
799           ;
800   0615          IF FX170  = 0 THEN
801   0617          IF 'WFI-E104':AI_MEAS <= 12.0 THEN FX170  =  1                    ; 170
802   0618          ELSE FX170  = 0
803   0620          ELSE IF 'WFI-E104':AI_MEAS >=  12.0 + 'WFI-E104':AI_DEAD THEN FX170  =  0
804           ;
805   0621          FX171 = 'ZS-105PS1':DIN_VAL      ; 171
806   0622          FX172 = 'ZS-105PS2':DIN_VAL      ; 172
807   0623          FX173 = 'YS-P-42':DIN_VAL        ; 173  +mct4-
808   0624          FX174 = 'YS-P-43':DIN_VAL        ; 174  +mct4-
809   0625          FX175 = 'YS-P-44':DIN_VAL        ; 175  +mct4-
810                 ;
811                 ;  COMPARE ALARM SETPT (PREV) TO CURRENT SIGNAL STATUS (PSTAT)
812                 ;       TO DETERMINE IF ALARM OR CLEAR:
813                 ;
814   0626    LOP:  ASSIGN PSTAT = FX1
815   0627          ASSIGN PALARM = FAL1
816   0628          ASSIGN PREV = FRV1
817   0629          ASSIGN PTYPE = FDS1
818   0630          EPN3 = "LOP:"
819   0631          IN1 = TIMSTR(TIME3)
820           ;
821   0632          FOR ILOOP = 1 TO 175             ; +mct4-
822   0633          IF !(PTYPE)THEN                          ; IF AN ACTIVE ALARM (not inha)
823                    BEGIN
824   0634             IF!(PALARM)THEN                       ; IF NOT PREVIOUSLY ALARMED
825                       BEGIN
```

```
826   0636                IF PREV = PSTAT THEN GOSUB ALM      ; IF ALARM CONDITION MET
827                       END
828   0638              ELSE IF PREV!=PSTAT THEN GOSUB CLR    ; IF ALARM CLEARED
829                     END
830                   ELSE
831   0640            IF(PALARM)THEN GOSUB CLR
832   0641            STEP PSTAT BY 1
833   0642            STEP PALARM BY 1
834   0643            STEP PREV BY 1
835   0644            STEP PTYPE BY 1
836   0645            NEXT ILOOP
837           ;
838   0647            IF IN4 > 16 THEN FSPAR4 = 1       ; "*MORE" ON GRAPHIC 100
839   0648               ELSE FSPAR4 = 0
840   0650            IF IN4 > 32 THEN FSPAR5 = 1       ; "*MORE" ON GRAHIC 101
841   0651               ELSE FSPAR5 = 0
842   0652            WAIT 2
843   0653            GOTO AGAIN
844           ;
845           ;       E N D   P R O G R A M   L O O P
846           ;
847           ;       BEGIN CLEAR SUBROUTINE:
848           ;
849   0654    CLR:    EPN4 = "CLR"                      ; SET PALARM = 0, SET PLINE = 0
850                                                     ; MOVE LOWER LINES DOWN
851   0655            PALARM = 0
852   0656            PRINT 0,"           ~~~  CLEAR ALARM,  ILOOP = %i",ILOOP
853   0657            IN1 = TIMSTR(TIME4)
854   0658            ASSIGN PLINE = INT0
855   0659            STEP PLINE BY ILOOP
856   0660            IF PLINE = 0 THEN
857                      begin
858   0661               PRINT 0,"          ~~~  W A R N I N G  PLINE = 0, ILOOP = %i",ILOOP
859   0662               LAMP 0,60,1
860   0663               goto term
861                      end
862   0664            IN4 = IN4 - 1
863   0665            IN2 = ILOOP
864   0666            FIRST = 46 - PLINE - 1
865   0667            STPPER = 1
866           ;
867   0668            ASSIGN NXTGRA  = GRAFX0
868   0669            ASSIGN PGRAFX   = GRAFX1
869           ;
870   0670            ASSIGN NXTLSG =  LSG0
871   0671            ASSIGN PLRGSG =  LSG1
872           ;
873   0672            ASSIGN NXTMSG = MSSG0
874   0673            ASSIGN PMESSG = MSSG1
875           ;
876   0674            STEP NXTGRA BY (PLINE - 1)
877   0675            STEP PGRAFX BY (PLINE - 1)
878   0676            STEP NXTMSG BY PLINE - 1
879   0677            STEP PMESSG BY PLINE - 1
880   0678            STEP NXTLSG BY PLINE - 1
```

```
881   0679            STEP PLRGSG BY PLINE - 1
882   0680            GOSUB SRT
883           ;
884   0681            LAST = PLINE
885   0682            ASSIGN PLINE = INT1
886           ;
887   0683            FOR KLOOP = 1 TO 175              ; +mct4-
888   0684            IF KLOOP = ILOOP THEN
889                       BEGIN
890   0685                PLINE = 0
891           ;           PRINT 0,"            ~~~  PLINE NOW = 0, KLOOP = %i",KLOOP
892                       END
893   0686               ELSE IF PLINE > LAST THEN
894                       BEGIN
895   0687                   PLINE = PLINE - 1
896           ;               PRINT 0,"            ~~~  (PLINE - 1), KLOOP = %i, PLINE = %i",KLOOP,pline
897   0689                    if pline = 0 then goto term
898                       END
899   0690            STEP PLINE BY 1
900   0691            NEXT KLOOP
901   0692            RETURN
902           ;
903           ;       END CLEAR SUBROUTINE
904           ;
905           ;       BEGIN SORT SUB-SUBROUTINE (CALLED BY CLEAR):
906           ;
907   0693    SRT:    EPN3 = "SRT"
908   0694            IN1 = TIMSTR(TIME3)
909   0695            FOR BVAL =  0 TO FIRST
910   0696            NXTGRA = PGRAFX
911   0697            NXTMSG = PMESSG
912   0698            NXTLSG = PLRGSG
913   0699            STEP NXTGRA BY STPPER
914   0700            STEP PGRAFX BY STPPER
915   0701            STEP NXTMSG BY STPPER
916   0702            STEP PMESSG BY STPPER
917   0703            STEP NXTLSG BY STPPER
918   0704            STEP PLRGSG BY STPPER
919   0705            NEXT BVAL
920   0706            RETURN
921           ;
922           ;       END SORT SUB-SUBROUTINE
923           ;
924           ;       BEGIN ALARM SUBROUTINE:
925           ;
926   0707    ALM:    EPN4 = "ALM:"
927   0708            IN4 = IN4 + 1
928   0709            IN1 = TIMSTR(TIME4)
929   0710            IN2 = ILOOP
930   0711            PALARM = 1
931           ;       print adi,"alarm routine , iloop = %i, palarm = %i",iloop,palarm
932   0712            LAMP 0,60,3
933   0713            FIRST = 45
934   0714            STPPER = -1
935   0715            ASSIGN NXTGRA = GRAF46
```

```
936   0716            ASSIGN PGRAFX = GRAF45
937   0717            ASSIGN NXTLSG =  LSG46
938   0718            ASSIGN PLRGSG =  LSG45
939   0719            ASSIGN NXTMSG = MSSG46
940   0720            ASSIGN PMESSG = MSSG45
941   0721            GOSUB SRT
942           ;
943           ;       GET STRINGS FOR ALARM SUBROUTINE:
944           ;
945   0722            ON ILOOP GOTO S1,S2,S3,S4,S5,S6,S7,S8,S9,S10
946   0723            ON ILOOP - 10 GOTO S11,S12,S13,S14,S15,S16,S17,S18,S19,S20
947   0724            ON ILOOP - 20 GOTO S21,S22,S23,S24,S25,S26,S27,S28,S29,S30
948   0725            ON ILOOP - 30 GOTO S31,S32,S33,S34,S35,S36,S37,S38,S39,S40
949   0726            ON ILOOP - 40 GOTO S41,S42,S43,S44,S45,S46,S47,S48,S49,S50
950   0727            ON ILOOP - 50 GOTO S51,S52,S53,S54,S55,S56,S57,S58,S59,S60
951   0728            ON ILOOP - 60 GOTO S61,S62,S63,S64,S65,S66,S67,S68,S69,S70
952   0729            ON ILOOP - 70 GOTO S71,S72,S73,S74,S75,S76,S77,S78,S79,S80
953   0730            ON ILOOP - 80 GOTO S81,S82,S83,S84,S85,S86,S87,S88,S89,S90
954   0731            ON ILOOP - 90 GOTO S91,S92,S93,S94,S95,S96,S97,S98,S99,S100
955   0732            ON ILOOP - 100 GOTO S101,S102,S103,S104,S105,S106,S107,S108,S109,S110
956   0733            ON ILOOP - 110 GOTO S111,S112,S113,S114,S115,S116,S117,S118,S119,S120
957   0734            ON ILOOP - 120 GOTO S121,S122,S123,S124,S125,S126,S127,S128,S129,S130
958   0735            ON ILOOP - 130 GOTO S131,S132,S133,S134,S135,S136,S137,S138,S139,S140
959   0736            ON ILOOP - 140 GOTO S141,S142,S143,S144,S145,S146,S147,S148,S149,S150
960   0737            ON ILOOP - 150 GOTO S151,S152,S153,S154,S155,S156,S157,S158,S159,S160
961   0738            ON ILOOP - 160 GOTO S161,S162,S163,S164,S165,S166,S167,S168,S169,S170
962   0739            ON ILOOP - 170 GOTO S171,S172,S173,S174,S175                          ; +mct4-
963   0740            GOTO AMSG
964           ;
965   0741    S1:     MSSG0 = "AN-101 PUMP OFF              "
966   0742            EPN0 = "YS-AN-101"
967   0743            GRAFX0 = 270
968   0744            GOTO AMSG
969   0745    S2:     MSSG0 = "AN-102 PUMP OFF              "
970   0746            EPN0 = "YS-AN-102"
971   0747            GRAFX0 = 270
972   0748            GOTO AMSG
973   0749    S3:     MSSG0 = "AN-103 PUMP OFF          "
974   0750            EPN0 = "YS-AN-103"
975   0751            GRAFX0 = 270
976   0752            GOTO AMSG
977   0753    S4:     MSSG0 = "AN-104 PUMP OFF          "
978   0754            EPN0 = "YS-AN-104"
979   0755            GRAFX0 = 270
980   0756            GOTO AMSG
981   0757    S5:     MSSG0 = "AN-105 PUMP OFF          "
982   0758            EPN0 = "YS-AN-105"
983   0759            GRAFX0 = 270
984   0760            GOTO AMSG
985   0761    S6:     MSSG0 = "AN-106 PUMP OFF          "
986   0762            EPN0 = "YS-AN-106"
987   0763            GRAFX0 = 270
988   0764            GOTO AMSG
989   0765    S7:     MSSG0 = "AN-107 PUMP OFF          "
990   0766            EPN0 = "YS-AN-107"
```

```
 991   0767            GRAFX0 = 270
 992   0768            GOTO AMSG
 993   0769    S8:     MSSG0 = "AN-FARM HEAT TRACE OVERTEMP"
 994   0770            EPN0 = "TXS-AN-2"
 995   0771            GRAFX0 = 274
 996   0772            GOTO AMSG
 997   0773    S9:     MSSG0 = "AP-101 PUMP MCC OFF        "
 998   0774            EPN0 = "YS-AP-101"
 999   0775            GRAFX0 = 290
1000   0776            GOTO AMSG
1001   0777    S10:    MSSG0 = "AP-102 PUMP MCC OFF        "
1002   0778            EPN0 = "YS-AP-102"
1003   0779            GRAFX0 = 290
1004   0780            GOTO AMSG
1005   0781    S11:    MSSG0 = "AP-103 PUMP MCC OFF        "
1006   0782            EPN0 = "YS-AP-103"
1007   0783            GRAFX0 = 290
1008   0784            GOTO AMSG
1009   0785    S12:    MSSG0 = "AP03D DRAIN PIT LEAK       "
1010   0786            EPN0 = "LDS-AP03D"
1011   0787            GRAFX0 = 290
1012   0788            GOTO AMSG
1013   0789    S13:    MSSG0 = "AP-104 PUMP MCC OFF        "
1014   0790            EPN0 = "YS-AP-104"
1015   0791            GRAFX0 = 290
1016   0792            GOTO AMSG
1017   0793    S14:    MSSG0 = "AP-105 PUMP MCC OFF        "
1018   0794            EPN0 = "YS-AP-105"
1019   0795            GRAFX0 = 290
1020   0796            GOTO AMSG
1021   0797    S15:    MSSG0 = "AP-106 PUMP MCC OFF        "
1022   0798            EPN0 = "YS-AP-106"
1023   0799            GRAFX0 = 290
1024   0800            GOTO AMSG
1025   0801    S16:    MSSG0 = "AP-107 PUMP MCC OFF        "
1026   0802            EPN0 = "YS-AP-107"
1027   0803            GRAFX0 = 290
1028   0804            GOTO AMSG
1029   0805    S17:    MSSG0 = "AP-108 PUMP MCC OFF        "
1030   0806            EPN0 = "YS-AP-108"
1031   0807            GRAFX0 = 290
1032   0808            GOTO AMSG
1033   0809    S18:    MSSG0 = "MIXER PUMP AP-102-3 MCC OFF"
1034   0810            EPN0 = "YS-AP102M"
1035   0811            GRAFX0 = 291
1036   0812            GOTO AMSG
1037   0813    S19:    MSSG0 = "MIXER PUMP AP-102-3 PRESS HI"
1038   0814            EPN0 = "PSHAP102M"
1039   0815            GRAFX0 = 291
1040   0816            GOTO AMSG
1041   0817    S20:    MSSG0 = "MIXER PUMP AP-102-3 PRESS LOW"
1042   0818            EPN0 = "PSLAP102M"
1043   0819            GRAFX0 = 291
1044   0820            GOTO AMSG
1045   0821    S21:    MSSG0 = "AP-102-3 ROTATOR MOTR MCC OFF "
```

```
1046    0822            EPNO = "YS-AP102R"
1047    0823            GRAFXO = 291
1048    0824            GOTO AMSG
1049    0825    S22:    MSSGO = "MIXER PUMP AP-102-3 STRAIN HI "
1050    0826            EPNO = "WSHAP102M"
1051    0827            GRAFXO = 291
1052    0828            GOTO AMSG
1053    0829    S23:    MSSGO = "MIXER PUMP AP-102-3 TEMP HI   "
1054    0830            EPNO = "TSHAP102M"
1055    0831            GRAFXO = 291
1056    0832            GOTO AMSG
1057    0833    S24:    MSSGO = "AP-102-3 VIBRATION HIGH       "
1058    0834            EPNO = "VSHAP102M"
1059    0835            GRAFXO = 291
1060    0836            GOTO AMSG
1061    0837    S25:    MSSGO = "AP-FARM HEAT TRACE OVER TEMP  "
1062    0838            EPNO = "TXS-AP-2"
1063    0839            GRAFXO = 295
1064    0840            GOTO AMSG
1065    0841    S26:    MSSGO = "271-AP BUILDING CAM FLOW LOW  "
1066    0842            EPNO = "FSLAP2711"
1067    0843            GRAFXO = 295
1068    0844            GOTO AMSG
1069    0845    S27:    MSSGO = "AP-FLSPIT FLUSH LINE PRESS HI "
1070    0846            EPNO = "PSH-APFP1"
1071    0847            GRAFXO = 292
1072    0848            GOTO AMSG
1073    0849    S28:    MSSGO = "AP-FARM VALVE PIT V-FP-1 OPEN "
1074    0850            EPNO = "ZS-AP-FP1"
1075    0851            GRAFXO = 292
1076    0852            GOTO AMSG
1077    0853    S29:    MSSGO = "AP-FARM K1 EXH MON SMPL FLO LO"
1078    0854            EPNO = "FSL-APK13"
1079    0855            GRAFXO = 294
1080    0856            GOTO AMSG
1081    0857    S30:    MSSGO = "AP-FARM K1 EXH REC SMPL FLO LO"
1082    0858            EPNO = "FSL-APK12"
1083    0859            GRAFXO = 294
1084    0860            GOTO AMSG
1085    0861    S31:    MSSGO = "AP-FARM K1 EXH STACK FLOW LOW "
1086    0862            EPNO = "FSL-APK11"
1087    0863            GRAFXO = 294
1088    0864            GOTO AMSG
1089    0865    S32:    MSSGO = "AP-FARM K2 EXH MON SMPL FLO LO"
1090    0866            EPNO = "FSL-APK23"
1091    0867            GRAFXO = 294
1092    0868            GOTO AMSG
1093    0869    S33:    MSSGO = "AP-FARM K2 EXH REC SMPL FLO LO"
1094    0870            EPNO = "FSL-APK22"
1095    0871            GRAFXO = 294
1096    0872            GOTO AMSG
1097    0873    S34:    MSSGO = "AP-FARM K2 EXH STACK FLOW LOW "
1098    0874            EPNO = "FSL-APK21"
1099    0875            GRAFXO = 294
1100    0876            GOTO AMSG
```

```
1101  0877  S35:   MSSG0 = "AP-FARM VLVPIT DR-712 FLOW HI "
1102  0878         EPN0 = "FSH-DR712"
1103  0879         GRAFX0 = 293
1104  0880         GOTO AMSG
1105  0881  S36:   MSSG0 = "AP-VLVPIT DR-712 FLO DET FAIL "
1106  0882         EPN0 = "FXS-DR712"
1107  0883         GRAFX0 = 293
1108  0884         GOTO AMSG
1109  0885  S37:   MSSG0 = "AP-VLVPIT FLUSH LINE PRES HIGH"
1110  0886         EPN0 = "PSH-APFL1"
1111  0887         GRAFX0 = 293
1112  0888         GOTO AMSG
1113  0889  S38:   MSSG0 = "AP-FARM VALVE PIT V-DR-1 OPEN "
1114  0890         EPN0 = "ZS-AP-DR1"
1115  0891         GRAFX0 = 293
1116  0892         GOTO AMSG
1117  0893  S39:   MSSG0 = "AP-FARM VALVE PIT V-FL-1 OPEN "
1118  0894         EPN0 = "ZS-AP-FL1"
1119  0895         GRAFX0 = 293
1120  0896         GOTO AMSG
1121  0897  S40:   MSSG0 = "AW-101 PUMP OFF              "
1122  0898         EPN0 = "YS-AW-101"
1123  0899         GRAFX0 = 300
1124  0900         GOTO AMSG
1125       S41:    ;MSSG0 = "AW-102 PUMP OFF              "      commented out, -mct5+
1126       ;       EPN0 = "YS-AW102"
1127       ;       GRAFX0 = 301
1128       ;       GOTO AMSG
1129  0901  S42:   MSSG0 = "AW-103 PUMP OFF              "
1130  0902         EPN0 = "YS-AW-103"
1131  0903         GRAFX0 = 300
1132  0904         GOTO AMSG
1133  0905  S43:   MSSG0 = "AW-104 PUMP OFF              "
1134  0906         EPN0 = "YS-AW-104"
1135  0907         GRAFX0 = 300
1136  0908         GOTO AMSG
1137  0909  S44:   MSSG0 = "AW-105 PUMP OFF              "
1138  0910         EPN0 = "YS-AW-105"
1139  0911         GRAFX0 = 300
1140  0912         GOTO AMSG
1141  0913  S45:   MSSG0 = "AW-106 PUMP OFF              "
1142  0914         EPN0 = "YS-AW-106"
1143  0915         GRAFX0 = 300
1144  0916         GOTO AMSG
1145  0917  S46:   MSSG0 = "AW-FARM HEAT TRACE OVERTEMP   "
1146  0918         EPN0 = "TXS-AW-2"
1147  0919         GRAFX0 = 302
1148  0920         GOTO AMSG
1149       S47:    ;       MSSG0 = "AY-101 PUMP MCC OFF          "
1150       ;       EPN0 = "YS-AY-101"
1151       ;       GRAFX0 = 322
1152       ;       GOTO AMSG
1153       S48:    ;       MSSG0 = "AY-102 PUMP MCC OFF          "
1154       ;       EPN0 = "YS-AY-102"
1155       ;       GRAFX0 = 322
```

```
1156            ;       GOTO AMSG
1157            S49:    ;       MSSG0 = "AZ-101 PUMP MCC OFF          "
1158            ;       EPN0 = "YS-AZ-101"
1159            ;       GRAFX0 = 323
1160            ;       GOTO AMSG
1161            S50:    ;       MSSG0 = "AZ-102 PUMP MCC OFF          "
1162            ;       EPN0 = "YS-AZ-102"
1163            ;       GRAFX0 = 323
1164            ;       GOTO AMSG
1165    0921    S51:    MSSG0 = "AX-A/B SEAL POT LEVEL HIGH    "
1166    0922            EPN0 = "LSH-AX-1"
1167    0923            GRAFX0 = 320
1168    0924            GOTO AMSG
1169    0925    S52:    MSSG0 = "AX-A/B SEAL POT LEVEL LOW     "
1170    0926            EPN0 = "LSL-AX-1"
1171    0927            GRAFX0 = 320
1172    0928            GOTO AMSG
1173            ;-mct2-
1174            S53:    ;       MSSG0 = "PUMP AN-101 SHUTDOWN "
1175            ;       EPN0 = "P-AN-101"
1176            ;       GRAFX0 = 270
1177            ;       GOTO AMSG
1178            S54:    ;       MSSG0 = "PUMP AN-102 SHUTDOWN         "
1179            ;       EPN0 = "P-AN-102"
1180            ;       GRAFX0 = 270
1181            ;       GOTO AMSG
1182            S55:    ;       MSSG0 = "PUMP AN-103 SHUTDOWN         "
1183            ;       EPN0 = "P-AN-103"
1184            ;       GRAFX0 = 270
1185            ;       GOTO AMSG
1186            S56:    ;       MSSG0 = "PUMP AN-104 SHUTDOWN         "
1187            ;       EPN0 = "P-AN-104"
1188            ;       GRAFX0 = 270
1189            ;       GOTO AMSG
1190            S57:    ;       MSSG0 = "PUMP AN-105 SHUTDOWN         "
1191            ;       EPN0 = "P-AN-105"
1192            ;       GRAFX0 = 270
1193            ;       GOTO AMSG
1194            S58:    ;       MSSG0 = "PUMP AN-106 SHUTDOWN         "
1195            ;       EPN0 = "P-AN-106"
1196            ;       GRAFX0 = 270
1197            ;       GOTO AMSG
1198            S59:    ;       MSSG0 = "PUMP AN-107 SHUTDOWN         "
1199            ;       EPN0 = "P-AN-107"
1200            ;       GRAFX0 = 270
1201            ;       GOTO AMSG
1202            ;-mct2-
1203    0929    S60:    MSSG0 = "PUMP AP-101 SHUTDOWN          "
1204    0930            EPN0 = "P-AP-101"
1205    0931            GRAFX0 = 290
1206    0932            GOTO AMSG
1207    0933    S61:    MSSG0 = "PUMP AP-102 SHUTDOWN          "
1208    0934            EPN0 = "P-AP-102"
1209    0935            GRAFX0 = 291
1210    0936            GOTO AMSG
```

```
1211  0937  S62:    MSSG0 = "PUMP AP-103 SHUTDOWN          "
1212  0938          EPN0 = "P-AP-103"
1213  0939          GRAFX0 = 290
1214  0940          GOTO AMSG
1215  0941  S63:    MSSG0 = "PUMP AP-104 SHUTDOWN          "
1216  0942          EPN0 = "P-AP-104"
1217  0943          GRAFX0 = 290
1218  0944          GOTO AMSG
1219  0945  S64:    MSSG0 = "PUMP AP-105 SHUTDOWN          "
1220  0946          EPN0 = "P-AP-105"
1221  0947          GRAFX0 = 290
1222  0948          GOTO AMSG
1223  0949  S65:    MSSG0 = "PUMP AP-106 SHUTDOWN          "
1224  0950          EPN0 = "P-AP-106"
1225  0951          GRAFX0 = 290
1226  0952          GOTO AMSG
1227  0953  S66:    MSSG0 = "PUMP AP-107 SHUTDOWN          "
1228  0954          EPN0 = "P-AP-107"
1229  0955          GRAFX0 = 290
1230  0956          GOTO AMSG
1231  0957  S67:    MSSG0 = "PUMP AP-108 SHUTDOWN          "
1232  0958          EPN0 = "P-AP-108"
1233  0959          GRAFX0 = 290
1234  0960          GOTO AMSG
1235  0961  S68:    MSSG0 = "MIXER PUMP AP-102M SHUTDOWN   "
1236  0962          EPN0 = "P-AP-102M"
1237  0963          GRAFX0 = 291
1238  0964          GOTO AMSG
1239  0965  S69:    MSSG0 = "ROTOR AP-102R SHUTDOWN        "
1240  0966          EPN0 = "P-AP-102R"
1241  0967          GRAFX0 = 291
1242  0968          GOTO AMSG
1243        ;-mct1-
1244        S70:    ;       MSSG0 = "PUMP AW-101 SHUTDOWN          "
1245        ;       EPN0 = "P-AW-101"
1246        ;       GRAFX0 = 300
1247        ;       GOTO AMSG
1248        S71:    ;       MSSG0 = "PUMP AW-102 SHUTDOWN          "
1249        ;       EPN0 = "P-AW-102"
1250        ;       GRAFX0 = 301
1251        ;       GOTO AMSG
1252        S72:    ;       MSSG0 = "PUMP AW-103 SHUTDOWN          "
1253        ;       EPN0 = "P-AW-103"
1254        ;       GRAFX0 = 300
1255        ;       GOTO AMSG
1256        S73:    ;       MSSG0 = "PUMP AW-104 SHUTDOWN          "
1257        ;       EPN0 = "P-AW-104"
1258        ;       GRAFX0 = 300
1259        ;       GOTO AMSG
1260        S74:    ;       MSSG0 = "PUMP AW-105 SHUTDOWN          "
1261        ;       EPN0 = "P-AW-105"
1262        ;       GRAFX0 = 300
1263        ;       GOTO AMSG
1264        S75:    ;       MSSG0 = "PUMP AW-106 SHUTDOWN          "
1265        ;       EPN0 = "P-AW-106"
```

```
1266          ;       GRAFX0 = 300
1267          ;       GOTO AMSG
1268          ;-mct1-
1269    0969  S76:    MSSG0 = "PUMP AY-101 SHUTDOWN          "
1270    0970          EPN0 = "P-AY-101"
1271    0971          GRAFX0 = 322
1272    0972          GOTO AMSG
1273    0973  S77:    MSSG0 = "PUMP AY-102 SHUTDOWN          "
1274    0974          EPN0 = "P-AY-102"
1275    0975          GRAFX0 = 322
1276    0976          GOTO AMSG
1277    0977  S78:    MSSG0 = "PUMP AZ-101 SHUTDOWN          "
1278    0978          EPN0 = "P-AZ-101"
1279    0979          GRAFX0 = 323
1280    0980          GOTO AMSG
1281    0981  S79:    MSSG0 = "PUMP AZ-102 SHUTDOWN          "
1282    0982          EPN0 = "P-AZ-102"
1283    0983          GRAFX0 = 323
1284    0984          GOTO AMSG
1285          S80:    ;       MSSG0 = "AW K1 STACK MON SMPL PUMP OFF "
1286          ;       EPN0 = "PSH-AWK11"
1287          ;       GRAFX0 = 304
1288          ;       GOTO AMSG
1289          S81:    ;       MSSG0 = "AW K1 RECORD SAMPLE PUMP OFF  "
1290          ;       EPN0 = "PSH-AWK12"
1291          ;       GRAFX0 = 304
1292          ;       GOTO AMSG
1293          S82:    ;       MSSG0 = "AW K2 STACK MON SMPL PUMP OFF "
1294          ;       EPN0 = "PSH-AWK21"
1295          ;       GRAFX0 = 304
1296          ;       GOTO AMSG
1297          S83:    ;       MSSG0 = "AW K2 RECORD SAMPLE PUMP OFF  "
1298          ;       EPN0 = "PSH-AWK22"
1299          ;       GRAFX0 = 304
1300          ;       GOTO AMSG
1301    0985  S84:    MSSG0 = "244-A FLUSH PIT PRESS HIGH    "
1302    0986          EPN0 = "PSH-FP"
1303    0987          GRAFX0 = 340
1304    0988          GOTO AMSG
1305          S85:    ;       MSSG0 = "TK-244-A PUMP 1 CURRENT HIGH  "
1306          ;       EPN0 = "II-P244A1"
1307          ;       GRAFX0 = 340
1308          ;       GOTO AMSG
1309          S86:    ;       MSSG0 = "TK-244-A PUMP 2 CURRENT HIGH  "
1310          ;       EPN0 = "II-P244A2"
1311          ;       GRAFX0 = 340
1312          ;       GOTO AMSG
1313          S87:    ;       MSSG0 = "244-A SUMP PUMP CURRENT HIGH  "
1314          ;       EPN0 = "II-P244A3"
1315          ;       GRAFX0 = 340
1316          ;       GOTO AMSG
1317          S88:    ;       MSSG0 = "244-A SUMP PUMP CURRENT HIHI  "
1318          ;       EPN0 = "II-P244A3"
1319          ;       GRAFX0 = 340
1320          ;       GOTO AMSG
```

```
1321           S89:    ;       MSSG0 = "TK-244-A AGITATOR CURRENT HI  "
1322           ;       EPN0 = "II-A-244A"
1323           ;       GRAFX0 = 340
1324           ;       GOTO AMSG
1325    0989   S90:    MSSG0 = "TK-244-A LIQUID TEMPERTURE HI "
1326    0990           EPN0 = "TI-244-TK"
1327    0991           GRAFX0 = 340
1328    0992           GOTO AMSG
1329    0993   S91:    MSSG0 = "244A-EXHAUST STACK FLOW LOW   "
1330    0994           EPN0 = "FI-244EXH"
1331    0995           GRAFX0 = 341
1332    0996           GOTO AMSG
1333    0997   S92:    MSSG0 = "244A-EXHAUST STACK FLOW HIGH  "
1334    0998           EPN0 = "FI-244EXH"
1335    0999           GRAFX0 = 341
1336    1000           GOTO AMSG
1337           S93:    ;       MSSG0 = "TK-A-350 PUMP CURRENT HIGH    "
1338           ;       EPN0 = "II-350-1"
1339           ;       GRAFX0 = 343
1340           ;       GOTO AMSG
1341    1001   S94:    MSSG0 = "AW-102 FEED PUMP CURRENT LOW "
1342    1002           EPN0 = "II-AW-102"
1343    1003           GRAFX0 = 301
1344    1004           GOTO AMSG
1345           S95:    ;       MSSG0 = "AW-102 FEED PUMP CURRENT HIGH"
1346           ;       EPN0 = "II-AW-102"
1347           ;       GRAFX0 = 301
1348           ;       GOTO AMSG
1349           S96:    ;       MSSG0 = "EVAP DENSITY DIFFERENCE LOLO  "
1350           ;       EPN0 = "DDI-CA1-1"
1351           ;       GRAFX0 = 10
1352           ;       GOTO AMSG
1353           S97:    ;       MSSG0 = "EVAP DENSITY DIFFERENCE HIHI  "
1354           ;       EPN0 = "DDI-CA1-1"
1355           ;       GRAFX0 = 10
1356           ;       GOTO AMSG
1357    1005   S98:    MSSG0 = "F-H-1 RAW WATER FILTER DP HIGH"
1358    1006           EPN0 = "PDI-FH1-1"
1359    1007           GRAFX0 = 10
1360    1008           GOTO AMSG
1361    1009   S99:    MSSG0 = "F-H-2 RAW WATER FILTER DP HIGH"
1362    1010           EPN0 = "PDI-FH2-1"
1363    1011           GRAFX0 = 10
1364    1012           GOTO AMSG
1365    1013   S100:   MSSG0 = "F-H-3 RAW WATER FILTER DP HIGH"
1366    1014           EPN0 = "PDI-FH3-1"
1367    1015           GRAFX0 = 10
1368    1016           GOTO AMSG
1369    1017   S101:   MSSG0 = "SLURRY PUMP SPEED CNTRLR HIGH "
1370    1018           EPN0 = "SIC-PB2-1"
1371    1019           GRAFX0 = 15
1372    1020           GOTO AMSG
1373    1021   S102:   MSSG0 = "REBOILER 10 LB STM PRESS LOW  "
1374    1022           EPN0 = "PSL-EA1-1"
1375    1023           GRAFX0 = 13
```

```
1376    1024            GOTO AMSG
1377    1025    S103:   MSSG0 = "90 PSI STEAM HEADER PRESS LOLO"
1378    1026            EPN0 = "PI-EA1-13"
1379    1027            GRAFX0 = 13
1380    1028            GOTO AMSG
1381    1029    S104:   MSSG0 = "90 PSI STEAM HEADER PRESS LOW "
1382    1030            EPN0 = "PI-EA1-13"
1383    1031            GRAFX0 = 13
1384    1032            GOTO AMSG
1385    1033    S105:   MSSG0 = "REBOILER STEAM FLOW LOW       "
1386    1034            EPN0 = "FIC-EA1-1"
1387    1035            GRAFX0 = 13
1388    1036            GOTO AMSG
1389    1037    S106:   MSSG0 = "REBOILER STEAM INLET PRESS LOW"
1390    1038            EPN0 = "PI-EA1-1"
1391    1039            GRAFX0 = 13
1392    1040            GOTO AMSG
1393    1041    S107:   MSSG0 = "REBOILER STEAM INLET PRESS HI "
1394    1042            EPN0 = "PI-EA1-1"
1395    1043            GRAFX0 = 13
1396    1044            GOTO AMSG
1397    1045    S108:   MSSG0 = "REBOILER STM CONDNSATE TEMP HI"
1398    1046            EPN0 = "TI-EA1-5"
1399    1047            GRAFX0 = 13
1400    1048            GOTO AMSG
1401    1049    S109:   MSSG0 = "E-C-1 RAW WATER INLET PRESS HI"
1402    1050            EPN0 = "PI-EC1-5"
1403    1051            GRAFX0 = 16
1404    1052            GOTO AMSG
1405    1053    S110:   MSSG0 = "E-C-3 CONDENSR URW FLOW LOW   "
1406    1054            EPN0 = "FIC-EC3-1"
1407    1055            GRAFX0 = 16
1408    1056            GOTO AMSG
1409    1057    S111:   MSSG0 = "E-C-1 PC OUTLET TEMP HIGH     "
1410    1058            EPN0 = "TIC-EC1-2"
1411    1059            GRAFX0 = 16
1412    1060            GOTO AMSG
1413    1061    S112:   MSSG0 = "WATER SERVICE BLD BFP-1 LEAK  "
1414    1062            EPN0 = "LDS-BFP-1"
1415    1063            GRAFX0 = 11
1416    1064            GOTO AMSG
1417    1065    S113:   MSSG0 = "WATER SERVCE BLD SUMP LEVEL LO"
1418    1066            EPN0 = "LSL-RWSMP"
1419    1067            GRAFX0 = 11
1420    1068            GOTO AMSG
1421    1069    S114:   mssg0 = "WTR SRVICE BUILDING BFP-2 LEAK"
1422    1070            EPN0 = "LDS-BFP-2"
1423    1071            GRAFX0 = 11
1424    1072            GOTO AMSG
1425    1073    S115:   MSSG0 = "F-RW-1 FILTER DELTA PRESS  HI "
1426    1074            EPN0 = "PDI-RW-1"
1427    1075            GRAFX0 = 11
1428    1076            GOTO AMSG
1429    1077    S116:   MSSG0 = "F-RW-2 FILTER DELTA PRESS HI  "
1430    1078            EPN0 = "PDI-RW-2"
```

```
1431    1079            GRAFX0 = 11
1432    1080            GOTO AMSG
1433    1081    S117:   MSSG0 = "F-RW-1/2 FLTR UPSTREAM PRES LO"
1434    1082            EPN0 = "PI-RW-2"
1435    1083            GRAFX0 = 11
1436            S118:   ;       MSSG0 = "F-RW-1/2 FLTR UPSTRM PRES HIHI"
1437            ;       EPN0 = "PI-RW-2"
1438            ;       GRAFX0 = 11
1439            ;       GOTO AMSG
1440    1084    S119:   MSSG0 = "E-C-1 CONDENSR URW FLOW LOW   "
1441    1085            EPN0 = "FIC-EC1-1"
1442    1086            GRAFX0 = 17
1443    1087            GOTO AMSG
1444            S120:   ;       MSSG0 = "E-C-1 CONDENSR URW FLOW HIGH  "
1445            ;       EPN0 = "FIC-EC1-1"
1446            ;       GRAFX0 = 17
1447            ;       GOTO AMSG
1448    1088    S121:   MSSG0 = "VESSEL VENT EXHSTR CURRENT LO"
1449    1089            EPN0 = "II-EXC-1"
1450    1090            GRAFX0 = 20
1451    1091            GOTO AMSG
1452            S122:   ;       MSSG0 = "VESSEL VENT EXHSTR CURRENT HI "
1453            ;       EPN0 = "II-EXC-1"
1454            ;       GRAFX0 = 20
1455            ;       GOTO AMSG
1456            S123:   ;       MSSG0 = "VESSEL VENT EXHAUSTER FLO LOLO"
1457            ;       EPN0 = "FI-AS-5"
1458            ;       GRAFX0 = 20
1459            ;       GOTO AMSG
1460            S124:   ;       MSSG0 = "VESSEL VENT EXHAUSTER FLOW LOW"
1461            ;       EPN0 = "FI-AS-5"
1462            ;       GRAFX0 = 20
1463            ;       GOTO AMSG
1464            S125:   ;       MSSG0 = "VESSEL VENT EXHAUSTER FLOW HI "
1465            ;       EPN0 = "FI-AS-5"
1466            ;       GRAFX0 = 20
1467            ;       GOTO AMSG
1468    1092    S126:   MSSG0 = "AIR SAMPLE PUMP CURRENT LOW   "
1469    1093            EPN0 = "II-AS-1"
1470    1094            GRAFX0 = 20
1471    1095            GOTO AMSG
1472            S127:   ;       MSSG0 = "AIR SAMPLE PUMP CURRENT HIGH  "
1473            ;       EPN0 = "II-AS-1"
1474            ;       GRAFX0 = 20
1475            ;       GOTO AMSG
1476    1096    S128:   MSSG0 = "A-C100 AGITATOR SHUTDOWN      "
1477    1097            EPN0 = "YS-A-C100"
1478    1098            GRAFX0 = 18
1479    1099            GOTO AMSG
1480    1100    S129:   MSSG0 = "P-C100 PUMP SHUTDOWN          "
1481    1101            EPN0 = "YS-P-C100"
1482    1102            GRAFX0 = 18
1483    1103            GOTO AMSG
1484    1104    S130:   MSSG0 = "HANDVALVE 1-29 CLOSED         "
1485    1105            EPN0 = "ZS-IDX-2"
```

```
1486    1106            GRAFX0 = 18
1487    1107            GOTO AMSG
1488    1108    S131:   MSSG0 = "FC3 FILTER INLET PRESSURE HIGH"
1489    1109            EPN0 = "PSH-ILF-1"
1490    1110            GRAFX0 = 18
1491    1111            GOTO AMSG
1492    1112    S132:   MSSG0 = "HNDVLV 1-16 PC TO TK-C100 OPEN"
1493    1113            EPN0 = "ZS-RC3-4"
1494    1114            GRAFX0 = 18
1495    1115            GOTO AMSG
1496    1116    S133:   MSSG0 = "HNDVLV 1-17 PC TO TK-102AW OPN"
1497    1117            EPN0 = "ZS-RC3-5"
1498    1118            GRAFX0 = 18
1499    1119            GOTO AMSG
1500            S134:   ;       MSSG0 = "PROCESS CONDENS8 AGIT AMPS HI "
1501            ;       EPN0 = "II-A-C100"
1502            ;       GRAFX0 = 18
1503            ;       GOTO AMSG
1504    1120    S135:   MSSG0 = "TK-C-100 UPPER INTERFACE LOW  "
1505    1121            EPN0 = "LI-C100-1"
1506    1122            GRAFX0 = 18
1507    1123            GOTO AMSG
1508            S136:   ;MSSG0 = "PROCESS CONDENS8 PUMP AMPS LOW"       ;commented out, -mct5+
1509            ;       EPN0 = "II-P-C100"
1510            ;       GRAFX0 = 18
1511            ;       GOTO AMSG
1512    1124    S137:   MSSG0 = "F-CA1-L/R DELTA PRESS HI "
1513    1125            EPN0 = "PDI-CA1-3"
1514    1126            GRAFX0 = 12
1515    1127            GOTO AMSG
1516    1128    S138:   MSSG0 = "STM CONDSATE BASIN-1 LEVEL LOW"
1517    1129            EPN0 = "LSL-SC-1"
1518    1130            GRAFX0 = 22
1519    1131            GOTO AMSG
1520    1132    S139:   MSSG0 = "STM CONDSATE BASIN-2 LEVEL LOW"
1521    1133            EPN0 = "LSL-SC-2"
1522    1134            GRAFX0 = 22
1523    1135            GOTO AMSG
1524    1136    S140:   MSSG0 = "STM CONDSATE BASIN-3 LEVEL LOW"
1525    1137            EPN0 = "LSL-SC-3"
1526    1138            GRAFX0 = 22
1527    1139            GOTO AMSG
1528            S141:   ;       MSSG0 = "207-A SUMP PUMP P1 CURRENT HI "
1529            ;       EPN0 = "II-SUMPP1"
1530            ;       GRAFX0 = 22
1531            ;       GOTO AMSG
1532            S142:   ;       MSSG0 = "STEAM CONDENS8 PUMP P3 AMPS HI"
1533            ;       EPN0 = "II-SC-P3"
1534            ;       GRAFX0 = 999
1535            ;       GOTO AMSG
1536    1140    S143:   MSSG0 = "PROCESS AIR CMPRESSR NO. 1 OFF"
1537    1141            EPN0 = "YS-CPE1-1"
1538    1142            GRAFX0 = 21
1539    1143            GOTO AMSG
1540    1144    S144:   MSSG0 = "PROCESS AIR CMPRESSR NO. 2 OFF"
```

```
1541    1145            EPNO = "YS-CPE2-1"
1542    1146            GRAFXO = 21
1543    1147            GOTO AMSG
1544    1148    S145:   MSSGO = "INSTRUMENT AIR HEADER PRESS LO"
1545    1149            EPNO = "PI-IA-1"
1546    1150            GRAFXO = 21
1547    1151            GOTO AMSG
1548    1152    S146:   MSSGO = "INSTRUMENT AIR HEADER PRESS HI"
1549    1153            EPNO = "PI-IA-1"
1550    1154            GRAFXO = 21
1551    1155            GOTO AMSG
1552            S147:   ;       MSSGO = "PROCESS AIR HEADER PRESSURE LO"
1553            ;       EPNO = "PI-PA-1"
1554            ;       GRAFXO = 21
1555            ;       GOTO AMSG
1556    1156    S148:   MSSGO = "PROCESS AIR HEADER PRESSURE HI"
1557    1157            EPNO = "PI-PA-1"
1558    1158            GRAFXO = 21
1559    1159            GOTO AMSG
1560    1160    S149:   MSSGO = "COLD JNCTN AMBIENT SENSOR LOLO"
1561    1161            EPNO = "TI-AMBS-1"
1562    1162            GRAFXO = 21
1563    1163            GOTO AMSG
1564    1164    S150:   MSSGO = "COLD JNCTN AMBIENT SENSOR LOW "
1565    1165            EPNO = "TI-AMBS-1"
1566    1166            GRAFXO = 21
1567    1167            GOTO AMSG
1568    1168    S151:   MSSGO = "COLD JNCTN AMBIENT SENSOR HIGH"
1569    1169            EPNO = "TI-AMBS-1"
1570    1170            GRAFXO = 21
1571    1171            GOTO AMSG
1572    1172    S152:   MSSGO = "COLD JNCTN AMBIENT SENSOR HIHI"
1573    1173            EPNO = "TI-AMBS-1"
1574    1174            GRAFXO = 21
1575    1175            GOTO AMSG
1576    1176    S153:   MSSGO = "COLD JNCTN AMBIENT SENSOR LOLO"
1577    1177            EPNO = "TI-AMBS-2"
1578    1178            GRAFXO = 21
1579    1179            GOTO AMSG
1580    1180    S154:   MSSGO = "COLD JNCTN AMBIENT SENSOR LOW "
1581    1181            EPNO = "TI-AMBS-2"
1582    1182            GRAFXO = 21
1583    1183            GOTO AMSG
1584    1184    S155:   MSSGO = "COLD JNCTN AMBIENT SENSOR HIGH"
1585    1185            EPNO = "TI-AMBS-2"
1586    1186            GRAFXO = 21
1587    1187            GOTO AMSG
1588    1188    S156:   MSSGO = "COLD JNCTN AMBIENT SENSOR HIHI"
1589    1189            EPNO = "TI-AMBS-2"
1590    1190            GRAFXO = 21
1591    1191            GOTO AMSG
1592    1192    S157:   MSSGO = "ELUANT AGITATOR MOTOR CF-ON   "
1593    1193            EPNO = "YS-A-E101"
1594    1194            GRAFXO = 19
1595    1195            GOTO AMSG
```

```
1596  1196  S158:   MSSG0 = "ELUANT PUMP MOTOR CONFIRM ON  "
1597  1197          EPN0 = "YS-P-E101"
1598  1198          GRAFX0 = 19
1599  1199          GOTO AMSG
1600  1200  S159:   MSSG0 = "ANTIFOAM AGITATOR MOTOR CF-ON "
1601  1201          EPN0 = "YS-A-E102"
1602  1202          GRAFX0 = 19
1603  1203          GOTO AMSG
1604  1204  S160:   MSSG0 = "ANTIFOAM PUMP MOTOR CONFIRM ON"
1605  1205          EPN0 = "YS-P-E102"
1606  1206          GRAFX0 = 19
1607  1207          GOTO AMSG
1608  1208  S161:   MSSG0 = "DECON AGITATOR MOTOR CF-ON    "
1609  1209          EPN0 = "YS-A-104"
1610  1210          GRAFX0 = 19
1611  1211          GOTO AMSG
1612  1212  S162:   MSSG0 = "DECON PUMP MOTOR CONFIRM ON   "
1613  1213          EPN0 = "YS-P-E104"
1614  1214          GRAFX0 = 19
1615  1215          GOTO AMSG
1616  1216  S163:   MSSG0 = "TK-E-101 LIQUID TEMPERATURE HI"
1617  1217          EPN0 = "TI-E101-2"
1618  1218          GRAFX0 = 19
1619  1219          GOTO AMSG
1620  1220  S164:   MSSG0 = "TK-E-101 WEIGHT FACTOR LOLO   "
1621  1221          EPN0 = "WFI-E101"
1622  1222          GRAFX0 = 19
1623  1223          GOTO AMSG
1624  1224  S165:   MSSG0 = "TK-E-101 WEIGHT FACTOR LO     "
1625  1225          EPN0 = "WFI-E101"
1626  1226          GRAFX0 = 19
1627  1227          GOTO AMSG
1628  1228  S166:   MSSG0 = "TK-E-102 WEIGHT FACTOR LOLO   "
1629  1229          EPN0 = "WFI-E102"
1630  1230          GRAFX0 = 19
1631  1231          GOTO AMSG
1632  1232  S167:   MSSG0 = "TK-E-102 WEIGHT FACTOR LO     "
1633  1233          EPN0 = "WFI-E102"
1634  1234          GRAFX0 = 19
1635  1235          GOTO AMSG
1636  1236  S168:   MSSG0 = "TK-E-104 LIQUID TEMP HIGH     "
1637  1237          EPN0 = "TI-E104-2"
1638  1238          GRAFX0 = 19
1639  1239          GOTO AMSG
1640  1240  S169:   MSSG0 = "TK-E-104 WEIGHT FACTOR LOLO   "
1641  1241          EPN0 = "WFI-E104"
1642  1242          GRAFX0 = 19
1643  1243          GOTO AMSG
1644  1244  S170:   MSSG0 = "TK-E-104 WEIGHT FACTOR LO     "
1645  1245          EPN0 = "WFI-E104"
1646  1246          GRAFX0 = 19
1647  1247          GOTO AMSG
1648  1248  S171:   MSSG0 = "LERF NORTH GATE OPEN          "
1649  1249          EPN0 = "ZS-105PS1"
1650  1250          GRAFX0 = 24
```

```
1651    1251            GOTO AMSG
1652    1252    S172:   MSSG0 = "LERF SOUTH GATE OPEN          "
1653    1253            EPN0 = "ZS-105PS2"
1654    1254            GRAFX0 = 24
1655    1255            GOTO AMSG
1656    1256    S173:   MSSG0 = "LERF LEACHATE PUMP 42 ON      " ; +mct4-
1657    1257            EPN0 = "YS-P-42"
1658    1258            GRAFX0 = 24
1659    1259            GOTO AMSG
1660    1260    S174:   MSSG0 = "LERF LEACHATE PUMP 43 ON      " ; +mct4-
1661    1261            EPN0 = "YS-P-43"
1662    1262            GRAFX0 = 24
1663    1263            GOTO AMSG
1664    1264    S175:   MSSG0 = "LERF LEACHATE PUMP 44 ON      " ; +mct4-
1665    1265            EPN0 = "YS-P-44"
1666    1266            GRAFX0 = 24
1667            ;
1668            ;       END STRINGS COLLECTION FOR ALARM SUBROUTINE
1669            ;
1670            ;       BEGIN MESSAGE CREATION FOR ALARM SUBROUTINE:
1671            ;
1672    1267    AMSG:   IN1 = TIMSTR(TIME0)
1673    1269            IF IN1 != 8 THEN TIME0 = "TIMEFAIL"
1674    1270            IN1 = DATSTR(DATE0)
1675    1272            IF IN1 != 9 THEN DATE0 = "DATE FAIL"
1676    1273            IN1 = APPEND(LSG0,DATE0," ",TIME0,"  ",EPN0)
1677    1274            LET PALARM = 1
1678            ;
1679    1275            ASSIGN PLINE = INT1
1680    1276            FOR JLOOP = 1 TO 175            ; +mct4-
1681    1277            IF PLINE THEN
1682                       BEGIN
1683    1278               PLINE = PLINE + 1
1684            ;          PRINT 0,"     ~~~ ALM: WHERE PLINE = PLINE + 1,  PLINE = %i, JLOOP = %i",pline,jloop
1685                       END
1686    1279            IF JLOOP = ILOOP THEN
1687                       BEGIN
1688    1280               IF(PLINE)THEN
1689                          BEGIN
1690            ;             PRINT 0,"  DURING ALM, PLINE LAERDY > 0, PLINE = %i",PLINE
1691    1281                  GOTO TERM
1692                          END
1693    1282               PLINE = 1
1694                       END
1695    1283            STEP PLINE BY 1
1696    1284            NEXT JLOOP
1697            ;
1698            ;       END MESSAGE CREATION FOR ALARM SUBROUTINE:
1699            ;
1700    1285            ASSIGN PLINE = INT0
1701    1286            STEP PLINE BY ILOOP
1702    1287            PLINE = 1
1703    1288            RETURN
1704            ;
1705            ;       END ALARM SUBROUTINE
```

```
1706            ;
1707            ;       BEGIN ERRORS SUBROUTINE:
1708            ;
1709    1289    ERRSUB:  IF ERRN != ERRNUM() THEN
1710                      BEGIN
1711    1290              TIMERON STMR0
1712    1291              TIMERCLR STMR0
1713    1292              FSPAR0 = 1
1714    1293              ERRN = ERRNUM()
1715    1294              ERRL = ERRLIN()
1716            ;         PRINT ADI,"ERROR # %i AT LINE %i",errn,errl
1717                      END
1718    1295             ERSCLR
1719    1296             WAIT 5
1720    1297             GOTO AGAIN
1721            ;
1722            ;       END ERRORS SUBROUTINE
1723            ;
1724            ;       TERMINATE SURBOUTINE:
1725            ;
1726    1298    TERM:    FABORT = 0
1727    1299             PRINT 0,"       ~~~ WHITE1 TERMINATED"
1728                     END

        Total Errors Detected      0     Total Warnings      0
             Sequence Program Size = 27712 Bytes, 1732 Pages
                  314 Forward Reference(s)
```

ANBATCH

Sequence Instruction Compiler Listing 15:32:44 24-Feb-95 Page 1

```
1           ;       A N B A T C H . S E Q                                   VERSION 2
2           ;
3           ;       Function:       Sequence program for Lamp function,
4           ;                       message display on graphics, and
5           ;                       printer messages.
6           ;
7           ;       Written By      : R. N. Kersey
8           ;       Date            : 05-31-92
9           ;       Facility        : 242-A Evaporator Bldg.
10          ;       Machine         : VAX 4104-A
11          ;       Op. System      : VAX/VMS 5.5-2AH
12          ;       D/3 Version     : 6.3-2 BASELINE
13          ;       Compiler        : SIC
14          ;       Build command   : @[320,20]ANBATCH
15          ;
16          ;       UNIT = 1, SKDKEY = 0, 1, 2, 3
17          ;
18          ;       Assoc. UPF:     [300,1]_TNKFRM.___      TNKFRM.LST
19          ;       Assoc. RPF:     [300,1]_FARM.___        FARM.LST
20          ;       SDB:            [300,1]PCM1ASDB.___     PCM1ASDB.CMD
21          ;
22          ;       POINTER   SDB VAR.      FUNCTION
23          ;       -------   ---------     ----------------------------------------
24          ;       PSTAT     INTn          Current EPN alarm state is an integer tied to
25          ;                               graphic flashing string of value:
26          ;       .         (n=0-100)             0 = previously not alarmed
27          ;       .                               1 = previously alarmed, not ack
28          ;       .                               2 = previously alarmed, is ack
29          ;       .                               3 = alarm is inhibited
30          ;       .                               4 = input is off scan
31          ;       .                               5 = input is bad
32          ;       PDIN      'EPN'         ---
33          ;       PTYPE     IALMn         alarm type 0 = none, 1 = white, 2 = yellow,
34          ;                                 3 = yellow and yellow, 4 = red,
35          ;                                 6 = yellow and red
36          ;       PREV      FXn           Reverse acting (not fail safe) flag (1=set)
37          ;       PSKID     SKIDn         Tracks which graphic has an alarm
38          ;       PPASS     FPASn         If set (1), turn lamp off when alarm is ACKnowledged
39          ;       PSN       FSNn          Flag for the graphic to ACKnowledge
40          ;       ---       FHOLD         Flag to update the lamps
41          ;       ---       FNING         Flag for DOG1A.C micro select
42          ;       ---       ACK           Flag for C program to ACK system wide
43          ;
44          ; MODIFICATION LOG:
45          ;
46          ;       05-APR-93       Modified MCC OFF white alarm on all 7           M. Teats
47          ;                       YS- signals to pump OFF white alarm because
48          ;                       the YS DINs do not indicate MCC status - they
49          ;                       indicate the pump status.
50          ;
51          ;-mct1+  13-MAY-93      Remove pump shutdown DEV white alarms, which    M. Teats
52          ;                       are based on the analog signals, because they
53          ;                       are not needed due to the 4/5/93 mods.  See
54          ;                       CCN# 93MAY13-01.
55          ;
```

```
56          ;       15-JUN-93       Added comments and baselined for        M. Teats
57          ;                       D/3 version 3.3.  CCN# 93JUN01-01.
58          ;
59          ;       15-DEC-94       Revised header to indicate the new VMS          J. Berger
60          ;                       system. Rebaselined document. Removed
61          ;                       obsolite Select() statements.
62          ;
63          ;  END MOD LOG
64          ;
65          UNIT    ANFARM
66          ;
67                  PUMP == FLSPIT
68                  AUTO == 'FQI-AN-1':CB_FLG:1             ; For FQI-RESET auto mode check
69                  CALOUT == 'FQI-AN-1':CB_OTVL:1          ; For FQI-RESET calc block outvalue
70          ;
71  0000            GETUNIT TRIP                            ; Which sequence unit is program running in
72  0001            IF TRIP != 1 THEN                       ; Abort if running in wrong seq. unit
73                     BEGIN
74  0002               FSTART = 1
75  0003               ILOOP = 54
76  0004               GOSUB CYCL
77  0005               FSTART = 1
78  0006               PRINT ADI," >>>>>  ATTEMPT TO RUN ANBATCH IN WRONG UNIT"
79  0007               GOTO TERM
80                     END
81          ;       IF !SELECT() THEN GOTO AGAIN
82          ;
83          ;       B E G I N   I N I T I A L I Z I N G :
84          ;
85  0008            NOP
86  0009    INIT:   LET FABORT = 0                  ; If FABORT = 1 then abort program
87  0010            LET FSTART = 0                  ; Program start flag
88  0011            LET FLDALM = 0
89  0012            LET FSL7 = 0                    ; For FQI-RESET manual mode flag
90  0013            LAMP 0,3,0                      ; Turn off the lamps
91  0014            LAMP 0,2,0
92  0015            LAMP 0,1,0
93  0016            LAMP 0,0,0
94  0017            ASSIGN PDEV = ANFARM                            ; First of 5 alarm ack devices
95  0018            FOR ILOOP = 1 TO 5
96  0019             PDEV:DEV_TMP0 = 1                              ; Set all 5 devices to ack state
97  0020            IF AND(PDEV:DEV_ALMCT,16) THEN                  ; If alarm device is inha, remove the inha
98  0021               PDEV:DEV_ALMCT=XOR(PDEV:DEV_ALMCT,16)
99  0022            IF AND(PDEV:DEV_ALMCT,32) THEN                  ; If alarm device is ack, clear it
100 0023               PDEV:DEV_ALMCT = XOR(PDEV:DEV_ALMCT,32)
101 0024            STEP PDEV BY 1
102 0025            NEXT ILOOP
103         ;
104         ;       First, set all flags and integers:
105         ;
106 0026            ASSIGN PDIN = 'YS-AN-101'                       ; Assign pointer to first DIN
107 0027            ASSIGN PSTAT = INT0
108 0028            ASSIGN PTYPE = IALM0
109 0029            ASSIGN PREV = FX0
110 0030            ASSIGN PSKID = SKID0
```

```
111   0031          ASSIGN PPASS = FPASO
112                 ;
113   0032          LET FARM = 0                              ; Set up values for different graphics
114   0033          SERPIT = 1                                ; SKID of each EPN indicates graphic
115   0034          VLVPIT = 2
116   0035          HVAC = 3
117   0036          BLD271 = 4
118                 ;
119   0037          LEAK0 = 3                                 ; FARM
120   0038          LEAK1 = 64                                ; SERVICE PIT ( NOT USED )
121   0039          LEAK2 = 1                                 ; VALVE PIT
122   0040          LEAK3 = 2                                 ; HVAC
123   0041          LEAK4 = 0                                 ; 271
124                 ;
125   0042          FOR ILOOP = 0 TO 44                       ; For DINs
126   0043          LET PPASS = 0
127   0044          LET PREV = 0
128   0045          LET PTYPE = 2
129                 ;       BLET PSTAT = 0
130   0046          IF AND(PDIN:DIN_DFLG,3) THEN              ; Check for alarms (must be red or yellow)
131                    BEGIN
132   0047             LET PTYPE = 2                          ; Default to yellow alarm
133   0049             IF AND(PDIN:DIN_DFLG,1) THEN LET PREV = 1
                                                                    ; Reverse acting
134   0051             IF AND(PDIN:DIN_DFLG,4) THEN LET PTYPE = 4
                                                                    ; Set to red critical alarm
135                    END
136   0053          IF ILOOP < 21 THEN LET PSKID = FARM       ; These are set by IPN order
137   0055          IF (ILOOP > 20 & ILOOP < 29) THEN LET PSKID = VLVPIT
138   0057          IF (ILOOP > 28 & ILOOP < 37) THEN LET PSKID = HVAC
139   0059          IF ILOOP > 36 THEN LET PSKID = BLD271
140   0060          IF AND(PDIN:DIN_ALMCT,32) THEN
141   0061             PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
142   0062    SKP:  STEP PSTAT BY 1
143   0063          STEP PDIN BY 1
144   0064          STEP PTYPE BY 1
145   0065          STEP PREV BY 1
146   0066          STEP PSKID BY 1
147   0067          STEP PPASS BY 1
148   0068          NEXT ILOOP
149                 ;
150                 ;-mct1-  Abandoned code for white alarms tied to pump devices:
151                 ;
152                 ;       FOR ILOOP = 45 TO 51
153                 ;       BLET PSTAT = 0
154                 ;       BLET PTYPE = 1
155                 ;       BLET PREV = 0
156                 ;       BLET PSKID = 0
157                 ;       BLET PPASS = 1
158                 ;       STEP PSTAT BY 1
159                 ;       STEP PTYPE BY 1
160                 ;       STEP PREV BY 1
161                 ;       STEP PSKID BY 1
162                 ;       STEP PPASS BY 1
163                 ;       NEXT ILOOP
164                 ;-mct1-
165                 ;
```

```
166                     ;  Set the white alarms:
167                     ;
168     0069            LET IALM0 = 1
169     0070            LET IALM3 = 1
170     0071            LET IALM6 = 1
171     0072            LET IALM9 = 1
172     0073            LET IALM12 = 1
173     0074            LET IALM15 = 1
174     0075            LET IALM18 = 1
175     0076            LET IALM42 = 1
176                     ;
177                     ;  Pumps alarm on 0, No lamp when Ack:
178                     ;
179     0077            FPAS0 = 1
180     0078            FPAS3 = 1
181     0079            FPAS6 = 1
182     0080            FPAS9 = 1
183     0081            FPAS12 = 1
184     0082            FPAS15 = 1
185     0083            FPAS18 = 1
186             ;
187             ;       E N D   I N I T I A L I Z I N G
188             ;
189             ;       S E N D   P R O G R A M   S T A R T   M E S S A G E
190             ;
191     0084            ILOOP = 53
192     0085            FSTART = 1
193     0086            TIMERCLR STMR2
194     0087            TIMERON STMR2
195     0088            GOSUB CYCL
196     0089            ON ERROR GOSUB ERRSUB
197             ;
198             ;       B E G I N   P R O G R A M   L O O P :
199             ;
200     0090    AGAIN:  MARK STEP "AGAIN",0,0
201             ;       IF !SELECT() THEN FNING = 0
202             ;          ELSE FNING = 1                    ; Flag 480 determines if selected micro for DOG1A.C
203             ;       WAITUNTIL SELECT() ¦ FABORT          ; Wait until micro selected or program aborted
204             ;       IF (FABORT) THEN GOTO TERM
205             ;       IF FNING = 0 THEN FHOLD = 1
206     0092            IF STMR2 > 30 THEN FHOLD = 1
207     0093            WAIT 1
208             ;
209     0094            DEV0 = 0                              ; Lamp 3
210     0095            DEV1 = 0
211     0096            DEV2 = 0                              ; Lamp 1
212     0097            DEV3 = 0                              ; Lamp 2
213     0098            DEV4 = 0                              ; Lamp 0
214             ;
215             ;       LOOP CHECKS IF ANY OF 5 ALARM DEVICES HAS BEEN ACKNOWLEDGED.
216             ;       IF ALARM DEVICE ACKNOWLEDGED AND DEV_TMP0 = 2 THEN ACKNOWLEDGE
217             ;       EVERY SIGNAL ON THAT GRAPHIC
218             ;
219             ;               DEVICE          FLAG            OFFSET
220             ;               ------          ------          ------
```

```
221             ;               ANFARM          FLEAK0          447
222             ;               AN-SERPIT       FLEAK1          448     (NOT USED)
223             ;               AN-VLVPIT       FLEAK2          449
224             ;               AN-HVAC         FLEAK3          450
225             ;               AN-271          FLEAK4          451
226             ;
227     0099            ASSIGN PDEV = ANFARM
228     0100            ASSIGN PSN = FSN0
229     0101            ASSIGN PACK = FLEAK0
230     0102            FOR ILOOP = 0 TO 4
231     0103            LET PSN = 0
232     0104            IF (PDEV:DEV_TMP0 = 2) & AND(PDEV:DEV_ALMCT,32) THEN
233                        BEGIN
234     0105               PDEV:DEV_TMP0 = 1                    ; Put device in ack state
235     0106               LET PSN = 1                          ; Set flag for the graphic to ack
236     0107               LET FLGACK = 1                       ; WATCH.C spawns ACKFLG.C, which resets flag
237     0108               LET ACK = 1                          ; Set flag for C program to ACK system wide
238     0109               LET PACK = 1
239     0110               ASSIGN PMSG0 = LEAK0
240     0111               STEP PMSG0 BY ILOOP
241     0113               IF ILOOP != 1 THEN LAMP 0,PMSG0,1    ; Flash Lamp
242                        END
243     0114            STEP PSN BY 1
244     0115            STEP PACK BY 1
245     0117            IF ILOOP < 6 THEN STEP PDEV BY 1 ; Only 5 devices
246     0118            WAIT 2
247     0119            NEXT ILOOP
248             ;
249             ;       FIELD ALARM SILENCE FEATURE REMOVED FROM ANFARM
250             ;       UNTIL 271-AN PANEL BOARD WIRING RESOLVED (1/24/92 RNK):
251             ;
252             ;       IF (ACK) THEN
253             ;         BEGIN
254             ;         BTIMERCLR STMR1
255             ;         BTIMERON STMR1
256             ;         BLET FLDALM = 1
257             ;         END
258             ;       IF (FLDALM) THEN IF STMR1 >= 30 THEN     ; If field alarm then ack in 30 sec
259             ;         BEGIN
260             ;         BTIMEROFF STMR1
261             ;         BTIMERCLR STMR1
262             ;         BLET FLDALM = 0
263             ;         OPEN 'HS-AN-SIL'                       ; Silence alarm from PCM
264             ;         END
265                     ;
266                     ;  If field alarm ack then set flag and start timer:
267                     ;
268     0121            IF (RESET) THEN IF LMPVAR < 2 THEN
269                        BEGIN
270     0122               LET FLDALM = 1
271     0123               TIMERON STMR1
272     0124               LET RESET = 0
273                        END
274     0126            IF LMPVAR = 2 THEN LET RESET = 1
275             ;
```

```
276           ;       BEGIN DINs:
277           ;
278    0127           ASSIGN PDIN = 'YS-AN-101'                ; First DIN in ANFARM
279    0128           ASSIGN PTYPE = IALM0
280    0129           ASSIGN PREV = FX0
281    0130           ASSIGN PSTAT = INT0
282    0131           ASSIGN PSKID = SKID0
283    0132           ASSIGN PPASS = FPAS0
284    0133           CURENT = 0                               ; Used to set lamp status
285    0134           NXT = 0                                  ; Temproary INT
286    0135           BITVAL = 0
287           ;
288    0136    LOP:   FOR ILOOP = 1 TO 45
289    0137           IF PTYPE THEN                                        ; If some alarm
290                      BEGIN                                        ;be1
291    0139              IF AND(PDIN:DIN_ALMCT,16) THEN LET PSTAT = 3   ; If EPN is inha
292    0141              IF AND(PDIN:DIN_ALMCT,4) THEN LET PSTAT = 5          ; If EPN is ibad
293    0143              IF !AND(PDIN:DIN_FLG,8) THEN LET PSTAT = 4           ; If EPN is dis
294    0146              IF ! PSTAT THEN IF PREV = PDIN:DIN_VAL THEN GOSUB ALM  ; If not previously alarmed
295    0147              IF PSTAT THEN
296                         BEGIN                                     ;be2
297    0148                 IF PSTAT < 3 THEN                                 ; If not inha
298                            BEGIN                                  ;be3
299    0149                    IF PSTAT = 1 THEN
300                               BEGIN                               ;be4
301    0150                       IF AND(PDIN:DIN_ALMCT,32) THEN              ; If alarm is ack
302                                  BEGIN                            ;be5
303    0151                          LET PSTAT = 2
304    0152                          FHOLD = 1
305                                  END                              ;en5
306    0153                       IF ACK THEN                                 ; If a graphic has been ack
307                                  BEGIN                            ;be6  -  go find which graphic
308    0154                          CRIT
309    0155                          ASSIGN PSN = FSN0                ; Routine to find graphic
310    0156                          STEP PSN BY PSKID
311    0157                          IF PSN THEN                             ; If this is the graphic
312                                     BEGIN                         ;be7  -  then ack it
313    0158                             LET PSTAT = 2
314    0160                             IF PTYPE = 1 THEN PDIN:DIN_ALMCT = OR(PDIN:DIN_ALMCT,32)
315    0161                             FPASS = 1
316    0162                             GOSUB ACKK                          ; Print signal was ack
317                                     END                           ;en7
318                                  END                              ;en6
319                               END                                 ;en4
320    0164                       IF PREV != PDIN:DIN_VAL THEN GOSUB CLR      ; If alarm has cleared
321    0165                       ASSIGN PMSG0 = DEV0                         ; Find active alarms & graphic:
322    0166                       STEP PMSG0 BY PSKID
323    0167                       IF PPASS THEN
324                                  BEGIN
325    0169                          IF PSTAT = 1 THEN PMSG0 = OR(PMSG0,PSTAT)
326                                  END
327    0170                       ELSE PMSG0 = OR(PMSG0,PSTAT)
328                            END                                    ;en3
329    0171                 ELSE IF !AND(PDIN:DIN_ALMCT,16)&!AND(PDIN:DIN_ALMCT,4)&AND(PDIN:DIN_FLG,8)THEN
330                            BEGIN
```

```
331  0172                    LET PSTAT = 0
332  0173                    FHOLD = 1
333                          END
334                       END                                          ;en2
335                    END                                             ;en1
336  0175           ELSE IF PSTAT THEN PSTAT = 0
337  0176           STEP PDIN BY 1
338  0177           STEP PTYPE BY 1
339  0178           STEP PREV BY 1
340  0179           STEP PSTAT BY 1
341  0180           STEP PSKID BY 1
342  0181           STEP PPASS BY 1
343  0182           NEXT ILOOP
344          ;
345          ;       END DINs
346          ;
347  0183           DEV6 = 0
348  0184           DEV7 = 0
349  0185           DEV8 = 0
350  0186           DEV9 = 0
351  0187           DEV10 = 0
352  0188           DEV11 = 0
353          ;
354          ;-mct1-  Abandoned code for white alarms tied to pump devices:
355          ;
356          ;AUX:    ASSIGN PEPN = 'P-AN-101'
357          ;
358          ;       FOR ILOOP = 46 TO 52
359          ;       IF PTYPE THEN
360          ;          BEGIN
361          ;          IF AND(PEPN:DEV_ALMCT,16)THEN BLET PSTAT = 3
362          ;          IF AND(PEPN:DEV_ALMCT,4) THEN BLET PSTAT = 5
363          ;          IF !AND(PEPN:DEV_FLG,8) THEN BLET PSTAT = 4
364          ;          IF PSTAT < 3 THEN
365          ;             BEGIN
366          ;             IF (PSTAT = 1) & AND(PEPN:DEV_ALMCT,32)THEN
367          ;                BEGIN
368          ;                BLET PSTAT = 2
369          ;                FHOLD = 1
370          ;                END
371          ;             IF ACK THEN
372          ;                BEGIN
373          ;                ASSIGN PSN = FSNO
374          ;                STEP PSN BY PSKID
375          ;                IF PSN & PSTAT = 1 THEN
376          ;                   BEGIN
377          ;                   BLET PSTAT = 2
378          ;                   IF PTYPE = 1 THEN PEPN:DEV_ALMCT = OR(PEPN:DEV_ALMCT,32)
379          ;                   FPASS = 1
380          ;                   GOSUB ACKK
381          ;                   END
382          ;                END
383          ;             IF !PSTAT THEN
384          ;                BEGIN
385          ;                IF AND(PEPN:DEV_STAT,16)THEN
```

```
386             ;                   BEGIN
387             ;                   BLET PSTAT = 1
388             ;                    IF AND(PEPN:DEV_ALMCT,32)THEN PEPN:DEV_ALMCT = XOR(PEPN:DEV_ALMCT,32)
389             ;                   ASSIGN PDEV = ANFARM
390             ;                    STEP PDEV BY PSKID
391             ;                    PDEV:DEV_TMPO = 2
392             ;                    GOSUB CYCL
393             ;                    END
394             ;                 END
395             ;              ELSE IF !AND(PEPN:DEV_STAT,16)THEN
396             ;                 BEGIN
397             ;                 BLET PSTAT = 0
398             ;                 IF AND(PEPN:DEV_ALMCT,32)THEN PEPN:DEV_ALMCT = XOR(PEPN:DEV_ALMCT,32)
399             ;                 FHOLD = 1
400             ;                 END
401             ;              IF PSTAT THEN
402             ;                 BEGIN
403             ;                 ASSIGN PMSG0 = DEV6
404             ;                 STEP PMSG0 BY PSKID
405             ;                 IF PPASS THEN
406             ;                    BEGIN
407             ;                    IF PSTAT = 1 THEN PMSG0 = OR(PMSG0,PSTAT)
408             ;                    END
409             ;                 ELSE PMSG0 = OR(PMSG0,PSTAT)
410             ;                 END
411             ;              END
412             ;           ELSE IF !AND(PEPN:DEV_ALMCT,16)&!AND(PEPN:DEV_ALMCT,4)&AND(PEPN:DEV_FLG,8)THEN PSTAT = 0
413             ;           END
414             ;        STEP PEPN BY 1
415             ;        STEP PTYPE BY 1
416             ;        STEP PREV BY 1
417             ;        STEP PSTAT BY 1
418             ;        STEP PSKID BY 1
419             ;        STEP PPASS BY 1
420             ;        NEXT ILOOP
421             ;-mct1-
422             ;
423     0189    S2ND:    ACK = 0
424     0190             SORT = 0
425             ;
426     0191             DEV12 = OR(DEV0,DEV6)
427     0192             DEV13 = OR(DEV1,DEV7)
428     0193             DEV14 = OR(DEV2,DEV8)
429     0194             DEV15 = OR(DEV3,DEV9)
430     0195             DEV16 = OR(DEV4,DEV10)
431     0196             DEV17 = OR(DEV5,DEV11)
432             ;
433     0197             ASSIGN PMSG0 = DEV12
434     0198             ASSIGN PMSG1 = FARM
435     0199             FOR LLOOP = 1 TO 5
436     0200             IF PMSG0 THEN
437                        BEGIN
438     0202               IF AND(PMSG0,1) THEN PMSG1 = 2
439     0203               ELSE PMSG1 = 1
440                        END
```

```
441   0204          ELSE PMSG1 = 0
442   0205          STEP PMSG0 BY 1
443   0206          STEP PMSG1 BY 1
444   0207          NEXT LLOOP
445         ;
446   0208          IF (FHOLD = 1) THEN
447                    BEGIN
448   0209             LAMP 0,3,FARM
449   0210             LAMP 0,2,HVAC
450   0211             LAMP 0,1,VLVPIT
451   0212             LAMP 0,0,BLD271
452   0213             FHOLD = 0
453   0214             TIMERCLR STMR2
454   0215             TIMERON STMR2
455                    END
456   0216          MARK STEP "BOTTOM",0,0
457   0217          NOCRIT
458         ;
459         ;       B E G I N   F Q I - R E S E T :
460         ;
461         ;       Function:       For totalizers that have a CALC block which performs the
462         ;                       integration, set the output of the CALC block to zero
463         ;                       after the FQI loop has been placed in AUTO from MANUAL.
464         ;
465   0219          IF !AND(AUTO,64) THEN FSL7 = 1                  ; If manual, set manual flag
466   0220          IF AND(AUTO,64) & AND(FSL7,1) THEN              ; For manual-auto transition
467                    BEGIN
468   0221             FSL7 = 0
469   0222             CALOUT = 0.0
470                    END
471         ;
472         ;       E N D   F Q I - R E S E T
473         ;
474   0223          GOTO AGAIN
475         ;
476         ;       E N D   P R O G R A M   L O O P
477         ;
478         ;       BEGIN CLEAR, ALARM, CYCLE, ACKNOWLEDGE, MESSAGE, ERROR, AND TERMINATE SUBROUTINES:
479         ;
480   0224  CLR:    LET PSTAT = 0
481   0226          IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
482   0227          FHOLD = 1
483   0228          RETURN
484         ;
485   0229  ALM:    LET PSTAT = 1           ; Print message, flash lamp, horn on, scroll message at bottom of graphic
486   0230          CRIT                                            ; Critical
487   0232          IF AND(PTYPE,3)THEN LET FALMON = 1              ; Audible alarm on
488   0233          IF AND(PDIN:DIN_ALMCT,32)THEN                   ; Clear the scan block
489   0234             PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
490   0235          ASSIGN PMSG0 = LEAK0
491   0236          STEP PMSG0 BY PSKID
492   0237          LAMP 0,PMSG0,2                                  ; Lamp flash
493   0238          ASSIGN PDEV = ANFARM                            ; Find DEV and turn it on
494   0239          STEP PDEV BY PSKID
495   0240          PDEV:DEV_TMP0 = 2
```

```
496           ;
497           ;       Scrolling alarm summary region in graphic (message, time, date, alarm type):
498           ;
499     0241  CYCL:   LET LSG4 = LSG3                          ; LSG = message
500     0242          LET LSG3 = LSG2
501     0243          LET LSG2 = LSG1
502     0244          LET LSG1 = LSG0
503     0245          LET COLOR4 = COLOR3                            ; COLOR = type of alarm
504     0246          LET COLOR3 = COLOR2
505     0247          LET COLOR2 = COLOR1
506     0248          LET COLOR1 = COLOR0
507     0249          LET EPNS4 = EPNS3                              ; EPN
508     0250          LET EPNS3 = EPNS2
509     0251          LET EPNS2 = EPNS1
510     0252          LET EPNS1 = STRNG1
511     0253          LET GRAFX4 = GRAFX3                            ; GRAFX = graphic number
512     0254          LET GRAFX3 = GRAFX2
513     0255          LET GRAFX2 = GRAFX1
514     0256          LET GRAFX1 = GRAFX0
515     0257          LET SDATE4 = SDATE3                            ; SDATE = time and date
516     0258          LET SDATE3 = SDATE2
517     0259          LET SDATE2 = SDATE1
518     0260          LET SDATE1 = SDATE0
519           ;
520           ;       ALARM TYPER STRINGS:
521           ;
522     0261  ACKK:   FHOLD = 1
523     0262          ON ILOOP GOTO S1,S2,S3,S4,S5,S6,S7,S8,S9,S10
524     0263          ON ILOOP - 10 GOTO S11,S12,S13,S14,S15,S16,S17,S18,S19,S20
525     0264          ON ILOOP - 20 GOTO S21,S22,S23,S24,S25,S26,S27,S28,S29,S30
526     0265          ON ILOOP - 30 GOTO S31,S32,S33,S34,S35,S36,S37,S38,S39,S40
527     0266          ON ILOOP - 40 GOTO S41,S42,S43,S44,S45,S46,S47,S48,S49,S50
528     0267          ON ILOOP - 50 GOTO S51,S52,S53,S54,S55
529     0268          GOTO AMSG
530     0269  S1:     LET STRNG0 = "PUMP AN-101 OFF       "
531     0270          LET STRNG1 ="YS-AN-101"
532     0271          GOTO AMSG
533     0272  S2:     LET STRNG0 = "AN-101 ANN RD HI/FAIL "
534     0273          LET STRNG1 ="RSH-AN101"
535     0274          GOTO AMSG
536     0275  S3:     LET STRNG0 = "AN-01C RAD HI/FAIL    "
537     0276          LET STRNG1 = "RSH-AN01C"
538     0277          GOTO AMSG
539     0278  S4:     LET STRNG0 = "PUMP AN-102 OFF       "
540     0279          LET STRNG1 = "YS-AN-102"
541     0280          GOTO AMSG
542     0281  S5:     LET STRNG0 = "AN-102 ANN RD HI/FAIL "
543     0282          LET STRNG1 = "RSH-AN102"
544     0283          GOTO AMSG
545     0284  S6:     LET STRNG0 = "AN-02C RADN HI/FAIL   "
546     0285          LET STRNG1 = "RSH-AN02C"
547     0286          GOTO AMSG
548     0287  S7:     LET STRNG0 = "PUMP AN-103 OFF       "
549     0288          LET STRNG1 = "YS-AN-103"
550     0289          GOTO AMSG
```

```
551  0290  S8:    LET STRNG0 = "AN-103 ANN RD HI/FAIL "
552  0291          LET STRNG1 = "RSH-AN103"
553  0292          GOTO AMSG
554  0293  S9:    LET STRNG0 = "AN-03C RADN HI/FAIL   "
555  0294          LET STRNG1 = "RSH-AN03C"
556  0295          GOTO AMSG
557  0296  S10:   LET STRNG0 = "PUMP AN-104 OFF       "
558  0297          LET STRNG1 = "YS-AN-104"
559  0298          GOTO AMSG
560  0299  S11:   LET STRNG0 = "AN-104 ANN RD HI/FAIL "
561  0300          LET STRNG1 = "RSH-AN104"
562  0301          GOTO AMSG
563  0302  S12:   LET STRNG0 = "AN-04C RADN HI/FAIL   "
564  0303          LET STRNG1 = "RSH-AN04C"
565  0304          GOTO AMSG
566  0305  S13:   LET STRNG0 = "PUMP AN-105 OFF       "
567  0306          LET STRNG1 = "YS-AN105"
568  0307          GOTO AMSG
569  0308  S14:   LET STRNG0 = "AN-105 ANN RD HI/FAIL "
570  0309          LET STRNG1 = "RSH-AN105"
571  0310          GOTO AMSG
572  0311  S15:   LET STRNG0 = "AN-05C RADN HI/FAIL   "
573  0312          LET STRNG1 = "RSH-AN05C"
574  0313          GOTO AMSG
575  0314  S16:   LET STRNG0 = "PUMP AN-106 OFF       "
576  0315          LET STRNG1 = "YS-AN-106"
577  0316          GOTO AMSG
578  0317  S17:   LET STRNG0 = "AN-106 ANN RD HI/FAIL "
579  0318          LET STRNG1 = "RSH-AN106"
580  0319          GOTO AMSG
581  0320  S18:   LET STRNG0 = "AN-06C RADN HI/FAIL   "
582  0321          LET STRNG1 = "RSH-AN06C"
583  0322          GOTO AMSG
584  0323  S19:   LET STRNG0 = "PUMP AN-107 OFF       "
585  0324          LET STRNG1 = "YS-AN-107"
586  0325          GOTO AMSG
587  0326  S20:   LET STRNG0 = "AN-107 ANN RD HI/FAIL "
588  0327          LET STRNG1 = "RSH-AN107"
589  0328          GOTO AMSG
590  0329  S21:   LET STRNG0 = "AN-07C RADN HI/FAIL   "
591  0330          LET STRNG1 = "RSH-AN07C"
592  0331          GOTO AMSG
593  0332  S22:   LET STRNG0 = "AN SERVICE PIT LEAK   "
594  0333          LET STRNG1 = "LDS-SP-AN"
595  0334          GOTO AMSG
596  0335  S23:   LET STRNG0 = "AN SERPIT RAD HI/FAIL "
597  0336          LET STRNG1 = "RSH-SP-AN"
598  0337          GOTO AMSG
599  0338  S24:   LET STRNG0 = "AN PROCESS PITS LEAK  "
600  0339          LET STRNG1 = "LDS-AN-PP"
601  0340          GOTO AMSG
602  0341  S25:   LET STRNG0 = "AN SL-160/SL-168 LEAK "
603  0342          LET STRNG1 = "LDS-ANSL1"
604  0343          GOTO AMSG
605  0344  S26:   LET STRNG0 = "AN SN-260/SN-268 LEAK "
```

```
606  0345          LET STRNG1 = "LDS-ANSN2"
607  0346          GOTO AMSG
608  0347  S27:    LET STRNG0 = "AN CLEANOUT BOX LEAK  "
609  0348          LET STRNG1 = "LDS-COBAN"
610  0349          GOTO AMSG
611  0350  S28:    LET STRNG0 = "AN FLUSHLINE PRESS HI "
612  0351          LET STRNG1 = "PSH-ANSN1"
613  0352          GOTO AMSG
614  0353  S29:    LET STRNG0 = "AN DR-368 ENCASE LEAK "
615  0354          LET STRNG1 = "LDS-DR368"
616  0355          GOTO AMSG
617  0356  S30:    LET STRNG0 = "AN TANK EXH RADN HI   "
618  0357          LET STRNG1 = "RSH-AN-K1"
619  0358          GOTO AMSG
620  0359  S31:    LET STRNG0 = "AN K1 STKMON SMPL PMP "
621  0360          LET STRNG1 = "PSH-ANK11"
622  0361          GOTO AMSG
623  0362  S32:    LET STRNG0 = "AN K1 REC SMPL PUMP   "
624  0363          LET STRNG1 = "PSH-ANK12"
625  0364          GOTO AMSG
626  0365  S33:    LET STRNG0 = "AN TNK EX RD MON FAIL "
627  0366          LET STRNG1 = "RXS-AN-K1"
628  0367          GOTO AMSG
629  0368  S34:    LET STRNG0 = "AN ANN EXH RADN HIGH  "
630  0369          LET STRNG1 = "RSH-AN-K2"
631  0370          GOTO AMSG
632  0371  S35:    LET STRNG0 = "AN K2 STK MN SMPL PMP "
633  0372          LET STRNG1 = "PSH-ANK21"
634  0373          GOTO AMSG
635  0374  S36:    LET STRNG0 = "AN K2 REC SMPL PUMP   "
636  0375          LET STRNG1 = "PSH-ANK22"
637  0376          GOTO AMSG
638  0377  S37:    LET STRNG0 = "AN ANN EX RD MON FAIL "
639  0378          LET STRNG1 = "RXS-AN-K2"
640  0379          GOTO AMSG
641  0380  S38:    LET STRNG0 = "AN-271 BLD POWER LOSS "
642  0381          LET STRNG1 = "JXS-AN271"
643  0382          GOTO AMSG
644  0383  S39:    LET STRNG0 = "AN-271  RADN HI/FAIL  "
645  0384          LET STRNG1 = "RSHAN2711"
646  0385          GOTO AMSG
647  0386  S40:    LET STRNG0 = "AN-271 RADN HI/FAIL   "
648  0387          LET STRNG1 = "RSHAN2712"
649  0388          GOTO AMSG
650  0389  S41:    LET STRNG0 = "AN-241 RADN HI/FAIL   "
651  0390          LET STRNG1 = "RSH-AN-12"
652  0391          GOTO AMSG
653  0392  S42:    LET STRNG0 = "AN FARM GENERAL ALARM "
654  0393          LET STRNG1 = "QS-AN-271"
655  0394          GOTO AMSG
656  0395  S43:    LET STRNG0 = "AN HEAT TRACE OVERTMP "
657  0396          LET STRNG1 = "TXS-AN-2"
658  0397          GOTO AMSG
659  0398  S44:    LET STRNG0 = "AN AREA RADN BYPASS   "
660  0399          LET STRNG1 = "YS-AN-ARM"
```

```
661   0400            GOTO AMSG
662   0401   S45:     LET STRNG0 = "AN MASTER SHUTDOWN    "
663   0402            LET STRNG1 = "YS-AN-MSD"
664   0403            GOTO AMSG
665           ;
666           ;-mct1- Abandoned code for white alarms tied to pump devices:
667           ;
668           S46:     ;       BLET STRNG0 = "PUMP AN-101 SHUTDOWN  "
669           ;       BLET STRNG1 = "P-AN-101"
670           ;       GOTO AMSG
671           S47:     ;       BLET STRNG0 = "PUMP AN-102 SHUTDOWN  "
672           ;       BLET STRNG1 = "P-AN-102"
673           ;       GOTO AMSG
674           S48:     ;       BLET STRNG0 = "PUMP AN-103 SHUTDOWN  "
675           ;       BLET STRNG1 = "P-AN-103"
676           ;       GOTO AMSG
677           S49:     ;       BLET STRNG0 = "PUMP AN-104 SHUTDOWN  "
678           ;       BLET STRNG1 = "P-AN-104"
679           ;       GOTO AMSG
680           S50:     ;       BLET STRNG0 = "PUMP AN-105 SHUTDOWN  "
681           ;       BLET STRNG1 = "P-AN-105"
682           ;       GOTO AMSG
683           S51:     ;       BLET STRNG0 = "PUMP AN-106 SHUTDOWN  "
684           ;       BLET STRNG1 = "P-AN-106"
685           ;       GOTO AMSG
686           S52:     ;       BLET STRNG0 = "PUMP AN-107 SHUTDOWN  "
687           ;       BLET STRNG1 = "P-AN-107"
688           ;       GOTO AMSG
689           ;-mct1-
690           ;
691   0404    S53:     LET STRNG0 = "ANBATCH PROGRAM START "
692   0405            GOTO AMSG
693   0406    S54:     LET STRNG0 = "ANBATCH IN WRONG UNIT "
694   0407            GOTO AMSG
695   0408    S55:     LET STRNG0 = "ANBATCH TERMINATING   "
696   0409            GOTO AMSG
697           ;
698   0410    AMSG:    IF (FPASS)THEN
699                       BEGIN
700   0411                PRINT ADI,"%p ALARM ACKNOWLEDGE",STRNG1
701   0412                LET FPASS = 0
702   0413                GOTO NXT
703                       END
704   0415              IF PTYPE = 4 THEN LET COLOR0 = 3
705   0416                 ELSE LET COLOR0 = PTYPE
706   0417              LET EPNS0 = STRNG1
707   0418              IN0 = TIMSTR(SSNG0)
708   0420              IF IN0 != 8 THEN LET SSNG0 = "TIMEFAIL"
709   0421              IN0 = APPEND(LSG0,STRNG0,SSNG0)
710   0423              IF IN0 < 0 THEN LSG0 = "APPEND FAILED!!"
711   0424              IN0 = DATSTR(SDATE0)
712   0426              IF IN0 != 9 THEN SDATE0 = "DATE FAIL"
713   0428              IF PSKID = 1 THEN LET GRAFX0 = 272
714   0430              IF PSKID = 2 THEN LET GRAFX0 = 272
715   0432              IF PSKID = 3 THEN LET GRAFX0 = 273
```

```
716   0434            IF PSKID = 4 THEN LET GRAFX0 = 274
717   0436            IF PSKID = 5 THEN LET GRAFX0 = 270
718   0438            IF PSKID = 0 THEN LET GRAFX0 = 270
719   0440            IF PTYPE = 1 THEN PRINT ADI,"WHITE ALARM %p %p GRAPHICS %i",EPNS0,STRNG0,GRAFX0
720   0441   NXT:     IF FSTART = 1 THEN
721                      BEGIN
722   0442               LET COLOR0 = 0
723   0443               LET GRAFX0 = 0
724   0444               LET STRNG1 =" "
725   0445               LET FSTART = 0
726   0446               PRINT ADI,"%p",STRNG0
727                      END
728   0447            RETURN
729         ;
730   0448   ERRSUB:  IF ERRN != ERRNUM() THEN
731                     BEGIN
732   0449              LET ERRN = ERRNUM()
733   0450              LET ERRL = ERRLIN()
734   0451              PRINT ADI,"ERROR # %i AT LINE %i, ILOOP = %i",errn,errl,ILOOP
735                     END
736   0452            ERSCLR
737   0453            GOTO AGAIN
738         ;
739         ; TERM:  LET FABORT = 0
740   0454   TERM:    STRNG0 = "ANBATCH TERMINATED    "
741   0455            LET ILOOP = 55
742   0456            LET FSTART = 1
743   0457'           GOSUB CYCL
744                   END

      Total Errors Detected      0     Total Warnings      0
           Sequence Program Size = 8864 Bytes, 554 Pages
                112 Forward Reference(s)
```

APBATCH

```
1           ;       A P B A T C H . S E Q                                   VERSION 2
2           ;
3           ;       Function:       Sequence program for Lamp function,
4           ;                       message display on graphics, and
5           ;                       printer messages.
6           ;
7           ;       Written By      : R. N. Kersey
8           ;       Date            : 05-31-92
9           ;       Facility        : 242-A Evaporator Bldg.
10          ;       Machine         : VAX 4104-A
11          ;       Op. System      : VAX/VMS 5.5-2AH
12          ;       D/3 Version     : 6.3-2 BASILINE
13          ;       Compiler        : SIC
14          ;       Build command   : @[320,20]APBATCH
15          ;
16          ;       UNIT = 2, SKDKEY = 4, 5, 6, 7
17          ;
18          ;       Assoc. UPF:     [300,1]_TNKFRM.___      TNKFRM.LST
19          ;       Assoc. RPF:     [300,1]_FARM.___        FARM.LST
20          ;       SDB:            [300,1]PCM1ASDB.___     PCM1ASDB.CMD
21          ;
22          ;       POINTER   SDB VAR.      FUNCTION
23          ;       -------   ---------     ----------------------------------------
24          ;       PSTAT     INTn          Current EPN alarm state is an integer tied to
25          ;                               graphic flashing string of value:
26          ;       .         (n=0-100)             0 = previously not alarmed
27          ;       .                               1 = previously alarmed, not ack
28          ;       .                               2 = previously alarmed, is ack
29          ;       .                               3 = alarm is inhibited
30          ;       .                               4 = input is off scan
31          ;       .                               5 = input is bad
32          ;       PDIN      'EPN'         ---
33          ;       PTYPE     IALMn         alarm type 0 = none, 1 = white, 2 = yellow,
34          ;                                 3 = yellow and yellow, 4 = red,
35          ;                                 6 = yellow and red
36          ;       PREV      FXn           Reverse acting (not fail safe) flag (1=set)
37          ;       PSKID     SKIDn         Tracks which graphic has an alarm
38          ;       PPASS     FPASn         If set (1), turn lamp off when alarm is ACKnowledged
39          ;       PSN       FSNn          Flag for the graphic to ACKnowledge
40          ;       ---       FHOLD         Flag to update the lamps
41          ;       ---       FNING         Flag for DOG1A.C micro select
42          ;       ---       ACK           Flag for C program to ACK system wide
43          ;
44          ;   MODIFICATION LOG:
45          ;
46          ;       12-DEC-91       30 Second SKID update added.                R. Kersey
47          ;
48          ;-mct1+  14-MAY-93      Added FALMON for DEV pump shutdown.         M. Teats
49          ;
50          ;       11-JUN-93       Added comments and baselined for        M. Teats
51          ;                       D/3 version 3.3.  CCN# 93JUN01-01.
52          ;
53          ;       15-DEC-94       Revised header to indicate the new VMS      J. Berger
54          ;                       system. Rebaselined document. Removed
55          ;                       obsolite Select() statements.
```

```
56          ;
57          ;  END MOD LOG
58          ;
59          UNIT     APFARM
60          ;
61                   GROUT == FLSPIT
62                   AUTO == 'FQI-APSP1':CB_FLG:1              ; For FQI-RESET auto mode check
63                   CALOUT == 'FQI-APSP1':CB_OTVL:1           ; For FQI-RESET calc block outvalue
64          ;
65   0000            ON HOLD GOSUB HLD
66          ;
67   0001            GETUNIT TRIP                              ; Which sequence unit is program running in
68   0002            IF TRIP != 2 THEN                         ; Abort program if running in wrong unit
69                      BEGIN
70   0003               FSTART = 1
71   0004               ILOOP = 95
72   0005               GOSUB CYCL
73   0006               FSTART = 1
74   0007               PRINT ADI," >>>>>  ATTEMPT TO RUN APBATCH IN WRONG UNIT"
75   0008               GOTO TERM
76                      END
77          ;        IF !SELECT() THEN GOTO AGAIN
78          ;
79          ;        B E G I N   I N I T I A L I Z I N G :
80          ;
81   0009            NOP
82   0010            ERRN = 0
83   0011            ERRL = 0
84          ;
85   0012    INIT:   LET FABORT = 0                    ; If FABORT = 1 then abort program
86   0013            LET FSTART = 0                    ; Program start flag
87   0014            LET FLDALM = 0
88   0015            LET FSL7 = 0                      ; For FQI-RESET manual mode flag
89   0016            LAMP 0,7,0                        ; Turn off the lamps
90   0017            LAMP 0,6,0
91   0018            LAMP 0,5,0
92   0019            LAMP 0,4,0
93   0020            LAMP 0,63,0
94   0021            LAMP 0,22,9
95   0022            ASSIGN PDEV = APFARM                              ; First of 6 alarm ack devices
96   0023            FOR ILOOP = 1 TO 6
97   0024            PDEV:DEV_TMP0 = 1                                 ; Set all devices to ack state
98   0025            IF AND(PDEV:DEV_ALMCT,16) THEN                    ; If alarm device is inha, remove the inha
99   0026              PDEV:DEV_ALMCT=XOR(PDEV:DEV_ALMCT,16)
100  0027            IF AND(PDEV:DEV_ALMCT,32) THEN                    ; If alarm device is ack, clear it
101  0028               PDEV:DEV_ALMCT = XOR(PDEV:DEV_ALMCT,32)
102  0029            STEP PDEV BY 1
103  0030            NEXT ILOOP
104         ;
105  0031            FARM = 0                          ; LAMP 7        LEAK0
106  0032            SERPIT = 1                        ; not used
107  0033            VLVPIT = 2                        ; LAMP 5        LEAK2
108  0034            HVAC = 3                          ; LAMP 6        LEAK3
109  0035            BLD271 = 4                        ; LAMP 4        LEAK4
110  0036            GROUT = 5                         ; LAMP22        LEAK5
```

```
111             ;
112     0037            LEAK0 = 7                       ; FARM
113     0038            LEAK1 = 63                      ; not used
114     0039            LEAK2 = 5                       ; VALVE PIT
115     0040            LEAK3 = 6                       ; HVAC
116     0041            LEAK4 = 4                       ; AW-271
117     0042            LEAK5 = 22                      ; AP-102 GROUT
118             ;
119             ;       First, set all flags and integers:
120             ;
121     0043            ASSIGN PDIN = 'YS-AP-101'                       ; Assign pointer to first DIN
122     0044            ASSIGN PSTAT = INT0
123     0045            ASSIGN PTYPE = IALM0
124     0046            ASSIGN PREV = FX0
125     0047            ASSIGN PSKID = SKID0
126     0048            ASSIGN PPASS = FPAS0
127                     ;
128                     ;  Note that iloop 0 is string 1 (S1:), and so on, in Alarm Typer String section
129                     ;
130     0049            FOR ILOOP = 0 TO 82                             ; For DINs
131     0050            LET PPASS = 0
132     0051            LET PREV = 0
133     0052            LET PTYPE = 2
134     0053            LET PSTAT = 0
135     0054            IF AND(PDIN:DIN_DFLG,3) THEN                    ; Check for alarms (must be red or yellow)
136                        BEGIN
137     0055               LET PTYPE = 2                                ; Default to yellow alarm
138     0057               IF AND(PDIN:DIN_DFLG,1) THEN  LET PREV = 1           ; Reverse acting
139     0059               IF AND(PDIN:DIN_DFLG,4) THEN  LET PTYPE = 4          ; Set to red critical alarm
140                        END
141     0061            IF ILOOP < 29 THEN  LET PSKID = FARM
142     0063            IF ILOOP = 3 THEN  LET PSKID = GROUT
143     0065            IF (ILOOP > 28 & ILOOP < 38) THEN  LET PSKID = GROUT
144     0067            IF (ILOOP > 37 & ILOOP < 51) THEN  LET PSKID = BLD271
145     0069            IF (ILOOP > 50 & ILOOP < 54) THEN  LET PSKID = VLVPIT
146     0071            IF (ILOOP > 53 & ILOOP < 67) THEN  LET PSKID = HVAC
147     0073            IF (ILOOP > 66 & ILOOP < 70) THEN  LET PSKID = VLVPIT
148     0075            IF (ILOOP = 70 | ILOOP = 71) THEN  LET PSKID = VLVPIT
149     0077            IF (ILOOP = 72 | ILOOP = 73) THEN  LET PSKID = GROUT
150     0079            IF (ILOOP = 74 | ILOOP = 75) THEN  LET PSKID = BLD271
151     0081            IF (ILOOP > 75 ) THEN  LET PSKID = VLVPIT
152     0083            IF (ILOOP = 82 ) THEN  LET PSKID = GROUT
153     0085            IF AND(PDIN:DIN_ALMCT,32) THEN  PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
154     0086            STEP PSTAT BY 1
155     0087            STEP PDIN BY 1
156     0088            STEP PTYPE BY 1
157     0089            STEP PREV BY 1
158     0090            STEP PSKID BY 1
159     0091            STEP PPASS BY 1
160     0092            NEXT ILOOP
161             ;
162             ;       White alarms tied to pump devices:
163             ;
164     0093            FOR ILOOP = 83 TO 92
165     0094            LET PSTAT = 0
```

```
166   0095          LET PTYPE = 1
167   0096          LET PREV = 1
168   0097          LET PPASS = 1
169   0099          IF ILOOP < 91 THEN LET PSKID = FARM
170   0100            ELSE LET PSKID = GROUT
171   0102          IF ILOOP = 84 THEN LET PSKID = GROUT
172   0103          STEP PSTAT BY 1
173   0104          STEP PTYPE BY 1
174   0105          STEP PREV BY 1
175   0106          STEP PSKID BY 1
176   0107          STEP PPASS BY 1
177   0108          NEXT ILOOP
178                 ;
179                 ;  Set the white alarms:
180                 ;
181   0109          IALM0 = 1
182   0110          IALM3 = 1
183   0111          IALM6 = 1
184   0112          IALM11 = 1
185   0113          IALM12 = 1
186   0114          IALM15 = 1
187   0115          IALM20 = 1
188   0116          IALM23 = 1
189   0117          IALM26 = 1
190   0118          IALM30 = 1
191   0119          IALM31 = 1
192   0120          IALM32 = 1
193   0121          IALM34 = 1
194   0122          IALM35 = 1
195   0123          IALM36 = 1
196   0124          IALM37 = 1
197   0125          IALM43 = 1
198   0126          IALM48 = 1
199   0127          IALM51 = 1
200   0128          IALM52 = 1
201   0129          IALM55 = 1
202   0130          IALM58 = 1
203   0131          IALM59 = 1
204   0132          IALM62 = 1
205   0133          IALM65 = 1
206   0134          IALM66 = 1
207   0135          IALM77 = 1
208   0136          IALM78 = 1
209   0137          IALM79 = 1
210   0138          IALM80 = 1
211   0139          IALM81 = 1
212               ;
213   0140          FX38 = 1
214   0141          FX39 = 1
215               ;   FX52 = 1
216               ;   FX80 = 1
217               ;   FX81 = 1
218               ;
219   0142          FPAS1 = 0
220   0143          FPAS0 = 1
```

```
221   0144            FPAS3 = 1
222   0145            FPAS6 = 1
223   0146            FPAS12 = 1
224   0147            FPAS15 = 1
225   0148            FPAS20 = 1
226   0149            FPAS23 = 1
227   0150            FPAS26 = 1
228   0151            FPAS30 = 1
229   0152            FPAS34 = 1
230           ;
231   0153            FX0 = 1
232   0154            FX3 = 1
233   0155            FX6 = 1
234   0156            FX12 = 1
235   0157            FX15 = 1
236   0158            FX20 = 1
237   0159            FX23 = 1
238   0160            FX26 = 1
239   0161            FX30 = 1
240   0162            FX34 = 1
241           ;
242           ;       E N D   I N I T I A L I Z I N G
243           ;
244           ;       S E N D   P R O G R A M   S T A R T   M E S S A G E
245           ;
246   0163            ILOOP = 94
247   0164            FSTART = 1
248   0165            TIMERCLR STMR2
249   0166            TIMERON STMR2
250   0167            GOSUB CYCL
251   0168            ON ERROR GOSUB ERRSUB
252           ;
253           ;       B E G I N   P R O G R A M   L O O P :
254           ;
255   0169    AGAIN:  MARK STEP "AGAIN",0,0
256           ;       IF !SELECT() THEN FNING = 0
257           ;       ELSE FNING = 1                          ; Flag 480 determines if selected micro for DOG1A.C
258           ;       WAITUNTIL SELECT() | FABORT             ; Wait until micro selected or program aborted
259           ;       IF (FABORT) THEN GOTO TERM              ; Terminate program if abort flag set
260           ;       IF FNING = 0 THEN FHOLD = 1
261   0171            IF STMR2 > 30 THEN FHOLD = 1
262   0172            WAIT 1
263           ;
264   0173            DEV0 = 0
265   0174            DEV1 = 0
266   0175            DEV2 = 0
267   0176            DEV3 = 0
268   0177            DEV4 = 0
269   0178            DEV5 = 0
270           ;
271           ;       If alarm device is alarmed and its ALMCT indicates alarms are ack'd:
272           ;
273           ;               DEVICE          FLAG    OFFSET
274           ;               ---------       ------  ------
275           ;               APFARM          FLEAK0  447
```

```
276          ;               AP-SERPIT       FLEAK1  448
277          ;               AP-VLVPIT       FLEAK2  449
278          ;               AP-HVAC         FLEAK3  450
279          ;               AP-271          FLEAK4  451
280          ;               GROUT           FLEAK5  452
281          ;
282  0179            ASSIGN PDEV = APFARM
283  0180            ASSIGN PSN = FSN0
284  0181            ASSIGN PACK = FLEAK0
285  0182            FOR ILOOP = 0 TO 5
286  0183            LET PSN = 0
287  0184            IF (PDEV:DEV_TMP0 = 2) & AND(PDEV:DEV_ALMCT,32) THEN
288                     BEGIN
289  0185                PDEV:DEV_TMP0 = 1                    ; Put device in ack state
290  0186                LET PSN = 1                          ; Set flag for the graphic to ack
291  0187                LET FLGACK = 1                       ; WATCH.C spawns ACKFLG.C which resets flag
292  0188                LET ACK = 1                          ; Set flag for C program to ACK system wide
293  0189                LET PACK = 1
294  0190                FHOLD = 1
295  0191                ASSIGN PMSG0 = LEAK0
296  0192                STEP PMSG0 BY ILOOP
297  0194                IF ILOOP != 1 THEN LAMP 0,PMSG0,1    ; Flash lamp
298                      END
299  0195            STEP PDEV BY 1
300  0196            STEP PSN BY 1
301  0197            STEP PACK BY 1
302  0198            NEXT ILOOP
303          ;
304  0199            LEAK0 = 7
305  0200            LEAK1 = 63
306  0201            LEAK2 = 5
307  0202            LEAK3 = 6
308  0203            LEAK4 = 4
309  0204            LEAK5 = 22
310          ;
311.         ;       Field Alarm Silence feature:
312          ;
313  0205            IF ACK THEN
314                     BEGIN
315  0206               TIMERCLR STMR1
316  0207               TIMERON STMR1
317  0208               FLDALM = 1
318                     END
319  0210            IF FLDALM THEN IF STMR1 >= 30 THEN       ;If field alarm then ack in 30 sec
320                     BEGIN
321  0211               TIMEROFF STMR1
322  0212               TIMERCLR STMR1
323  0213               FLDALM = 0
324  0214               OPEN 'HS-AP-SIL'                      ; Silence alarm from PCM
325                     END
326                 ;
327                 ;  If field alarm ack then set flag and start timer:
328                 ;
329  0216            IF RESET THEN IF LMPVAR < 2 THEN
330                     BEGIN
```

```
331  0217              FLDALM = 1
332  0218              TIMERON STMR1
333  0219              RESET = 0
334                    END
335  0221            IF LMPVAR = 2 THEN RESET = 1
336          ;
337          ;       BEGIN DINs:
338          ;
339  0222            ASSIGN PDIN = 'YS-AP-101'                ; First DIN in APFARM
340  0223            ASSIGN PTYPE = IALMO
341  0224            ASSIGN PREV = FXO
342  0225            ASSIGN PSTAT = INTO
343  0226            ASSIGN PSKID = SKIDO
344  0227            ASSIGN PPASS = FPASO
345  0228            CURENT = 0                               ; Used to set lamp status
346  0229            NXT = 0                                  ; Temporary INT
347  0230            BITVAL = 0
348          ;
349  0231    LOP:    FOR ILOOP = 1 TO 83
350  0232            IF PTYPE  THEN                                    ; If some alarm
351                     BEGIN                                  ;be1
352  0234               IF AND(PDIN:DIN_ALMCT,16) THEN LET PSTAT = 3   ; If EPN is inha
353  0236               IF AND(PDIN:DIN_ALMCT,4) THEN LET PSTAT = 5    ; If EPN is ibad
354  0238               IF !AND(PDIN:DIN_FLG,8) THEN LET PSTAT = 4     ; If EPN is dis
355  0241               IF ! PSTAT THEN IF PREV = PDIN:DIN_VAL THEN GOSUB ALM  ; If not prev alarmed
356  0242               IF PSTAT THEN
357                        BEGIN                               ;be2
358  0243                  IF PSTAT < 3 THEN                          ; If not inha
359                           BEGIN                            ;be3
360  0244                     IF PSTAT = 1 THEN
361                              BEGIN                         ;be4
362  0245                        IF AND(PDIN:DIN_ALMCT,32) THEN      ; If alarm is ack
363                                 BEGIN                      ;be5
364  0246                           LET PSTAT = 2
365  0247                           FHOLD = 1
366                                 END                        ;en5
367  0248                        IF ACK THEN                         ; If graphic has been ack
368                                 BEGIN                      ;be6  -  go find which graphic
369  0249                           CRIT
370  0250                           ASSIGN PSN = FSNO                ; Routine to find graphic
371  0251                           STEP PSN BY PSKID
372  0252                           IF PSN THEN                      ; If this is the graphic
373                                    BEGIN                   ;be7
374  0253                              LET PSTAT = 2
375  0255                              IF PTYPE = 1 THEN PDIN:DIN_ALMCT = OR(PDIN:DIN_ALMCT,32)
376  0256                              FPASS = 1
377  0257                              GOSUB ACKK                ; Print signal was ack
378                                    END                     ;en7
379                                 END                        ;en6
380                              END                           ;en4
381  0259                        IF PREV != PDIN:DIN_VAL THEN GOSUB CLR   ; If alarm has cleared
382  0260                        ASSIGN PMSGO = DEVO                      ; Find active alarms & graphic:
383  0261                        STEP PMSGO BY PSKID
384  0262                        IF PPASS THEN
385                                 BEGIN
```

```
386   0264                          IF PSTAT = 1 THEN PMSG0 = OR(PMSG0,PSTAT)
387                                 END
388   0265                        ELSE PMSG0 = OR(PMSG0,PSTAT)
389                             END                                          ;en3
390   0266                    ELSE IF !AND(PDIN:DIN_ALMCT,16)&!AND(PDIN:DIN_ALMCT,4)&AND(PDIN:DIN_FLG,8)THEN
391                               BEGIN
392   0267                        PSTAT = 0
393   0268                        FHOLD = 1
394                               END
395                           END                                            ;en2
396                     END                                                  ;en1
397   0270              ELSE IF PSTAT THEN PSTAT = 0
398   0271              STEP PDIN BY 1
399   0272              STEP PTYPE BY 1
400   0273              STEP PREV BY 1
401   0274              STEP PSTAT BY 1
402   0275              STEP PSKID BY 1
403   0276              STEP PPASS BY 1
404   0277              NEXT ILOOP
405             ;
406             ;         END DINs
407             ;
408   0278              DEV6 = 0
409   0279              DEV7 = 0
410   0280              DEV8 = 0
411   0281              DEV9 = 0
412   0282              DEV10 = 0
413   0283              DEV11 = 0
414             ;
415             ;         BEGIN WHITE ALARMS FOR PUMP DEVICES:
416             ;
417   0284      AUX:    ASSIGN PEPN = 'P-AP-101'
418   0285              FOR ILOOP = 84 TO 93
419   0286              IF PTYPE THEN
420                        BEGIN
421   0288                 IF AND(PEPN:DEV_ALMCT,16) THEN LET PSTAT = 3   ; If inha
422   0290                 IF AND(PEPN:DEV_ALMCT,4) THEN LET PSTAT = 5            ; If ibad
423   0292                 IF !AND(PEPN:DEV_FLG,8) THEN LET PSTAT = 4             ; If dis
424   0293                 IF PSTAT < 3 THEN                                      ; If not inha
425                           BEGIN
426   0294                    IF (PSTAT = 1) & AND(PEPN:DEV_ALMCT,32) THEN        ; If prev unack alarm, and is now ack
427                                 BEGIN
428   0295                          LET PSTAT = 2
429   0296                          FHOLD = 1
430                                 END
431   0297                    IF ACK THEN
432                                 BEGIN
433   0298                          ASSIGN PSN = FSN0
434   0299                          STEP PSN BY PSKID
435   0300                          IF PSN & PSTAT = 1 THEN
436                                    BEGIN
437   0301                             LET PSTAT = 2
438   0303                             IF PTYPE = 1 THEN PEPN:DEV_ALMCT = OR(PEPN:DEV_ALMCT,32)
439   0304                             FPASS = 1
440   0305                             GOSUB ACKK
```

```
441                                  END
442                               END
443   0306                     IF !PSTAT THEN                                    ; If prev not alarmed
444                               BEGIN
445   0307                        IF AND(PEPN:DEV_STAT,16) THEN                   ; If Shutdown
446                                  BEGIN
447   0308                           LET PSTAT = 1
448   0309                           FALMON = 1                                   ; -mct1+
449   0311                           IF AND(PEPN:DEV_ALMCT,32)THEN PEPN:DEV_ALMCT = XOR(PEPN:DEV_ALMCT,32)
450   0312                           ASSIGN PDEV = APFARM
451   0313                           STEP PDEV BY PSKID
452   0314                           PDEV:DEV_TMP0 = 2
453   0315                           GOSUB CYCL
454                                  END
455                               END
456   0316                     ELSE IF !AND(PEPN:DEV_STAT,16) THEN
457                                  BEGIN
458   0317                           LET PSTAT = 0
459   0319                           IF AND(PEPN:DEV_ALMCT,32)THEN PEPN:DEV_ALMCT = XOR(PEPN:DEV_ALMCT,32)
460   0320                           FHOLD = 1
461                                  END
462   0321                     IF PSTAT THEN
463                               BEGIN
464   0322                        ASSIGN PMSG0 = DEV6
465   0323                        STEP PMSG0 BY PSKID
466   0324                        IF PPASS THEN
467                                  BEGIN
468   0326                           IF PSTAT = 1 THEN PMSG0 = OR(PMSG0,PSTAT)
469                                  END
470   0327                        ELSE PMSG0 = OR(PMSG0,PSTAT)
471                               END
472                            END
473   0328                  ELSE IF !AND(PEPN:DEV_ALMCT,16)&!AND(PEPN:DEV_ALMCT,4)&AND(PEPN:DEV_FLG,8)THEN
474                            BEGIN
475   0329                     PSTAT = 0
476   0330                     FHOLD = 1
477                            END
478                         END
479   0331              STEP PEPN BY 1
480   0332              STEP PTYPE BY 1
481   0333              STEP PREV BY 1
482   0334              STEP PSTAT BY 1
483   0335              STEP PSKID BY 1
484   0336              STEP PPASS BY 1
485   0337              NEXT ILOOP
486             ;
487             ;         END WHITE ALARMS FOR PUMP DEVICES
488             ;
489   0338      S2ND:   ACK = 0
490   0339              SORT = 0
491             ;
492   0340              DEV12 = OR(DEV0,DEV6)
493   0341              DEV13 = OR(DEV1,DEV7)
494   0342              DEV14 = OR(DEV2,DEV8)
495   0343              DEV15 = OR(DEV3,DEV9)
```

```
496   0344          DEV16 = OR(DEV4,DEV10)
497   0345          DEV17 = OR(DEV5,DEV11)
498         ;
499   0346          ASSIGN PMSG0 = DEV12
500   0347          ASSIGN PMSG1 = FARM
501   0348          FOR LLOOP = 1 TO 6
502   0349          IF PMSG0 THEN
503                   BEGIN
504   0351            IF AND(PMSG0,1) THEN PMSG1 = 2
505   0352            ELSE PMSG1 = 1
506                   END
507   0353          ELSE PMSG1 = 0
508   0354          STEP PMSG0 BY 1
509   0355          STEP PMSG1 BY 1
510   0356          NEXT LLOOP
511         ;
512   0357          MARK STEP "BOTTOM",0,0
513   0358          NOCRIT
514         ;
515         ;       B E G I N   F Q I - R E S E T :
516         ;
517         ;       Function:       For totalizers that have a CALC block which performs the
518         ;                       integration, set the output of the CALC block to zero
519         ;                       after the FQI loop has been placed in AUTO from MANUAL.
520         ;
521   0360          IF !AND(AUTO,64) THEN FSL7 = 1                  ; If manual, set manual flag
522   0361          IF AND(AUTO,64) & AND(FSL7,1) THEN              ; For manual-auto transition
523                   BEGIN
524   0362            FSL7 = 0
525   0363            CALOUT = 0.0
526                   END
527         ;
528         ;       E N D   F Q I - R E S E T
529         ;
530   0364          IF FHOLD THEN
531                   BEGIN
532   0365            LAMP 0,7,FARM
533   0366            LAMP 0,6,HVAC
534   0367            LAMP 0,5,VLVPIT
535   0368            LAMP 0,4,BLD271
536   0369            LAMP 0,22,GROUT
537   0370            FHOLD = 0
538   0371            TIMERCLR STMR2
539   0372            TIMERON STMR2
540                   END
541         ;
542   0373          GOTO AGAIN
543         ;
544         ;       E N D   P R O G R A M   L O O P
545         ;
546         ;       BEGIN CLEAR, ALARM, CYCLE, ACKNOWLEDGE, MESSAGE, ERROR, AND TERMINATE SUBROUTINES:
547         ;
548   0374  CLR:    LET PSTAT = 0
549   0376          IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
550   0377          FHOLD = 1
```

```
551  0378            RETURN
552                  ;
553  0379    ALM:    LET PSTAT = 1           ; Print message, flash lamp, horn on, scroll message at bottom of graphic
554  0380            CRIT                                          ; Critical
555  0382            IF AND(PTYPE,3) THEN FALMON = 1               ; Audible alarm on
556  0383            IF AND(PDIN:DIN_ALMCT,32) THEN                ; Clear the scan block
557  0384               PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
558  0385            ASSIGN PMSG0 = LEAK0
559  0386            STEP PMSG0 BY PSKID
560  0387            LAMP 0,PMSG0,2                                ; Lamp flash
561  0388            ASSIGN PDEV = APFARM                          ; Find dev and turn it on
562  0389            STEP PDEV BY PSKID
563  0390            PDEV:DEV_TMP0 = 2
564                  ;
565                  ;       Scrolling alarm summary region in graphic (Message, time, date, alarm type):
566                  ;
567  0391    CYCL:   LET LSG4 = LSG3                          ; LSG = message
568  0392            LET LSG3 = LSG2
569  0393            LET LSG2 = LSG1
570  0394            LET LSG1 = LSG0
571  0395            LET COLOR4 = COLOR3                           ; COLOR = type of alarm
572  0396            LET COLOR3 = COLOR2
573  0397            LET COLOR2 = COLOR1
574  0398            LET COLOR1 = COLOR0
575  0399            LET EPNS4 = EPNS3                             ; EPN
576  0400            LET EPNS3 = EPNS2
577  0401            LET EPNS2 = EPNS1
578  0402            LET EPNS1 = STRNG1
579  0403            LET GRAFX4 = GRAFX3                           ; GRAFX = graphic number
580  0404            LET GRAFX3 = GRAFX2
581  0405            LET GRAFX2 = GRAFX1
582  0406            LET GRAFX1 = GRAFX0
583  0407            LET SDATE4 = SDATE3                           ; SDATE = time and date
584  0408            LET SDATE3 = SDATE2
585  0409            LET SDATE2 = SDATE1
586  0410            LET SDATE1 = SDATE0
587                  ;
588                  ;       ALARM TYPER STRINGS:
589                  ;
590  0411    ACKK:   FHOLD = 1
591  0412            ON ILOOP GOTO S1,S2,S3,S4,S5,S6,S7,S8,S9,S10
592  0413            ON ILOOP - 10 GOTO S11,S12,S13,S14,S15,S16,S17,S18,S19,S20
593  0414            ON ILOOP - 20 GOTO S21,S22,S23,S24,S25,S26,S27,S28,S29,S30
594  0415            ON ILOOP - 30 GOTO S31,S32,S33,S34,S35,S36,S37,S38,S39,S40
595  0416            ON ILOOP - 40 GOTO S41,S42,S43,S44,S45,S46,S47,S48,S49,S50
596  0417            ON ILOOP - 50 GOTO S51,S52,S53,S54,S55,S56,S57,S58,S59,S60
597  0418            ON ILOOP - 60 GOTO S61,S62,S63,S64,S65,S66,S67,S68,S69,S70
598  0419            ON ILOOP - 70 GOTO S71,S72,S73,S74,S75,S76,S77,S78,S79,S80
599  0420            ON ILOOP - 80 GOTO S81,S82,S83,S84,S85,S86,S87,S88,S89,S90
600  0421            ON ILOOP - 90 GOTO S91,S92,S93,S94,S95,S96
601  0422            GOTO AMSG
602  0423    S1:     STRNG0 = "AP-101 PUMP MCC OFF   "
603  0424            STRNG1 ="YS-AP-101"
604  0425            GOTO AMSG
605  0426    S2:    STRNG0 = "AP-101 ANN EX RAD HI  "
```

```
606   0427            STRNG1 ="RSH-101AP"
607   0428            GOTO AMSG
608   0429   S3:      STRNG0 = "AP-101 ANN MON FAIL   "
609   0430            STRNG1 ="RXS-101AP"
610   0431            GOTO AMSG
611   0432   S4:      STRNG0 = "AP-102 PUMP MCC OFF   "
612   0433            STRNG1 ="YS-AP-102"
613   0434            GOTO AMSG
614   0435   S5:      STRNG0 = "AP-102 ANN EX RAD HI  "
615   0436            STRNG1 ="RSH-102AP"
616   0437            GOTO AMSG
617   0438   S6:      STRNG0 = "AP-102 RAD MON FAIL   "
618   0439            STRNG1 ="RXS-102AP"
619   0440            GOTO AMSG
620   0441   S7:      STRNG0 = "AP-103 PUMP MCC OFF   "
621   0442            STRNG1 ="YS-AP103"
622   0443            GOTO AMSG
623   0444   S8:      STRNG0 = "AP-103 ANN EX RAD HI  "
624   0445            STRNG1 ="RSH-103AP"
625   0446            GOTO AMSG
626   0447   S9:      STRNG0 = "AP-103 RAD MON FAIL   "
627   0448            STRNG1 ="RXS-103AP"
628   0449            GOTO AMSG
629   0450   S10:     STRNG0 = "AP-03C RAD MON FAIL   "
630   0451            STRNG1 ="RSH-AP03C"
631   0452            GOTO AMSG
632   0453   S11:     STRNG0 = "AP-03C LK PIT RAD HI  "
633   0454            STRNG1 ="RXS-AP03C"
634   0455            GOTO AMSG
635   0456   S12:     STRNG0 = "AP-03D DRAN PIT LEAK  "
636   0457            STRNG1 ="LDS-AP03D"
637   0458            GOTO AMSG
638   0459   S13:     STRNG0 = "AP-104 PUMP MCC OFF   "
639   0460            STRNG1 ="YS-AP-104"
640   0461            GOTO AMSG
641   0462   S14:     STRNG0 = "AP-104 ANN EX RAD HI  "
642   0463            STRNG1 ="RSH-104AP"
643   0464            GOTO AMSG
644   0465   S15:     STRNG0 = "AP-104 RAD MON FAIL   "
645   0466            STRNG1 ="RXS-104AP"
646   0467            GOTO AMSG
647   0468   S16:     STRNG0 = "PUMP AP-105 MCC OFF   "
648   0469            STRNG1 ="YS-AP-105"
649   0470            GOTO AMSG
650   0471   S17:     STRNG0 = "AP-105 ANN EX RAD HI  "
651   0472            STRNG1 ="RSH-105AP"
652   0473            GOTO AMSG
653   0474   S18:     STRNG0 = "AP-105 RAD MON FAIL   "
654   0475            STRNG1 ="RXS-105AP"
655   0476            GOTO AMSG
656   0477   S19:     STRNG0 = "AP-05C LK-PIT RAD HI  "
657   0478            STRNG1 ="RSH-AP05C"
658   0479            GOTO AMSG
659   0480   S20:     STRNG0 = "AP-05C RAD MON FAIL   "
660   0481            STRNG1 ="RXS-AP05C"
```

```
661   0482          GOTO AMSG
662   0483   S21:   STRNG0 = "PUMP AP-106 MCC OFF   "
663   0484          STRNG1 ="YS-AP-106"
664   0485          GOTO AMSG
665   0486   S22:   STRNG0 = "AP-106 ANN EX RAD HI  "
666   0487          STRNG1 ="RSH-106AP"
667   0488          GOTO AMSG
668   0489   S23:   STRNG0 = "106 EXH RAD MON FAIL  "
669   0490          STRNG1 ="RXS-106AP"
670   0491          GOTO AMSG
671   0492   S24:   STRNG0 = "PUMP AP-107 MCC OFF   "
672   0493          STRNG1 ="YS-AP-107"
673   0494          GOTO AMSG
674   0495   S25:   STRNG0 = "AP-107 ANN EX RAD HI  "
675   0496          STRNG1 ="RSH-107AP"
676   0497          GOTO AMSG
677   0498   S26:   STRNG0 = "107 EXH RAD MON FAIL  "
678   0499          STRNG1 ="RXS-107AP"
679   0500          GOTO AMSG
680   0501   S27:   STRNG0 = "PUMP AP-108 MCC OFF   "
681   0502          STRNG1 ="YS-AP-108"
682   0503          GOTO AMSG
683   0504   S28:   STRNG0 = "AP-108 ANN EX RAD HI  "
684   0505          STRNG1 ="RSH-108AP"
685   0506          GOTO AMSG
686   0507   S29:   STRNG0 = "108 EXH RAD MON FAIL  "
687   0508          STRNG1 ="RXS-108AP"
688   0509          GOTO AMSG
689   0510   S30:   STRNG0 = "AP-02D PUMP PIT LEAK  "
690   0511          STRNG1 ="LDSAP02DC"
691   0512          GOTO AMSG
692   0513   S31:   STRNG0 = "AP-102-3 MCC OFF      "
693   0514          STRNG1 ="YS-AP102M"
694   0515          GOTO AMSG
695   0516   S32:   STRNG0 = "AP-102-3 DIS PRES HI  "
696   0517          STRNG1 ="PSHAP102M"
697   0518          GOTO AMSG
698   0519   S33:   STRNG0 = "AP-102-3 DIS PRES LO  "
699   0520          STRNG1 ="PSLAP102M"
700   0521          GOTO AMSG
701   0522   S34:   STRNG0 = "AP-102-3 LEAK         "
702   0523          STRNG1 ="LDSAP102M"
703   0524          GOTO AMSG
704   0525   S35:   STRNG0 = "AP-102-3 ROTATOR OFF  "
705   0526          STRNG1 ="YS-AP102R"
706   0527          GOTO AMSG
707   0528   S36:   STRNG0 = "AP-102-3 STRAIN HIGH  "
708   0529          STRNG1 ="WSHAP102M"
709   0530          GOTO AMSG
710   0531   S37:   STRNG0 = "AP-102-3 TEMP HIGH    "
711   0532          STRNG1 ="TSHAP102M"
712   0533          GOTO AMSG
713   0534   S38:   STRNG0 = "AP-102-3 VIBRATIN HI  "
714   0535          STRNG1 ="VSHAP102M"
715   0536          GOTO AMSG
```

```
716   0537   S39:    STRNG0 = "AP-MASTER SHUTDOWN    "
717   0538           STRNG1 ="YS-AP-MSD"
718   0539           GOTO AMSG
719   0540   S40:    STRNG0 = "AP-AREA RAD BYPASS    "
720   0541           STRNG1 ="YS-AP-ARM"
721   0542           GOTO AMSG
722   0543   S41:    STRNG0 = "AP-FARM RADIATION HI  "
723   0544           STRNG1 ="RSH-AP-2"
724   0545           GOTO AMSG
725   0546   S42:    STRNG0 = "AP-FARM RAD MON FAIL  "
726   0547           STRNG1 ="RXS-AP-2"
727   0548           GOTO AMSG
728   0549   S43:    STRNG0 = "AP GENERAL ALARM       "
729   0550           STRNG1 ="QS-AP-1"
730   0551           GOTO AMSG
731   0552   S44:    STRNG0 = "AP HEAT TRACE OVRTMP   "
732   0553           STRNG1 ="TXS-AP-2"
733   0554           GOTO AMSG
734   0555   S45:    STRNG0 = "CENTRAL PMP PIT LEAK   "
735   0556           STRNG1 ="LDS-AP-A"
736   0557           GOTO AMSG
737   0558   S46:    STRNG0 = "ANNULUS PMP PIT LEAK   "
738   0559           STRNG1 ="LDS-AP-B"
739   0560           GOTO AMSG
740   0561   S47:    STRNG0 = "LEAK-DET PMPIT LEAK    "
741   0562           STRNG1 ="LDS-AP-C"
742   0563           GOTO AMSG
743   0564   S48:    STRNG0 = "271 ANNUNC POWER OFF   "
744   0565           STRNG1 ="JXS-AP-1"
745   0566           GOTO AMSG
746   0567   S49:    STRNG0 = "271 BLDG CAM FLOW LO   "
747   0568           STRNG1 ="FSLAP2711"
748   0569           GOTO AMSG
749   0570   S50:    STRNG0 = "271 BLDG RADN HIGH     "
750   0571           STRNG1 ="RSH-AP271"
751   0572           GOTO AMSG
752   0573   S51:    STRNG0 = "271 BLD RAD MON FAIL   "
753   0574           STRNG1 ="RXS-AP271"
754   0575           GOTO AMSG
755   0576   S52:    STRNG0 = "FLSH PIT FL LINE PRS   "
756   0577           STRNG1 ="PSH-APFP1"
757   0578           GOTO AMSG
758   0579   S53:    STRNG0 = "FLSH PIT V-FP-1 OPEN   "
759   0580           STRNG1 ="ZS-AP-FP1"
760   0581           GOTO AMSG
761   0582   S54:    STRNG0 = "AP-FLUSH PIT LEAK      "
762   0583           STRNG1 ="LDS-AP-FP"
763   0584           GOTO AMSG
764   0585   S55:    STRNG0 = "K1 EXHAUST FAN OFF     "
765   0586           STRNG1 ="YS-AP-K1"
766   0587           GOTO AMSG
767   0588   S56:    STRNG0 = "K1 EXH SMPLE FLOW LO   "
768   0589           STRNG1 ="FSL-APK13"
769   0590           GOTO AMSG
770   0591   S57:    STRNG0 = "K1 EXHAUST RAD HIGH    "
```

```
771  0592          STRNG1 ="RSH-APK11"
772  0593          GOTO AMSG
773  0594  S58:    STRNG0 = "K1 EXH RAD MON FAIL   "
774  0595          STRNG1 ="RXS-APK11"
775  0596          GOTO AMSG
776  0597  S59:    STRNG0 = "K1 EX REC SMPL LOFLO  "
777  0598          STRNG1 ="FSL-APK12"
778  0599          GOTO AMSG
779  0600  S60:    STRNG0 = "K1 EXH STACK FLOW LO  "
780  0601          STRNG1 ="FSL-APK11"
781  0602          GOTO AMSG
782  0603  S61:    STRNG0 = "AP SEAL-POT PIT LEAK  "
783  0604          STRNG1 ="LDS-AP-K1"
784  0605          GOTO AMSG
785  0606  S62:    STRNG0 = "K2 EXH FAN SHUTDOWN   "
786  0607          STRNG1 ="YS-AP-K2"
787  0608          GOTO AMSG
788  0609  S63:    STRNG0 = "K2 EXH MON SMPL FLOLO "
789  0610          STRNG1 ="FSL-APK23"
790  0611          GOTO AMSG
791  0612  S64:    STRNG0 = "K2 EXH RADIATION HI   "
792  0613          STRNG1 ="RSH-APK21"
793  0614          GOTO AMSG
794  0615  S65:    STRNG0 = "K2 EXH RADN MON FAIL  "
795  0616          STRNG1 ="RXS-APK21"
796  0617          GOTO AMSG
797  0618  S66:    STRNG0 = "K2 EXH REC SMPL FLOLO "
798  0619          STRNG1 ="FSL-APK22"
799  0620          GOTO AMSG
800  0621  S67:    STRNG0 = "K2 EXH STACK FLOW LO  "
801  0622          STRNG1 ="FSL-APK21"
802  0623          GOTO AMSG
803  0624  S68:    STRNG0 = "AP SERVICE PIT LEAK   "
804  0625          STRNG1 ="LDS-AP-SP"
805  0626          GOTO AMSG
806  0627  S69:    STRNG0 = "AP SER PIT RAD HI     "
807  0628          STRNG1 ="RSH-AP-SP"
808  0629          GOTO AMSG
809  0630  S70:    STRNG0 = "AP SER PIT RAD FAIL   "
810  0631          STRNG1 ="RXS-AP-SP"
811  0632          GOTO AMSG
812  0633  S71:    STRNG0 = "AP SL-509/510 LEAK    "
813  0634          STRNG1 ="LDS-AP-SL"
814  0635          GOTO AMSG
815  0636  S72:    STRNG0 = "AP SN-609/610 LEAK    "
816  0637          STRNG1 ="LDS-AP-SN"
817  0638          GOTO AMSG
818  0639  S73:    STRNG0 = "AP SN-621 LINE LEAK   "
819  0640          STRNG1 ="LDSAP621A"
820  0641          GOTO AMSG
821  0642  S74:    STRNG0 = "AP SN-650 LINE LEAK   "
822  0643          STRNG1 ="LDSAP6502"
823  0644          GOTO AMSG
824  0645  S75:    STRNG0 = "AP TANKS EXCESS VACU  "
825  0646          STRNG1 ="PSH-AP-2"
```

```
826   0647          GOTO AMSG
827   0648   S76:   STRNG0 = "AP TANKS VACUUM LOSS  "
828   0649          STRNG1 ="PSL-AP-2"
829   0650          GOTO AMSG
830   0651   S77:   STRNG0 = "AP VALVE PIT LEAK     "
831   0652          STRNG1 ="LDS-AP-VP"
832   0653          GOTO AMSG
833   0654   S78:   STRNG0 = "DR-712 FLOW DETECTED  "
834   0655          STRNG1 ="FSH-DR712"
835   0656          GOTO AMSG
836   0657   S79:   STRNG0 = "DR-712 FLOW DET FAIL  "
837   0658          STRNG1 ="FXS-DR712"
838   0659          GOTO AMSG
839   0660   S80:   STRNG0 = "FLUSH LINE PRESS HIGH "
840   0661          STRNG1 ="PSH-APFL1"
841   0662          GOTO AMSG
842   0663   S81:   STRNG0 = "AP VPIT V-DR-1 OPEN   "
843   0664          STRNG1 ="ZS-AP-DR1"
844   0665          GOTO AMSG
845   0666   S82:   STRNG0 = "AP VPIT V-FL-1 OPEN   "
846   0667          STRNG1 ="ZS-AP-FL1"
847   0668          GOTO AMSG
848   0669   S83:   STRNG0 = "ROTOR 102-3 BYPASS ON "
849   0670          STRNG1 ="YS-1023BY"
850   0671          GOTO AMSG
851   0672   S84:   STRNG0 = "PUMP AP-101 SHUTDOWN  "
852   0673          STRNG1 = "P-AP-101"
853   0674          GOTO AMSG
854   0675   S85:   STRNG0 = "PUMP AP-102 SHUTDOWN  "
855   0676          STRNG1 = "P-AP-102"
856   0677          GOTO AMSG
857   0678   S86:   STRNG0 = "PUMP AP-103 SHUTDOWN  "
858   0679          STRNG1 = "P-AP-103"
859   0680          GOTO AMSG
860   0681   S87:   STRNG0 = "PUMP AP-104 SHUTDOWN  "
861   0682          STRNG1 = "P-AP-104"
862   0683          GOTO AMSG
863   0684   S88:   STRNG0 = "PUMP AP-105 SHUTDOWN  "
864   0685          STRNG1 = "P-AP-105"
865   0686          GOTO AMSG
866   0687   S89:   STRNG0 = "PUMP AP-106 SHUTDOWN  "
867   0688          STRNG1 = "P-AP-106"
868   0689          GOTO AMSG
869   0690   S90:   STRNG0 = "PUMP AP-107 SHUTDOWN  "
870   0691          STRNG1 = "P-AP-107"
871   0692          GOTO AMSG
872   0693   S91:   STRNG0 = "PUMP AP-108 SHUTDOWN  "
873   0694          STRNG1 = "P-AP-108"
874   0695          GOTO AMSG
875   0696   S92:   STRNG0 = "PUMP AP-102M SHUTDOWN "
876   0697          STRNG1 = "P-AP-102M"
877   0698          GOTO AMSG
878   0699   S93:   STRNG0 = "ROTR AP-102R SHUTDOWN "
879   0700          STRNG1 = "P-AP-102R"
880   0701          GOTO AMSG
```

```
881   0702   S94:    STRNG0 = "APBATCH PROGRAM START "
882   0703           GOTO AMSG
883   0704   S95:    STRNG0 = "APBATCH IN WRONG UNIT "
884   0705           GOTO AMSG
885   0706   S96:    STRNG0 = "APBATCH TERMINATING   "
886   0707           GOTO AMSG.
887          ;
888   0708   AMSG:   IF (FPASS)THEN
889                     BEGIN
890   0709              PRINT ADI,"%p ALARM ACKNOWLEDGE",STRNG1
891   0710              FPASS = 0
892   0711              GOTO NXT
893                     END
894   0713           IF PTYPE = 4 THEN COLOR0 = 3
895   0714              ELSE COLOR0 = PTYPE
896   0715           EPNS0 = STRNG1
897   0716           IN0 = TIMSTR(SSNG0)
898   0718           IF IN0 != 8 THEN SSNG0 = "TIMEFAIL"
899   0719           IN0 = APPEND(LSG0,STRNG0,SSNG0)
900   0721           IF IN0 < 0 THEN LSG0 = "APPEND FAILED!!"
901   0722           IN0 = DATSTR(SDATE0)
902   0724           IF IN0 != 9 THEN SDATE0 = "DATE FAIL"
903   0725           GRAFX0 = 291 + PSKID
904   0727           IF PSKID = 5 THEN GRAFX0 = 291
905   0729           IF PSKID = 0 THEN GRAFX0 = 290
906   0731           IF PTYPE = 1 THEN PRINT ADI,"WHITE ALARM %p %p GRAPHICS %i",EPNS0,STRNG0,GRAFX0
907   0732   NXT:    IF FSTART = 1 THEN
908                     BEGIN
909   0733              COLOR0 = 0
910   0734              GRAFX0 = 0
911   0735              STRNG1 =" "
912   0736              FSTART = 0
913   0737              PRINT ADI,"%p",STRNG0
914                     END
915   0738           RETURN
916          ;
917   0740   ERRSUB: IF(ERRNUM() = -187)THEN ERSRET
918                  ELSE
919                   BEGIN
920   0741            ERRN = ERRNUM()
921   0742            ERRL = ERRLIN()
922   0743            PRINT ADI,"ERROR # %i AT LINE %i, ILOOP = %i",errn,errl,ILOOP
923   0744            ERSCLR
924                   END
925   0745           GOTO AGAIN
926          ;
927   0746   HLD:    HRSCLR
928   0747           WAITUNTIL HLDSTP != 0
929   0748           ON HLDSTP GOTO INIT,AGAIN,TERM
930          ;
931          ; TERM: FABORT = 0
932   0749   TERM:   STRNG0 = "APBATCH TERMINATED    "
933   0750           ILOOP = 96
934   0751           FSTART = 1
935   0752           TIMEROFF STMR1
```

```
936   0753            TIMERCLR STMR1
937   0754            FLDALM = 0
938   0755            OPEN 'HS-AP-SIL'
939   0756            GOSUB CYCL
940                   END

      Total Errors Detected      0     Total Warnings      0
          Sequence Program Size = 14768 Bytes, 923 Pages
                205 Forward Reference(s)
```

AWBATCH

```
1           ;       A W B A T C H . S E Q                                   VERSION 2
2           ;
3           ;       Function:       Sequence program for Lamp function,
4           ;                       message display on graphics, and
5           ;                       printer messages.
6           ;
7           ;       Written By      : R. N. Kersey
8           ;       Date            : 05-31-92
9           ;       Facility        : 242-A Evaporator Bldg.
10          ;       Machine         : VAX 4104-A
11          ;       Op. System      : VAX/VMS 5.5-2AH
12          ;       D/3 Version     : 6.3-2 BASELINE
13          ;       Compiler        : SIC
14          ;       Build command   : @[320,20]AWBATCH
15          ;
16          ;       UNIT = 3, SKDKEY = 8, 9, 10, 11
17          ;
18          ;       Assoc. UPF:     [300,1]_TNKFRM.___      TNKFRM.LST
19          ;       Assoc. RPF:     [300,1]_FARM.___        FARM.LST
20          ;       SDB:            [300,1]PCM1ASDB.___.    PCM1ASDB.CMD
21          ;
22          ;       POINTER   SDB VAR.      FUNCTION
23          ;       -------   ---------     ----------------------------------------
24          ;       PSTAT     INTn          Current EPN alarm state is an integer tied to
25          ;                               graphic flashing string of value:
26          ;       .         (n=0-100)             0 = previously not alarmed
27          ;       .                               1 = previously alarmed, not ack
28          ;       .                               2 = previously alarmed, is ack
29          ;       .                               3 = alarm is inhibited
30          ;       .                               4 = input is off scan
31          ;       .                               5 = input is bad
32          ;       PDIN      'EPN'         ---
33          ;       PTYPE     IALMn         alarm type 0 = none, 1 = white, 2 = yellow,
34          ;                                  3 = yellow and yellow, 4 = red,
35          ;                                  6 = yellow and red
36          ;       PREV      FXn           Reverse acting (not fail safe) flag (1=set)
37          ;       PSKID     SKIDn         Tracks which graphic has an alarm
38          ;       PPASS     FPASn         If set (1), turn lamp off when alarm is ACKnowledged
39          ;       PSN       FSNn          Flag for the graphic to ACKnowledge
40          ;       ---       FHOLD         Flag to update the lamps
41          ;       ---       FNING         Flag for DOG1A.C micro select
42          ;       ---       ACK           Flag for C program to ACK system wide
43          ;
44          ; MODIFICATION LOG:
45          ;
46          ;       12-DEC-91       Include MCC shutdown, pump shutdown, and R. Kersey
47          ;                       30 second SKID update.
48          ;
49          ;       05-APR-93       Modified MCC OFF white alarm on all 5           M. Teats
50          ;                       YS- signals to pump OFF white alarm because
51          ;                       the YS DINs do not indicate MCC status - they
52          ;                       indicate the pump status.
53          ;
54          ;-mct1+ 13-MAY-93       Remove pump shutdown DEV white alarms, which    M. Teats
55          ;                       are based on the analog signals, because they
```

```
56          ;                       are not needed due to the 4/5/93 mods.  See
57          ;                       CCN# 93MAY13-01.
58          ;
59          ;       10-JUN-93       Added comments and baselined for        M. Teats
60          ;                       D/3 version 3.3.  CCN# 93JUN01-01
61          ;
62          ;       15-DEC-94       Revised header to indicate the new VMS          J. Berger
63          ;                       system. Rebaselined document. Removed
64          ;                       obsolite Select() statement.
65          ;
66          ;  END MOD LOG
67          ;
68          UNIT    AWFARM
69          ;
70                  PUMP == FLSPIT
71                  AUTO == 'FQI-AW-1':CB_FLG:1          ; For FQI-RESET auto mode check
72                  CALOUT == 'FQI-AW-1':CB_OTVL:1       ; For FQI-RESET calc block outvalue
73          ;
74  0000            GETUNIT TRIP                         ; Which sequence unit is program running
75  0001            IF TRIP != 3 THEN                    ; Abort if running in wrong seq. unit
76                     BEGIN
77  0002               FSTART = 1
78  0003               ILOOP = 54
79  0004               GOSUB CYCL
80  0005               FSTART = 1
81  0006               PRINT ADI," >>>>>  ATTEMPT TO RUN AWBATCH IN WRONG UNIT"
82  0007               GOTO TERM
83                     END
84          ;       IF !SELECT() THEN GOTO AGAIN
85          ;
86          ;       B E G I N   I N I T I A L I Z I N G :
87          ;
88  0008            NOP
89  0009    INIT:   LET FABORT = 0                  ; If FABORT = 1 then abort program
90  0010            LET FSTART = 0                  ; Program start flag = 0
91  0011            LET ERRN = 0                    ; Set error number to zero
92  0012            LET ERRL = 0                    ; Set error line to zero
93  0013            LET FLDALM = 0
94  0014            LET FSL7 = 0                    ; For FQI-RESET manual mode flag
95  0015            LAMP 0,11,0                     ; Turn off the lamps
96  0016            LAMP 0,10,0
97  0017            LAMP 0,9,0
98  0018            LAMP 0,8,0
99  0019            ASSIGN PDEV = AWFARM                        ; First of 6 alarm ack devices
100 0020            FOR ILOOP = 1 TO 6
101 0021            PDEV:DEV_TMP0 = 1                           ; Set all devices to ack state
102 0022            IF AND(PDEV:DEV_ALMCT,16) THEN              ; If alarm device is inha, remove the inha
103 0023              PDEV:DEV_ALMCT=XOR(PDEV:DEV_ALMCT,16)
104 0024            IF AND(PDEV:DEV_ALMCT,32) THEN              ; If alarm device is ack, clear it
105 0025               PDEV:DEV_ALMCT = XOR(PDEV:DEV_ALMCT,32)
106 0026            STEP PDEV BY 1
107 0027            NEXT ILOOP
108         ;
109         ;       First, set all flags and integers:
110         ;
```

```
111   0028          ASSIGN PDIN = 'YS-AW-101'                    ; Assign pointer to first DIN
112   0029          ASSIGN PSTAT = INT0
113   0030          ASSIGN PTYPE = IALM0
114   0031          ASSIGN PREV = FX0
115   0032          ASSIGN PSKID = SKID0
116   0033          ASSIGN PPASS = FPAS0
117         ;
118   0034          LET FARM = 0                                 ; Set up values for different graphics
119   0035          SERPIT = 1                                   ; SKID of each EPN indicates graphic
120   0036          VLVPIT = 2
121   0037          HVAC = 3
122   0038          BLD271 = 4
123   0039          PUMP = 5
124         ;
125   0040          LEAK0 = 11                                   ; FARM
126   0041          LEAK1 = 63                                   ; not used
127   0042          LEAK2 = 9                                    ; VALVE PIT
128   0043          LEAK3 = 10                                   ; HVAC
129   0044          LEAK4 = 8                                    ; AW-271
130         ;
131   0045          FOR ILOOP = 0 TO 46                          ; For DINs
132   0046          LET PPASS =.0
133   0047          LET PREV = 0
134   0049          IF ILOOP > 31 & ILOOP < 36 THEN GOTO SKP
135   0051          IF ILOOP = 44 THEN GOTO SKP
136   0052          LET PTYPE = 2
137         ;       BLET PSTAT = 0
138   0053          IF AND(PDIN:DIN_DFLG,3) THEN                 ; Check for alarms (must be red or yellow)
139                   BEGIN
140   0054            LET PTYPE = 2
141   0056            IF AND(PDIN:DIN_DFLG,1) THEN LET PREV = 1          ; Reverse acting
142   0058            IF AND(PDIN:DIN_DFLG,4) THEN LET PTYPE = 4         ; Set to red critical alarm
143                   END
144   0060          IF ILOOP < 17 THEN LET PSKID = FARM          ; These are set by IPN order
145   0062          IF (ILOOP > 16 & ILOOP < 31) THEN LET PSKID = VLVPIT
146   0064          IF ILOOP = 28 THEN LET PSKID = BLD271
147   0066          IF (ILOOP > 30 & ILOOP < 39) THEN LET PSKID = HVAC
148   0068          IF ILOOP > 38 THEN LET PSKID = BLD271
149   0070          IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
150   0071    SKP:  STEP PSTAT BY 1
151   0072          STEP PDIN BY 1
152   0073          STEP PTYPE BY 1
153   0074          STEP PREV BY 1
154   0075          STEP PSKID BY 1
155   0076          STEP PPASS BY 1
156   0077          NEXT ILOOP
157         ;
158         ;-mct1- abandoned code for white alarms tied to pump devices:
159         ;
160         ;       FOR ILOOP = 47 TO 51
161         ;       PSTAT = 0
162         ;       PTYPE = 1
163         ;       PREV = 0
164         ;       PSKID = 0
165         ;       PPASS = 1
```

```
166             ;       STEP PSTAT BY 1
167             ;       STEP PTYPE BY 1
168             ;       STEP PREV BY 1
169             ;       STEP PSKID BY 1
170             ;       STEP PPASS BY 1
171             ;       NEXT ILOOP
172             ;-mct1-
173                     ;
174                     ;  Set the white alarms:
175                     ;
176     0078            LET IALM0 = 1
177     0079            LET IALM5 = 1
178     0080            LET IALM8 = 1
179     0081            LET IALM11= 1
180     0082            LET IALM14 = 1
181     0083            LET IALM32 = 0
182     0084            LET IALM33 = 0
183     0085            LET IALM34 = 0
184     0086            LET IALM35 = 0
185     0087            LET IALM44 = 0
186     0088            LET IALM46 = 1
187             ;
188     0089            FX0 = 0
189     0090            FX5 = 0
190     0091            FX8 = 0
191     0092            FX11 = 0
192     0093            FX14 = 0
193     0094            FX28 = 1
194             ;
195     0095            FPAS0 = 1
196     0096            FPAS5 = 1
197     0097            FPAS8 = 1
198     0098            FPAS11 = 1
199     0099            FPAS14 = 1
200             ;
201             ;       E N D  ·I N I T I A L I Z I N G
202             ;
203             ;       S E N D   P R O G R A M   S T A R T   M E S S A G E
204             ;
205     0100            ILOOP = 53
206     0101            FSTART = 1
207     0102            TIMERCLR STMR2
208     0103            TIMERON STMR2
209     0104            GOSUB CYCL
210     0105            ON ERROR GOSUB ERRSUB
211             ;
212             ;       B E G I N   P R O G R A M   L O O P :
213             ;
214     0106    AGAIN:  MARK STEP "AGAIN",0,0
215             ;       IF !SELECT() THEN FNING = 0
216             ;          ELSE FNING = 1                    ; Flag 480 determines if selected micro for DOG1A.C
217             ; JH01  WAITUNTIL SELECT() ¦ FABORT          ; Wait until micro selected or program aborted
218             ;       IF FABORT THEN GOTO TERM
219     0108            IF STMR2 > 30 THEN FHOLD = 1
220             ;       IF FNING = 0 THEN FHOLD = 1
```

```
221   0109          WAIT 1
222           ;
223   0110          DEV0 = 0                              ; Lamp 11
224   0111          DEV1 = 0                              ; not used
225   0112          DEV2 = 0                              ; Lamp 9
226   0113          DEV3 = 0                              ; Lamp 10
227   0114          DEV4 = 0                              ; Lamp 8
228   0115          DEV5 = 0                              ; not used
229           ;
230   0116          ASSIGN PDEV = AWFARM
231   0117          ASSIGN PSN = FSN0
232   0118          ASSIGN PACK = FLEAK0
233   0119          FOR ILOOP = 0 TO 5                    ; For the 6 alarm devices (2 unused)
234   0120          PSN = 0
235                 ;
236                 ;  If alarm device is alarmed and its ALMCT indicates alarms are ack'd:
237                 ;
238   0121          IF (PDEV:DEV_TMP0 = 2) & AND(PDEV:DEV_ALMCT,32) THEN
239                    BEGIN
240   0122             PDEV:DEV_TMP0 = 1                  ; Put device in ack state
241   0123             LET PSN = 1                        ; Set flag for graphic to ack
242   0124             LET FLGACK = 1                     ; WATCH.C spawns ACKFLG.C, which resets flag
243   0125             LET ACK = 1                        ; Set flag for C program to ACK system wide
244   0126             LET PACK = 1
245   0127             ASSIGN PMSG0 = LEAK0
246   0128             STEP PMSG0 BY ILOOP
247   0130             IF ILOOP !=1 THEN LAMP 0,PMSG0,1   ; Flash lamp
248                    END
249   0131          STEP PDEV BY 1
250   0132          STEP PSN BY 1
251   0133          STEP PACK BY 1
252   0134          NEXT ILOOP
253           ;
254           ;     Field Alarm Silence feature:
255           ;
256   0135          IF(ACK)THEN
257                   BEGIN
258   0136            TIMERCLR STMR1
259   0137            TIMERON STMR1
260   0138            FLDALM = 1
261                   END
262   0140          IF(FLDALM)THEN IF STMR1 >= 30 THEN    ; If field alarm then ack in 30 sec
263                   BEGIN
264   0141            TIMEROFF STMR1
265   0142            TIMERCLR STMR1
266   0143            FLDALM = 0
267   0144            OPEN 'HS-AW-SIL'                    ; Silence alarm from PCM
268                   END
269                 ;
270                 ;  If field alarm ack then set flag and start timer:
271                 ;
272   0146          IF(RESET)THEN IF LMPVAR < 2 THEN
273                   BEGIN
274   0147            FLDALM = 1
275   0148            TIMERON STMR1
```

```
276   0149              RESET = 0
277                     END
278   0151            IF LMPVAR = 2 THEN RESET = 1
279           ;
280           ;       BEGIN DINs:
281           ;
282   0152            ASSIGN PDIN = 'YS-AW-101'               ; First DIN in AWFARM
283   0153            ASSIGN PTYPE = IALM0
284   0154            ASSIGN PREV = FX0
285   0155            ASSIGN PSTAT = INT0
286   0156            ASSIGN PSKID = SKID0
287   0157            ASSIGN PPASS = FPAS0
288   0158            CURENT = 0                              ; Used to set lamp status
289   0159            NXT = 0                                 ; Temporary INT
290   0160            BITVAL = 0
291           ;
292   0161    LOP:    FOR ILOOP = 1 TO 47
293   0162            IF PTYPE THEN                                       ; If some alarm
294                       BEGIN                                   ;be1
295   0164                IF AND(PDIN:DIN_ALMCT,16) THEN LET PSTAT = 3    ; If EPN is inha
296   0166                IF AND(PDIN:DIN_ALMCT,4) THEN LET PSTAT = 5             ; If EPN is ibad
297   0168                IF !AND(PDIN:DIN_FLG,8) THEN LET PSTAT = 4              ; If EPN is dis
298   0171                IF ! PSTAT THEN IF PREV = PDIN:DIN_VAL THEN GOSUB ALM   ; If not prev alarmed
299   0172                IF PSTAT THEN
300                           BEGIN                               ;be2
301   0173                    IF PSTAT < 3 THEN                               ; If not inha
302                               BEGIN                           ;be3
303   0174                        IF PSTAT = 1 THEN
304                                   BEGIN                       ;be4
305   0175                            IF AND(PDIN:DIN_ALMCT,32) THEN          ; If alarm is ack
306                                       BEGIN                   ;be5
307   0176                                LET PSTAT = 2
308   0177                                FHOLD = 1
309                                       END                     ;en5
310   0178                            IF ACK THEN                             ; If a graphic has been ack
311                                       BEGIN                   ;be6 -  go find which graphic
312   0179                                CRIT
313   0180                                ASSIGN PSN = FSN0        ; Routine to find graphic
314   0181                                STEP PSN BY PSKID
315   0182                                IF PSN THEN                     ; If this is the graphic
316                                           BEGIN               ;be7 -  then ack it
317   0183                                    LET PSTAT = 2
318   0185                                    IF PTYPE = 1 THEN PDIN:DIN_ALMCT = OR(PDIN:DIN_ALMCT,32)
319   0186                                    FPASS = 1
320   0187                                    GOSUB ACKK              ; Print signal was ack
321                                           END                 ;en7
322                                       END                     ;en6
323                                   END                         ;en4
324   0189                            IF PREV != PDIN:DIN_VAL THEN GOSUB CLR  ; If alarm has cleared
325   0190                            ASSIGN PMSG0 = DEV0                     ; Find active alarms & graphic
326   0191                            STEP PMSG0 BY PSKID
327   0192                            IF PPASS THEN
328                                       BEGIN
329   0194                                IF PSTAT = 1 THEN PMSG0 = OR(PMSG0,PSTAT)
330                                       END
```

```
331   0195                        ELSE PMSGO = OR(PMSGO,PSTAT)
332                             END                                          ;en3
333   0196                    ELSE IF !AND(PDIN:DIN_ALMCT,16)&!AND(PDIN:DIN_ALMCT,4)&AND(PDIN:DIN_FLG,8)THEN
334                               BEGIN
335   0197                        LET PSTAT = 0
336   0198                        FHOLD = 1
337                               END
338                           END                                            ;en2
339                       END                                              ;en1
340   0200              ELSE IF PSTAT THEN PSTAT = 0
341   0201              STEP PDIN BY 1
342   0202              STEP PTYPE BY 1
343   0203              STEP PREV BY 1
344   0204              STEP PSTAT BY 1
345   0205              STEP PSKID BY 1
346   0206              STEP PPASS BY 1
347   0207              NEXT ILOOP
348             ;
349             ;         END DINs
350             ;
351   0208              DEV6 = 0
352   0209              DEV7 = 0
353   0210              DEV8 = 0
354   0211              DEV9 = 0
355   0212              DEV10 = 0
356   0213              DEV11 = 0
357             ;
358             ;-mct1-   abandoned code for white alarms tied to pump devices:
359             ;
360             ;AUX:     ASSIGN PEPN = 'P-AW-101'
361             ;
362             ;         FOR ILOOP = 48 TO 52
363             ;         IF PTYPE THEN
364             ;            BEGIN
365             ;            IF AND(PEPN:DEV_ALMCT,16)THEN BLET PSTAT = 3
366             ;            IF AND(PEPN:DEV_ALMCT,4) THEN BLET PSTAT = 5
367             ;            IF !AND(PEPN:DEV_FLG,8) THEN BLET PSTAT = 4
368             ;            IF PSTAT < 3 THEN
369             ;               BEGIN
370             ;               IF (PSTAT = 1) & AND(PEPN:DEV_ALMCT,32)THEN
371             ;                  BEGIN
372             ;                  BLET PSTAT = 2
373             ;                  FHOLD = 1
374             ;                  END
375             ;               IF ACK THEN
376             ;                  BEGIN
377             ;                  ASSIGN PSN = FSNO
378             ;                  STEP PSN BY PSKID
379             ;                  IF PSN & PSTAT = 1 THEN
380             ;                     BEGIN
381             ;                     BLET PSTAT = 2
382             ;                     IF PTYPE = 1 THEN PEPN:DEV_ALMCT = OR(PEPN:DEV_ALMCT,32)
383             ;                     FPASS = 1
384             ;                     GOSUB ACKK
385             ;                     END
```

```
386             ;                  END
387             ;               IF !PSTAT THEN
388             ;                  BEGIN
389             ;                  IF AND(PEPN:DEV_STAT,16)THEN
390             ;                     BEGIN
391             ;                     BLET PSTAT = 1
392             ;                     IF AND(PEPN:DEV_ALMCT,32)THEN PEPN:DEV_ALMCT = XOR(PEPN:DEV_ALMCT,32)
393             ;                     ASSIGN PDEV = AWFARM
394             ;                     STEP PDEV BY PSKID
395             ;                     PDEV:DEV_TMP0 = 2
396             ;                     GOSUB CYCL
397             ;                     END
398             ;                  END
399             ;               ELSE IF !AND(PEPN:DEV_STAT,16)THEN
400             ;                  BEGIN
401             ;                  BLET PSTAT = 0
402             ;                  FHOLD = 1
403             ;                  IF AND(PEPN:DEV_ALMCT,32)THEN PEPN:DEV_ALMCT = XOR(PEPN:DEV_ALMCT,32)
404             ;                  END
405             ;               IF PSTAT THEN
406             ;                  BEGIN
407             ;                  ASSIGN PMSG0 = DEV6
408             ;                  STEP PMSG0 BY PSKID
409             ;                  IF PPASS THEN
410             ;                     BEGIN
411             ;                     IF PSTAT = 1 THEN PMSG0 = OR(PMSG0,PSTAT)
412             ;                     END
413             ;                  ELSE PMSG0 = OR(PMSG0,PSTAT)
414             ;                  END
415             ;               END
416             ;            ELSE IF !AND(PEPN:DEV_ALMCT,16)&!AND(PEPN:DEV_ALMCT,4)&AND(PEPN:DEV_FLG,8)THEN
417             ;               BEGIN
418             ;               BLET PSTAT = 0
419             ;               FHOLD = 1
420             ;               END
421             ;            END
422             ;         STEP PEPN BY 1
423             ;         STEP PTYPE BY 1
424             ;         STEP PREV BY 1
425             ;         STEP PSTAT BY 1
426             ;         STEP PSKID BY 1
427             ;         STEP PPASS BY 1
428             ;         NEXT ILOOP
429             ;-mct1-
430             ;
431   0214      S2ND:     ACK = 0
432   0215                SORT = 0
433             ;
434   0216                DEV12 = OR(DEV0,DEV6)
435   0217                DEV13 = OR(DEV1,DEV7)
436   0218                DEV14 = OR(DEV2,DEV8)
437   0219                DEV15 = OR(DEV3,DEV9)
438   0220                DEV16 = OR(DEV4,DEV10)
439   0221                DEV17 = OR(DEV5,DEV11)
440             ;
```

```
441   0222          ASSIGN PMSG0 = DEV12
442   0223          ASSIGN PMSG1 = FARM
443   0224          FOR LLOOP = 1 TO 5
444   0225          IF PMSG0 THEN
445                   BEGIN
446   0227            IF AND(PMSG0,1) THEN PMSG1 = 2
447   0228            ELSE PMSG1 = 1
448                   END
449   0229          ELSE PMSG1 = 0
450   0230          STEP PMSG0 BY 1
451   0231          STEP PMSG1 BY 1
452   0232          NEXT LLOOP
453         ;
454   0233          IF FHOLD THEN
455                   BEGIN
456   0234            LAMP 0,11,FARM
457   0235            LAMP 0,10,HVAC
458   0236            LAMP 0,9,VLVPIT
459   0237            LAMP 0,8,BLD271
460   0238            FHOLD = 0
461   0239            TIMERCLR STMR2
462   0240            TIMERON STMR2
463                   END
464   0241          MARK STEP "BOTTOM",0,0
465   0242          NOCRIT
466         ;
467         ;       B E G I N   F Q I - R E S E T :
468         ;
469         ;       Function:       For totalizers that have a CALC block which performs the
470         ;                       integration, set the output of the CALC block to zero
471         ;                       after the FQI loop has been placed in AUTO from MANUAL.
472         ;
473   0244          IF !AND(AUTO,64) THEN FSL7 = 1                  ; If manual, set manual flag
474   0245          IF AND(AUTO,64) & AND(FSL7,1) THEN              ; For manual-auto transition
475                   BEGIN
476   0246            FSL7 = 0
477   0247            CALOUT = 0.0
478                   END
479         ;
480         ;       E N D   F Q I - R E S E T
481         ;
482   0248          GOTO AGAIN
483         ;
484         ;       E N D   P R O G R A M   L O O P
485         ;
486         ;       BEGIN CLEAR, ALARM, CYCLE, ACKNOWLEDGE, MESSAGE, ERROR, AND TERMINATE SUBROUTINES:
487         ;
488   0249  CLR:    LET PSTAT = 0
489   0251          IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
490   0252          FHOLD = 1
491   0253          RETURN
492         ;
493   0254  ALM:    LET PSTAT = 1           ; Print message, flash lamp, horn on, scroll message at bottom of graphic
494   0255          CRIT                                            ; Critical
495   0257          IF AND(PTYPE,3)THEN FALMON = 1                  ; Audible alarm on
```

```
496   0258          IF AND(PDIN:DIN_ALMCT,32)THEN                    ; Clear the scan block
497   0259             PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
498   0260          ASSIGN PMSG0 = LEAK0
499   0261          STEP PMSG0 BY PSKID
500   0262          LAMP 0,PMSG0,2                                   ; Lamp flash
501   0263          ASSIGN PDEV = AWFARM                             ; Find DEV and turn it on
502   0264          STEP PDEV BY PSKID
503   0265          PDEV:DEV_TMP0 = 2
504         ;
505         ;       Scrolling alarm summary region in graphic (message, time, date, alarm type):
506         ;
507   0266  CYCL:   LSG4 = LSG3                                      ; LSG = message
508   0267          LSG3 = LSG2
509   0268          LSG2 = LSG1
510   0269          LSG1 = LSG0
511   0270          COLOR4 = COLOR3                                  ; COLOR = type of alarm
512   0271          COLOR3 = COLOR2
513   0272          COLOR2 = COLOR1
514   0273          COLOR1 = COLOR0
515   0274          EPNS4 = EPNS3                                    ; EPN
516   0275          EPNS3 = EPNS2
517   0276          EPNS2 = EPNS1
518   0277          EPNS1 = STRNG1
519   0278          GRAFX4 = GRAFX3                                  ; GRAFX = graphic number
520   0279          GRAFX3 = GRAFX2
521   0280          GRAFX2 = GRAFX1
522   0281          GRAFX1 = GRAFX0
523   0282          SDATE4 = SDATE3                                  ; SDATE = time and date
524   0283          SDATE3 = SDATE2
525   0284          SDATE2 = SDATE1
526   0285          SDATE1 = SDATE0
527         ;
528         ;       ALARM TYPER STRINGS:
529         ;
530   0286  ACKK:   FHOLD = 1
531   0287          ON ILOOP GOTO S1,S2,S3,S4,S5,S6,S7,S8,S9,S10
532   0288          ON ILOOP - 10 GOTO S11,S12,S13,S14,S15,S16,S17,S18,S19,S20
533   0289          ON ILOOP - 20 GOTO S21,S22,S23,S24,S25,S26,S27,S28,S29,S30
534   0290          ON ILOOP - 30 GOTO S31,S32,S33,S34,S35,S36,S37,S38,S39,S40
535   0291          ON ILOOP - 40 GOTO S41,S42,S43,S44,S45,S46,S47,S48,S49,S50
536   0292          ON ILOOP - 50 GOTO S51,S52,S53,S54,S55
537   0293          GOTO AMSG
538   0294  S1:     STRNG0 = "PUMP AW-101 OFF       "
539   0295          STRNG1 ="YS-AW-101"
540   0296          GOTO AMSG
541   0297  S2:     STRNG0 = "AW-101 ANN RD HI/FAIL "
542   0298          STRNG1 ="RSH-AW101"
543   0299          GOTO AMSG
544   0300  S3:     STRNG0 = "AW-01C RAD HI/FAIL    "
545   0301          STRNG1 = "RSH-AW01C"
546   0302          GOTO AMSG
547   0303  S4:     STRNG0 = "AW-102 ANN RD HI/FAIL "
548   0304          STRNG1 = "RSH-AW102"
549   0305          GOTO AMSG
550   0306  S5:     STRNG0 = "AW-02C RAD HI/FAIL    "
```

```
551  0307          STRNG1 = "RSH-AW02C"
552  0308          GOTO AMSG
553  0309  S6:     STRNG0 = "PUMP AW-103 OFF       "
554  0310          STRNG1 = "YS-AW-103"
555  0311          GOTO AMSG
556  0312  S7:     STRNG0 = "AW-103 ANN RD HI/FAIL "
557  0313          STRNG1 = "RSH-AW103"
558  0314          GOTO AMSG
559  0315  S8:     STRNG0 = "AW-03C RADN HIGH/FAIL "
560  0316          STRNG1 = "RSH-AW03C"
561  0317          GOTO AMSG
562  0318  S9:     STRNG0 = "PUMP AW-104 OFF       "
563  0319          STRNG1 = "YS-AW-104"
564  0320          GOTO AMSG
565  0321  S10:    STRNG0 = "AW-104 ANN RD HI/FAIL "
566  0322          STRNG1 = "RSH-AW104"
567  0323          GOTO AMSG
568  0324  S11:    STRNG0 = "AW-04C RADN HIGH/FAIL "
569  0325          STRNG1 = "RSH-AW04C"
570  0326          GOTO AMSG
571  0327  S12:    STRNG0 = "PUMP AW-105 OFF       "
572  0328          STRNG1 = "YS-AW-105"
573  0329          GOTO AMSG
574  0330  S13:    STRNG0 = "AW-105 ANN RD HI/FAIL "
575  0331          STRNG1 = "RSH-AW105"
576  0332          GOTO AMSG
577  0333  S14:    STRNG0 = "AW-05C RADN HIGH/FAIL "
578  0334          STRNG1 = "RSH-AW05C"
579  0335          GOTO AMSG
580  0336  S15:    STRNG0 = "PUMP AW-106 OFF       "
581  0337          STRNG1 = "YS-AW-106"
582  0338          GOTO AMSG
583  0339  S16:    STRNG0 = "AW-106 ANN RD HI/FAIL "
584  0340          STRNG1 = "RSH-AW106"
585  0341          GOTO AMSG
586  0342  S17:    STRNG0 = "AW-06C RADN HIGH/FAIL "
587  0343          STRNG1 = "RSH-AW06C"
588  0344          GOTO AMSG
589  0345  S18:    STRNG0 = "AW TL-21 LINE LEAK    "
590  0346          STRNG1 = "LDS-TL021"
591  0347          GOTO AMSG
592  0348  S19:    STRNG0 = "AW CIRCUIT TROBL/LEAK "
593  0349          STRNG1 = "LDSTL2223"
594  0350          GOTO AMSG
595  0351  S20:    STRNG0 = "AW DRAIN ENCASE LEAK  "
596  0352          STRNG1 = "LDS-AW-DR"
597  0353          GOTO AMSG
598  0354  S21:    STRNG0 = "AW PROCESS PITS LEAK  "
599  0355          STRNG1 = "LDS-AW-PP"
600  0356          GOTO AMSG
601  0357  S22:    STRNG0 = "AW SLURRY ENCASE LEAK "
602  0358          STRNG1 = "LDS-AW-SL"
603  0359          GOTO AMSG
604  0360  S23:    STRNG0 = "AW SUPRN8 ENCASE LEAK "
605  0361          STRNG1 = "LDS-AW-SN"
```

```
606   0362            GOTO AMSG
607   0363   S24:     STRNG0 = "AW SN-269 LINE LEAK   "
608   0364            STRNG1 = "LDS-SN269"
609   0365            GOTO AMSG
610   0366   S25:     STRNG0 = "AW SN-270 LINE LEAK   "
611   0367            STRNG1 = "LDS-SN270"
612   0368            GOTO AMSG
613   0369   S26:     STRNG0 = "AW CLEAKOUT BOX LEAK  "
614   0370            STRNG1 = "LDS-COBAW"
615   0371            GOTO AMSG
616   0372   S27:     STRNG0 = "AW SN FLSH LN PRES HI "
617   0373            STRNG1 = "PSH-AWSN1"
618   0374            GOTO AMSG
619   0375   S28:     STRNG0 = "AW FEEDLNE PRESSUR HI "
620   0376            STRNG1 = "PSH-AWSN2"
621   0377            GOTO AMSG
622   0378   S29:     STRNG0 = "AW FARM GENERAL ALARM "
623   0379            STRNG1 = "QS-AW-271"
624   0380            GOTO AMSG
625   0381   S30:     STRNG0 = "AW SERPIT RAD HI/FAIL "
626   0382            STRNG1 = "RSH-SP-AW"
627   0383            GOTO AMSG
628   0384   S31:     STRNG0 = "AW SERVICE PIT LEAK   "
629   0385            STRNG1 = "LDS-SP-AW"
630   0386            GOTO AMSG
631   0387   S32:     STRNG0 = "AW-FRM TNK EXH RAD HI "
632   0388            STRNG1 = "RSH-AW-K1"
633   0389            GOTO AMSG
634   0390   S33:     STRNG0 = "K1 SK MN SMPL PMP OFF "
635   0391            STRNG1 = "PSH-AWK11"
636   0392            GOTO AMSG
637   0393   S34:     STRNG0 = "K1 REC SMPL PUMP OFF  "
638   0394            STRNG1 = "PSH-AWK12"
639   0395            GOTO AMSG
640   0396   S35:     STRNG0 = "K2 STK MN SPL PMP OFF "
641   0397            STRNG1 = "PSH-AWK21"
642   0398            GOTO AMSG
643   0399   S36:     STRNG0 = "K2 REC SMPL PUMP OFF  "
644   0400            STRNG1 = "PSH-AWK22"
645   0401            GOTO AMSG
646   0402   S37:     STRNG0 = "AW TANK EXH FANS OFF  "
647   0403            STRNG1 = "QXS-AWK15"
648   0404            GOTO AMSG
649   0405   S38:     STRNG0 = "ANN EX RD/FAN HI/FAIL "
650   0406            STRNG1 = "RSH-AW-K2"
651   0407            GOTO AMSG
652   0408   S39:     STRNG0 = "AW TNK EX RADMON FAIL "
653   0409            STRNG1 = "QXSAWK1A2"
654   0410            GOTO AMSG
655   0411   S40:     STRNG0 = "241-AW RADN HI/FAIL   "
656   0412            STRNG1 = "RSH-AW-12"
657   0413            GOTO AMSG
658   0414   S41:     STRNG0 = "271-AW RADN HI/FAIL   "
659   0415            STRNG1 = "RSHAW2711"
660   0416            GOTO AMSG
```

```
661   0417   S42:    STRNG0 = "271-AW RADN HI/FAIL   "
662   0418           STRNG1 = "RSHAW2712"
663   0419           GOTO AMSG
664   0420   S43:    STRNG0 = "AW AREA RADN BYPASS   "
665   0421           STRNG1 = "YS-AW-ARM"
666   0422           GOTO AMSG
667   0423   S44:    STRNG0 = "271-AW BLD POWER LOSS "
668   0424           STRNG1 = "JXS-AW271"
669   0425           GOTO AMSG
670   0426   S45:    STRNG0 = "EPN DELETED           "
671   0427           STRNG1 = "NO EPN"
672   0428           GOTO AMSG
673   0429   S46:    STRNG0 = "AW-FARM MASTER SHTDWN "
674   0430           STRNG1 = "YS-AW-MSD"
675   0431           GOTO AMSG
676   0432   S47:    STRNG0 = "AW HEAT TRACE OVRTMP  "
677   0433           STRNG1 = "TXS-AW-2"
678   0434           GOTO AMSG
679          ;
680          ;-mct1-  abandoned code for white alarms tied to pump devices:
681          ;
682          S48:    ;       STRNG0 = "PUMP AW-101 SHUTDOWN  "
683          ;       STRNG1 = "P-AW-101"
684          ;       GOTO AMSG
685          S49:    ;       STRNG0 = "PUMP AW-103 SHUTDOWN  "
686          ;       STRNG1 = "P-AW-103"
687          ;       GOTO AMSG
688          S50:    ;       STRNG0 = "PUMP AW-104 SHUTDOWN  "
689          ;       STRNG1 = "P-AW-104"
690          ;       GOTO AMSG
691          S51:    ;       STRNG0 = "PUMP AW-105 SHUTDOWN  "
692          ;       STRNG1 = "P-AW-105"
693          ;       GOTO AMSG
694          S52:    ;       STRNG0 = "PUMP AW-106 SHUTDOWN  "
695          ;       STRNG1 = "P-AW-106"
696          ;       GOTO AMSG
697          ;-mct1-
698          ;
699   0435   S53:    STRNG0 = "AWBATCH PROGRAM START "
700   0436           GOTO AMSG
701   0437   S54:    STRNG0 = "AWBATCH IN WRONG UNIT "
702   0438           GOTO AMSG
703   0439   S55:    STRNG0 = "AWBATCH TERMINATING   "
704   0440           GOTO AMSG
705          ;
706   0441   AMSG:   IF (FPASS)THEN
707                     BEGIN
708   0442              PRINT ADI,"%p ALARM ACKNOWLEDGE",STRNG1
709   0443              FPASS = 0
710   0444              GOTO NXT
711                     END
712   0446           IF PTYPE = 4 THEN COLOR0 = 3
713   0447              ELSE COLOR0 = PTYPE
714   0448           EPNS0 = STRNG1
715   0449           IN0 = TIMSTR(SSNG0)
```

```
716   0451            IF INO != 8 THEN SSNGO = "TIMEFAIL"
717   0452            INO = APPEND(LSGO,STRNGO,SSNGO)
718   0454            IF INO < 0 THEN LSGO = "APPEND FAILED!!"
719   0455            INO = DATSTR(SDATEO)
720   0457            IF INO != 9 THEN SDATEO = "DATE FAIL"
721   0459            IF PSKID = 1 THEN GRAFXO = 303    ; SERVICE PIT
722   0461            IF PSKID = 2 THEN GRAFXO = 303    ; VALVE PIT
723   0463            IF PSKID = 3 THEN GRAFXO = 303    ; HVAC
724   0465            IF PSKID = 4 THEN GRAFXO = 302    ; BLD-271
725   0467            IF PSKID = 5 THEN GRAFXO = 301    ; PUMP
726   0469            IF PSKID = 0 THEN GRAFXO = 300    ; FARM
727   0471            IF PTYPE = 1 THEN PRINT ADI,"WHITE ALARM %p %p GRAPHICS %i",EPNSO,STRNGO,GRAFXO
728   0472  NXT:      IF FSTART = 1 THEN
729                      BEGIN
730   0473               COLORO = 0
731   0474               GRAFXO = 0
732   0475               STRNG1 =" "
733   0476               FSTART = 0
734   0477               PRINT ADI,"%p",STRNGO
735                      END
736   0478            RETURN
737         ;
738   0480  ERRSUB:   IF(ERRNUM() != -187)THEN ERSRET
739                   ELSE
740                    BEGIN
741   0481             ERRN = ERRNUM()
742   0482             ERRL = ERRLIN()
743   0483             PRINT ADI,"ERROR # %i AT LINE %i, ILOOP = %i",errn,errl,ILOOP
744   0484             ERSCLR
745                    END
746   0485            GOTO AGAIN
747         ;
748         ; TERM:   FABORT = 0
749   0486  TERM:     STRNGO = "AWACK TERMINATED    "
750   0487            ILOOP = 55
751   0488            FSTART = 1
752   0489            TIMEROFF STMR1
753   0490            TIMERCLR STMR1
754   0491            FLDALM = 0
755   0492            OPEN 'HS-AW-SIL'
756   0493            GOSUB CYCL
757                   END
```

Total Errors Detected 0 Total Warnings 0
Sequence Program Size = 9408 Bytes, 588 Pages
116 Forward Reference(s)

XYZBATCH

Sequence Instruction Compiler Listing 15:38:41 24-Feb-95 Page 1

```
1           ;       X Y Z B A T C H . S E Q                                 VERSION 2
2           ;
3           ;       Function:       Sequence program for Lamp function,
4           ;                       message display on graphics, and
5           ;                       printer messages.
6           ;
7           ;       Written By      : R. N. Kersey
8           ;       Date            : 05-31-92
9           ;       Facility        : 242-A Evaporator Bldg.
10          ;       Machine         : VAX 4104-A
11          ;       Op. System      : VAX/VMS 5.5-2AH
12          ;       D/3 Version     : 6.3-2 BASELINE
13          ;       Compiler        : SIC
14          ;       Build command   : @[320,20]XYZBATCH
15          ;
16          ;       UNIT = 4, SKDKEY = 12, 13, 14, 15
17          ;
18          ;       Assoc. UPF:     [300,1]_TNKFRM.___      TNKFRM.LST
19          ;       Assoc. RPF:     [300,1]_FARM.___        FARM.LST
20          ;       SDB:            [300,1]PCM1ASDB.___     PCM1ASDB.CMD
21          ;
22          ;       POINTER   SDB VAR.      FUNCTION
23          ;       -------   ---------     ----------------------------------------
24          ;       PSTAT     INTn          Current EPN alarm state is an integer tied to
25          ;                               graphic flashing string of value:
26          ;       .         (n=0-100)             0 = previously not alarmed
27          ;       .                               1 = previously alarmed, not ack
28          ;       .                               2 = previously alarmed, is ack
29          ;       .                               3 = alarm is inhibited
30          ;       .                               4 = input is off scan
31          ;       .                               5 = input is bad
32          ;       PDIN      'EPN'         ---
33          ;       PTYPE     IALMn         alarm type 0 = none, 1 = white, 2 = yellow,
34          ;                                 3 = yellow and yellow, 4 = red,
35          ;                                 6 = yellow and red
36          ;       PREV      FXn           Reverse acting (not fail safe) flag (1=set)
37          ;       PSKID     SKIDn         Tracks which graphic has an alarm
38          ;       PPASS     FPASn         If set (1), turn lamp off when alarm is ACKnowledged
39          ;       PSN       FSNn          Flag for the graphic to ACKnowledge
40          ;       ---       FHOLD         Flag to update the lamps
41          ;       ---       FNING         Flag for DOG1A.C micro select
42          ;       ---       ACK           Flag for C program to ACK system wide
43          ;
44          ;   MODIFICATION LOG:
45          ;
46          ;           11-JUN-93       Added comments.  CCN# 93JUN01-01.           M. Teats
47          ;
48          ;-mct1+  25-JUN-93          Set QSCONSOLE to no alarms, since the       M. Teats
49          ;                           Gamewell is not connected to the MCS.
50          ;                           CCN# 93JUN25-01.
51          ;
52          ;-mct2+  19-JUL-93          Delete QSCONSOLE per ECN 190990, since the  M. Teats
53          ;                           Gamewell is not connected to the MCS.  CCN
54          ;                           93JUL15-01.  Baselined for D/3 version 3.3.
55          ;
```

```
56          ;       15-DEC-94       Revised header to indicate the new VMS          J. Berger
57          ;                       system. Rebaselined document. Removed
58          ;                       obsolite Select() statement.
59          ;
60          ;  END MOD LOG
61          ;
62          UNIT    XYZFRM
63                  COMMON A244                     ; Include PSH-SN215, PSH-SN216
64                  PUMP == FLSPIT
65    0000          GETUNIT TRIP                    ; Which sequence unit is program running in
66    0001          IF TRIP != 4 THEN               ; Abort if running in wrong seq. unit
67                     BEGIN
68    0002             FSTART = 1
69    0003             ILOOP = 38
70    0004             GOSUB CYCL
71    0005             FSTART = 1
72    0006             PRINT ADI," >>>>>  ATTEMPT TO RUN XYZBAT IN WRONG UNIT"*
73    0007             GOTO TERM
74                     END
75          ;       IF !SELECT() THEN GOTO AGAIN
76          ;
77          ;       B E G I N   I N I T I A L I Z I N G :
78          ;
79    0008          NOP
80    0009  INIT:   LET FABORT = 0                  ; Abort program if FABORT = 1
81    0010          LET FSTART = 0                  ; Program start flag
82    0011          LET FLDALM = 0
83    0012          LET FSL6 = 0                    ; QS-204AR1 flag (mct)
84    0013          LET FSL7 = 0                    ; QS-204AR2 flag (mct)
85    0014          LET ERRN = 0                    ; Set error number to zero
86    0015          LET ERRL = 0                    ; Set error line to zero
87    0016          LAMP 0,15,0                     ; Turn off the lamps
88    0017          LAMP 0,14,0
89    0018          LAMP 0,13,0
90    0019          LAMP 0,12,0
91    0020          ASSIGN PDEV = AFARM                             ; First of 5 alarm ack devices
92    0021          FOR ILOOP = 1 TO 5
93    0022          PDEV:DEV_TMP0 = 1                               ; Set all 5 devices to ack state
94    0023          IF AND(PDEV:DEV_ALMCT,16) THEN                  ; If alarm device is inha, remove the inha
95    0024            PDEV:DEV_ALMCT=XOR(PDEV:DEV_ALMCT,16)
96    0025          IF AND(PDEV:DEV_ALMCT,32) THEN                  ; If alarm device is ack, clear it
97    0026             PDEV:DEV_ALMCT = XOR(PDEV:DEV_ALMCT,32)
98    0027          STEP PDEV BY 1
99    0028          NEXT ILOOP
100         ;
101         ;       First, set all flags and integers:
102         ;
103   0029          ASSIGN PDIN = 'PSH-SL100'                       ; Assign pointer to first DIN
104   0030          ASSIGN PSTAT = INT4
105   0031          ASSIGN PTYPE = IALM4
106   0032          ASSIGN PREV = FX4
107   0033          ASSIGN PSKID = SKID4
108   0034          ASSIGN PPASS = FPAS4
109         ;
110   0035          FARM = 0                                        ; Set up values for different graphics
```

```
111   0036          SERPIT = 1                                  ; SKID of each EPN indicates graphic
112   0037          VLVPIT = 2
113   0038          HVAC = 3
114   0039          BLD271 = 4
115         ;       PUMP = 5
116         ;
117   0040          LEAK0 = 15                                  ; AXYZ FARM lamp
118   0041          LEAK1 = 14                                  ; A FARM lamp
119   0042          LEAK2 = 13                                  ; AY FARM lamp
120   0043          LEAK3 = 12                                  ; AZ FARM lamp
121         ;
122   0044          FOR ILOOP = 4 TO 29                         ; For DINs
123   0045          LET PPASS = 0
124   0046          LET PREV = 0
125   0047          LET PTYPE = 2
126         ;       BLET PSTAT = 0
127   0048          IF ILOOP = 29 THEN                          ; -mct2+
128                    BEGIN
129   0049             LET IALM29 = 0                           ; -mct1+
130   0050             GOTO SKP
131                    END
132   0051          IF AND(PDIN:DIN_DFLG,3) THEN                ; Check for alarms (must be red or yellow)
133                    BEGIN
134   0052             LET PTYPE = 2                            ; Default to yellow alarm
135   0054             IF AND(PDIN:DIN_DFLG,1) THEN LET PREV = 1       ; Reverse acting
136   0056             IF AND(PDIN:DIN_DFLG,4) THEN LET PTYPE = 4      ; Set to red critical
137                    END
138   0058          IF ILOOP = 0 | ILOOP = 1 THEN LET PSKID = VLVPIT
139   0060          IF ILOOP = 2 | ILOOP = 3 THEN LET PSKID = HVAC
140   0062          IF ILOOP > 3 & ILOOP < 10 THEN LET PSKID = SERPIT
141   0064          IF ILOOP = 10 |ILOOP = 11 THEN LET PSKID = HVAC
142   0066          IF ILOOP > 11 & ILOOP < 16 THEN LET PSKID = VLVPIT
143   0068          IF ILOOP = 16 | ILOOP = 17 THEN LET PSKID = HVAC
144   0070          IF ILOOP = 18 THEN LET PSKID = VLVPIT
145   0072          IF ILOOP > 18 THEN LET PSKID = FARM
146   0074          IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
147   0075    SKP:  STEP PSTAT BY 1
148   0076          STEP PDIN BY 1
149   0077          STEP PTYPE BY 1
150   0078          STEP PREV BY 1
151   0079          STEP PSKID BY 1
152   0080          STEP PPASS BY 1
153   0081          NEXT ILOOP
154         ;
155         ;       For White alarms on pump devices:
156         ;
157   0082          FOR ILOOP = 30 TO 33
158         ;       BLET PSTAT = 0
159   0083          LET PTYPE = 1
160   0084          LET PREV = 1
161   0086          IF ILOOP = 30 | ILOOP = 31 THEN LET PSKID = VLVPIT
162   0088          IF ILOOP = 32 | ILOOP = 33 THEN LET PSKID = HVAC
163   0089          LET PPASS = 1
164   0090          STEP PSTAT BY 1
165   0091          STEP PTYPE BY 1
```

```
166   0092          STEP PREV BY 1
167   0093          STEP PSKID BY 1
168   0094          STEP PPASS BY 1
169   0095          NEXT ILOOP
170           ;
171   0096          FX0 = 0
172   0097          FX1 = 0
173   0098          FX2 = 0
174   0099          FX3 = 0
175                 ;
176                 ;  Set the white alarms:
177                 ;
178   0100          IALM0 = 0
179   0101          IALM1 = 0
180   0102          IALM2 = 0
181   0103          IALM3 = 0
182   0104          IALM20 = 1
183   0105          IALM21 = 1
184   0106          IALM30 = 1
185   0107          IALM31 = 1
186   0108          IALM32 = 1
187   0109          IALM33 = 1
188           ;
189   0110          FPAS0 = 1
190   0111          FPAS1 = 1
191   0112          FPAS2 = 1
192   0113          FPAS3 = 1
193           ;
194           ;       For PSH-SN215 (#34) and PSH-SN216 (#35) signals in PCM0A:
195           ;
196   0114          FX34 = 0
197   0115          FX35 = 0
198           ;       INT34 = 0
199           ;       INT35 = 0
200   0116          IALM34 = 2
201   0117          IALM35 = 2
202   0118          SKID34 = SERPIT
203   0119          SKID35 = SERPIT
204   0120          FPAS34 = 0
205   0121          FPAS35 = 0
206           ;
207           ;       E N D   I N I T I A L I Z I N G
208           ;
209           ;       S E N D   P R O G R A M   S T A R T   M E S S A G E
210           ;
211   0122          ILOOP = 37
212   0123          FSTART = 1
213   0124          GOSUB CYCL
214   0125          TIMERON STMR2
215   0126          TIMERCLR STMR2
216   0127          ON ERROR GOSUB ERRSUB
217           ;
218           ;       B E G I N   P R O G R A M   L O O P :
219           ;
220           ;       PLEASE NOTE:    ALARM ACKNOWLEDGE DEVICES USED IN
```

```
221           ;                       GRAPHICS 320 THROUGH 343 START WITH 'AFARM' AND GO
222           ;                       TO DEVICE 'AXYZ'.  GRAPHICS GO FROM XYZFRM TO AZFARM.
223           ;                       THIS IS TRANSPARENT TO THE OPERATOR.
224           ;
225    0128   AGAIN:   MARK STEP "AGAIN",0,0
226           ;        IF !SELECT()THEN FNING = 0
227           ;           ELSE FNING = 1                    ; Flag 480 determines if selected micro for DOG1A.C
228           ;        WAITUNTIL SELECT() ¦ FABORT          ; Wait until micro selected or program aborted
229           ;        IF (FABORT) THEN GOTO TERM           ; Terminate program if abort flag is set
230           ;        IF FNING = 0 THEN FHOLD = 1
231    0130            IF STMR2 > 30 THEN FHOLD = 1
232    0131            WAIT 1
233           ;
234    0132            DEV0 = 0
235    0133            DEV1 = 0
236    0134            DEV2 = 0
237    0135            DEV3 = 0
238    0136            DEV4 = 0
239    0137            DEV5 = 0
240                    ;
241                    ;  If alarm device is alarmed and its ALMCT indicates alarms are ack'd:
242                    ;
243    0138            ASSIGN PDEV = AFARM                  ; DEV ACCIDENTALLY OFFSET BY 1 (RNK)
244    0139            ASSIGN PSN = FSN0
245    0140            ASSIGN PACK = FLEAK0
246    0141            FOR ILOOP = 0 TO 3
247    0142            PSN =.0
248    0143            IF (PDEV:DEV_TMP0 = 2) & AND(PDEV:DEV_ALMCT,32) THEN
249                       BEGIN
250    0144               PDEV:DEV_TMP0 = 1                 ; Put device in ack state
251    0145               LET PSN = 1                       ; Set flag for graphic to ack
252    0146               LET FLGACK = 1                    ; WATCH.C spawns ACKFLG.C, which resets flag
253    0147               LET ACK = 1                       ; Set flag for C program to ACK system wide
254    0148               LET PACK = 1                ; ACKFLG.C to acknowledge this graphic
255    0149               ASSIGN PMSG0 = LEAK0              ; Make the appropriate lamp steady:
256    0150               STEP PMSG0 BY ILOOP
257    0152               IF ILOOP !=1 THEN LAMP 0,PMSG0,1  ; Flash lamp
258                       END
259    0153            ELSE PSN = 0
260    0154            STEP PSN BY 1
261    0155            STEP PDEV BY 1
262    0156            STEP PACK BY 1
263    0157            NEXT ILOOP
264           ;
265           ;        Field Alarm Silence feature:
266           ;
267    0158            IF(ACK)THEN
268                      BEGIN
269    0159              TIMErCLR STMR1
270    0160              TIMErON STMR1
271    0161              FLDALM = 1
272                      END
273    0163            IF(FLDALM)THEN IF STMR1 >= 30 THEN   ; If field alarm then ack in 30 sec
274                      BEGIN
275    0164              TIMErOFF STMR1
```

```
276  0165            TIMErCLR STMR1
277  0166            FLDALM = 0
278  0167            OPEN 'HSAXYZSIL'                    ; Silence alarm from PCM
279                  END
280                ;
281                ;  If field alarm ack then set flag and start timer:
282                ;
283  0169          IF(RESET)THEN IF LMPVAR < 2 THEN
284                  BEGIN
285  0170            FLDALM = 1
286  0171            TIMErON STMR1
287  0172            TIMErCLR STMR1
288  0173            RESET = 0
289                  END
290  0175          IF LMPVAR = 2 THEN RESET = 1
291          ;
292          ;       BEGIN DINs:
293          ;
294  0176          ASSIGN PDIN = 'PSH-SL100'            ; Assign pointer to first DIN
295  0177          ASSIGN PTYPE = IALM4
296  0178          ASSIGN PREV = FX4
297  0179          ASSIGN PSTAT = INT4
298  0180          ASSIGN PSKID = SKID4
299  0181          ASSIGN PPASS = FPAS4
300  0182          CURENT = 0                           ; Used to set lamp status
301  0183          NXT = 0                              ; Temporary INT
302  0184          BITVAL = 0
303          ;
304  0185    LOP:  FOR ILOOP = 5 TO 30
305  0187          IF ILOOP = 30 THEN GOTO SKP1                          ; -mct2+
306  0188          IF PTYPE THEN                                         ; If some alarm
307                   BEGIN                                      ;be1
308  0190             IF AND(PDIN:DIN_ALMCT,16) THEN LET PSTAT = 3   ; If EPN is inha
309  0192             IF AND(PDIN:DIN_ALMCT,4) THEN LET PSTAT = 5          ; If EPN is ibad
310  0194             IF !AND(PDIN:DIN_FLG,8) THEN LET PSTAT = 4           ; If EPN is dis
311  0197             IF ! PSTAT THEN IF PREV = PDIN:DIN_VAL THEN GOSUB ALM  ; If not prev alarmed
312  0198             IF PSTAT THEN
313                      BEGIN                                  ;be2
314  0199                IF PSTAT < 3 THEN                                ; If not inha
315                          BEGIN                              ;be3
316  0200                    IF PSTAT = 1 THEN
317                              BEGIN                          ;be4
318  0201                        IF AND(PDIN:DIN_ALMCT,32) THEN           ; If alarm is ack
319                                  BEGIN                      ;be5
320  0202                            LET PSTAT = 2
321  0203                            FHOLD = 1
322                                  END                        ;en5
323  0204                        IF ACK THEN                              ; If a graphic has been ack
324                                  BEGIN                      ;be6  -   go find which graphic
325  0205                            CRIT
326  0206                            ASSIGN PSN = FSNO                    ; Routine to find graphic
327  0207                            STEP PSN BY PSKID
328  0208                            IF PSN THEN                          ; If this is the graphic
329                                      BEGIN                  ;be7  -   then ack it
330  0209                                LET PSTAT = 2
```

```
331  0211                          IF PTYPE = 1 THEN PDIN:DIN_ALMCT = OR(PDIN:DIN_ALMCT,32)
332  0212                          FPASS = 1
333  0213                          GOSUB ACKK
334                                END                                    ;en7
335                              END                                      ;en6
336                            END                                        ;en4
337  0215                      IF PREV != PDIN:DIN_VAL THEN GOSUB CLR      ; If alarm has cleared
338  0216                      ASSIGN PMSG0 = DEV0                         ; Find active alarms & graphic:
339  0217                      STEP PMSG0 BY PSKID
340  0218                      IF PPASS THEN
341                               BEGIN
342  0219                         FHOLD = 1
343  0221                         IF PSTAT = 1 THEN PMSG0 = OR(PMSG0,PSTAT)
344                               END
345  0222                      ELSE PMSG0 = OR(PMSG0,PSTAT)
346                          END                                           ;en3
347  0224                ELSE IF !AND(PDIN:DIN_ALMCT,16)&!AND(PDIN:DIN_ALMCT,4)&AND(PDIN:DIN_FLG,8)THEN PSTAT = 0
348                       END                                           ;en2
349                   END                                               ;en1
350  0226           ELSE IF PSTAT THEN PSTAT = 0
351  0227  SKP1:    STEP PDIN BY 1
352  0228           STEP PTYPE BY 1
353  0229           STEP PREV BY 1
354  0230           STEP PSTAT BY 1
355  0231           STEP PSKID BY 1
356  0232           STEP PPASS BY 1
357  0233           NEXT ILOOP
358          ;
359          ;       END DINs
360          ;
361  0234           DEV6 = 0
362  0235           DEV7 = 0
363  0236           DEV8 = 0
364  0237           DEV9 = 0
365  0238           DEV10 = 0
366  0239           DEV11 = 0
367          ;
368          ;       BEGIN WHITE ALARMS FOR PUMP DEVICES:
369          ;
370  0240  AUX:     ASSIGN PEPN = 'P-AY-101'
371  0241           FOR ILOOP = 31 TO 34
372  0242           IF PTYPE THEN
373                    BEGIN
374  0244              IF AND(PEPN:DEV_ALMCT,16) THEN LET PSTAT = 3   ; If EPN is inha
375  0246              IF AND(PEPN:DEV_ALMCT,4) THEN LET PSTAT = 5           ; If EPN is ibad
376  0248              IF !AND(PEPN:DEV_FLG,8) THEN LET PSTAT = 4            ; If EPN is dis
377  0249              IF PSTAT < 3 THEN                                     ; If not inha
378                       BEGIN
379  0250                 IF (PSTAT = 1) & AND(PEPN:DEV_ALMCT,32) THEN       ; If prev unack alrarm, and is now ack
380                          BEGIN
381  0251                    LET PSTAT = 2
382  0252                    FHOLD = 1
383                          END
384  0253                 IF ACK THEN
385                          BEGIN
```

```
386  0254                    ASSIGN PSN = FSN0
387  0255                    STEP PSN BY PSKID
388  0256                    IF PSN & PSTAT = 1 THEN
389                             BEGIN
390  0257                       LET PSTAT = 2
391  0259                       IF PTYPE = 1 THEN PEPN:DEV_ALMCT = OR(PEPN:DEV_ALMCT,32)
392  0260                       FPASS = 1
393  0261                       GOSUB ACKK
394                             END
395                          END
396  0262                 IF !PSTAT THEN                                  ; If prev not alarmed
397                          BEGIN
398  0263                    IF AND(PEPN:DEV_STAT,16)THEN                 ; If shutdown
399                             BEGIN
400  0264                       LET PSTAT = 1
401  0266                       IF AND(PEPN:DEV_ALMCT,32)THEN PEPN:DEV_ALMCT = XOR(PEPN:DEV_ALMCT,32)
402  0267                       ASSIGN PDEV = AFARM
403  0268                       STEP PDEV BY PSKID
404  0269                       PDEV:DEV_TMP0 = 2
405  0270                       GOSUB CYCL
406                             END
407                          END
408  0271                 ELSE IF !AND(PEPN:DEV_STAT,16)THEN
409                             BEGIN
410  0272                       LET PSTAT = 0
411  0273                       FHOLD = 1
412  0275                       IF AND(PEPN:DEV_ALMCT,32)THEN PEPN:DEV_ALMCT = XOR(PEPN:DEV_ALMCT,32)
413                             END
414  0276                    IF PSTAT THEN
415                             BEGIN
416  0277                       ASSIGN PMSG0 = DEV6
417  0278                       STEP PMSG0 BY PSKID
418  0279                       IF PPASS THEN
419                                BEGIN
420  0281                          IF PSTAT = 1 THEN PMSG0 = OR(PMSG0,PSTAT)
421                                END
422  0282                       ELSE PMSG0 = OR(PMSG0,PSTAT)
423                             END
424                          END
425  0284                 ELSE IF !AND(PEPN:DEV_ALMCT,16)&!AND(PEPN:DEV_ALMCT,4)&AND(PEPN:DEV_FLG,8)THEN PSTAT = 0
426                       END
427  0285              STEP PEPN BY 1
428  0286              STEP PTYPE BY 1
429  0287              STEP PREV BY 1
430  0288              STEP PSTAT BY 1
431  0289              STEP PSKID BY 1
432  0290              STEP PPASS BY 1
433  0291              NEXT ILOOP
434            ;
435            ;       END WHITE ALARMS FOR PUMP DEVICES
436            ;
437            ;       BEGIN PCM0A ALARMS:
438            ;
439  0292      PCM0A:  ILOOP = 35
440  0293              IF IALM34 THEN
```

```
441                     BEGIN                                                   ;be1
442    0295             IF AND('PSH-SN215':DIN_ALMCT,16) THEN LET INT34 = 3             ; If EPN is inha
443    0297             IF AND('PSH-SN215':DIN_ALMCT,4) THEN LET INT34 = 5              ; If EPN is ibad
444    0299             IF !AND('PSH-SN215':DIN_FLG,8) THEN LET INT34 = 4               ; If EPN is dis
445    0302             IF ! INT34 THEN IF FX34 = 'PSH-SN215':DIN_VAL THEN GOSUB ALM
446    0303             IF INT34 THEN
447                        BEGIN                                                ;be2
448    0304                IF INT34 < 3 THEN                                             ; If not inha
449                           BEGIN                                             ;be3
450    0305.                  IF INT34 = 1 THEN
451                              BEGIN                                          ;be4
452    0306                         IF FSN1 THEN
453                                    BEGIN                                    ;be5
454    0307                            LET INT34 = 2
455    0309                            IF IALM34 = 1 THEN 'PSH-SN215':DIN_ALMCT = OR('PSH-SN215':DIN_ALMCT,32)
456    0310                            FPASS = 1
457    0311                            GOSUB ACKK
458                                    END                                      ;en5
459                               END                                           ;en4
460    0313                       IF FX34 != 'PSH-SN215':DIN_VAL THEN GOSUB CLR
461    0314                       IF FPAS34 THEN
462                                  BEGIN
463    0316                          IF INT34 = 1 THEN DEV1 = OR(DEV1,INT34)
464                                  END
465    0317                       ELSE DEV1 = OR(DEV1,INT34)
466                           END                                               ;en3
467    0319                ELSE IF !AND('PSH-SN215':DIN_ALMCT,16)&!AND('PSH-SN215':DIN_ALMCT,4)&AND('PSH-SN215':DIN_FLG,8)THEN INT34 = 0
468                        END                                                  ;en2
469                  END                                                        ;en1
470    0321        ELSE IF INT34 THEN INT34 = 0
471    0322        STEP PTYPE BY 1
472    0323        STEP PREV BY 1
473    0324        STEP PSTAT BY 1
474    0325        STEP PSKID BY 1
475    0326        STEP PPASS BY 1
476                ;
477    0327        ILOOP = 36
478    0328        IF IALM35 THEN
479                   BEGIN                                                   ;be1
480    0330           IF AND('PSH-SN216':DIN_ALMCT,16) THEN LET INT35 = 3             ; If EPN is inha
481    0332           IF AND('PSH-SN216':DIN_ALMCT,4) THEN LET INT35 = 5              ; If EPN is ibad
482    0334           IF !AND('PSH-SN216':DIN_FLG,8) THEN LET INT35 = 4               ; If EPN is dis
483    0337           IF ! INT35 THEN IF FX35 = 'PSH-SN216':DIN_VAL THEN GOSUB ALM
484    0338           IF INT35 THEN
485                      BEGIN                                                ;be2
486.   0339               IF INT35 < 3 THEN                                             ; If not inha
487                          BEGIN                                             ;be3
488    0340                  IF INT35 = 1 THEN
489                             BEGIN                                          ;be4
490    0341                        IF FSN1 THEN
491                                    BEGIN                                    ;be5
492    0342                            LET INT35 = 2
493    0344                            IF IALM35 = 1 THEN 'PSH-SN216':DIN_ALMCT = OR('PSH-SN216':DIN_ALMCT,32)
494    0345                            FPASS = 1
495    0346                            GOSUB ACKK
```

```
496                                     END                                          ;en5
497                                  END                                             ;en4
498     0348                         IF FX35 != 'PSH-SN216':DIN_VAL THEN GOSUB CLR
499     0349                         IF FPAS35 THEN
500                                     BEGIN
501     0351                            IF INT35 = 1 THEN DEV1 = OR(DEV1,INT35)
502                                     END
503     0352                         ELSE DEV1 = OR(DEV1,INT35)
504                                  END                                             ;en3
505     0354                      ELSE IF !AND('PSH-SN216':DIN_ALMCT,16)&!AND('PSH-SN216':DIN_ALMCT,4)&AND('PSH-SN216':DIN_FLG,8)THEN INT35 = 0
506                               END                                                ;en2
507                            END                                                   ;en1
508     0356                ELSE IF INT35 THEN INT35 = 0
509             ;
510             ;           END PCM0A ALARMS
511             ;
512     0357    S2ND:       ACK = 0
513     0358                SORT = 0
514             ;
515     0359                DEV12 = OR(DEV0,DEV6)
516     0360                DEV13 = OR(DEV1,DEV7)
517     0361                DEV14 = OR(DEV2,DEV8)
518     0362                DEV15 = OR(DEV3,DEV9)
519     0363                DEV16 = OR(DEV4,DEV10)
520     0364                DEV17 = OR(DEV5,DEV11)
521             ;
522     0365                ASSIGN PMSG0 = DEV12
523     0366                ASSIGN PMSG1 = FARM
524     0367                FOR LLOOP = 1 TO 6
525     0368                IF PMSG0 THEN
526                           BEGIN
527     0370                  IF AND(PMSG0,1) THEN PMSG1 = 2
528     0371                  ELSE PMSG1 = 1
529                           END
530     0372                ELSE PMSG1 = 0
531     0373                STEP PMSG0 BY 1
532     0374                STEP PMSG1 BY 1
533     0375                NEXT LLOOP
534             ;
535     0376                IF FHOLD THEN
536                            BEGIN
537     0377                   FHOLD = 0
538     0378                   TIMERCLR STMR2
539     0379                   TIMERON STMR2
540     0380                   LAMP 0,15,FARM
541     0381                   LAMP 0,14,SERPIT
542     0382                   LAMP 0,13,VLVPIT
543     0383                   LAMP 0,12,HVAC
544                            END
545     0384                MARK STEP "BOTTOM",0,0
546     0385                NOCRIT
547             ;
548             ;           BEGIN 204-AR Alarms:  Relate to LED 31 (mct)
549             ;
550                         ;  If QS-204AR1 alarm and unack:
```

```
551                     ;
552     0386            IF AND('QS-204AR1':DIN_ALMCT,2) & !AND('QS-204AR1':DIN_ALMCT,32) THEN
553                             BEGIN
554     0387                    FSL6 = 1                                        ; Alarm not ack
555     0388                    LAMP 0,23,2                                     ; Blink lamp
556                             END
557     0389            IF AND('QS-204AR1':DIN_ALMCT,32) &  FSL6=1 THEN         ; If ACK & 1st pass
558                             BEGIN
559     0390                    FSL6 = 0                                        ; Alarm is ack
560     0392                    IF FSL7=0 THEN LAMP 0,23,1                      ; Lamp steady on
561                             END
562     0394            IF !AND('QS-204AR1':DIN_ALMCT,2) & FSL6=1 & AND('QS-204AR2':DIN_ALMCT,2) THEN LAMP 0,23,1
563     0396            IF !AND('QS-204AR1':DIN_ALMCT,2) THEN FSL6 = 0          ; If no alarm
564                     ;
565                     ;  If QS-204AR2 alarm and unack:
566                     ;
567     0397            IF AND('QS-204AR2':DIN_ALMCT,2) & !AND('QS-204AR2':DIN_ALMCT,32) THEN
568                             BEGIN
569     0398                    FSL7 = 1                                        ; Alarm not ack
570     0399                    LAMP 0,23,2                                     ; Blink lamp
571                             END
572     0400            IF AND('QS-204AR2':DIN_ALMCT,32) & FSL7=1 THEN          ; IF ACK
573                             BEGIN
574     0401                    FSL7 = 0                                        ; Alarm is ack
575     0403                    IF FSL6=0 THEN LAMP 0,23,1                      ; Lamp steady on
576                             END
577     0405            IF !AND('QS-204AR2':DIN_ALMCT,2) & FSL7=1 & AND('QS-204AR1':DIN_ALMCT,2) THEN LAMP 0,23,1
578     0407            IF !AND('QS-204AR2':DIN_ALMCT,2) THEN FSL7 = 0          ; If no alarm
579                     ;
580                     ;  If No Alarms:
581                     ;
582     0409            IF !AND('QS-204AR1':DIN_ALMCT,2) & !AND('QS-204AR2':DIN_ALMCT,2) THEN LAMP 0,23,0
583             ;
584             ;       END 204-AR Alarms (mct)
585             ;
586     0410            NOCRIT
587     0411            GOTO AGAIN
588             ;
589             ;       E N D   P R O G R A M   L O O P
590             ;
591             ;       BEGIN CLEAR, ALARM, CYCLE, ACKNOWLEDGE, MESSAGE, ERROR, AND TERMINATE SUBROUTINES:
592             ;
593     0412    CLR:    LET PSTAT = 0
594     0414            IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
595     0415            FHOLD = 1
596     0416            RETURN
597             ;
598     0417    ALM:    LET PSTAT = 1
599     0419            IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
600     0420            ASSIGN PDEV = AFARM
601     0421            STEP PDEV BY PSKID
602     0422            PDEV:DEV_TMP0 = 2
603     0423            CRIT
604     0424            ASSIGN PMSG0 = LEAK0
605     0425            STEP PMSG0 BY PSKID
```

```
606   0426            LAMP 0,PMSG0,2
607   0428            IF AND(PTYPE,3)THEN FALMON = 1
608           ;
609   0429    CYCL:   FHOLD = 1
610   0430            LSG4 = LSG3
611   0431            LSG3 = LSG2
612   0432            LSG2 = LSG1
613   0433            LSG1 = LSG0
614   0434            COLOR4 = COLOR3
615   0435            COLOR3 = COLOR2
616   0436            COLOR2 = COLOR1
617   0437            COLOR1 = COLOR0
618   0438            EPNS4 = EPNS3
619   0439            EPNS3 = EPNS2
620   0440            EPNS2 = EPNS1
621   0441            EPNS1 = STRNG1
622   0442            GRAFX4 = GRAFX3
623   0443            GRAFX3 = GRAFX2
624   0444            GRAFX2 = GRAFX1
625   0445            GRAFX1 = GRAFX0
626   0446            SDATE4 = SDATE3
627   0447            SDATE3 = SDATE2
628   0448            SDATE2 = SDATE1
629   0449            SDATE1 = SDATE0
630           ;
631           ;       ALARM TYPER STRINGS:
632           ;
633   0450    ACKK:   ON ILOOP GOTO S1,S2,S3,S4,S5,S6,S7,S8,S9,S10
634   0451            ON ILOOP - 10 GOTO S11,S12,S13,S14,S15,S16,S17,S18,S19,S20
635   0452            ON ILOOP - 20 GOTO S21,S22,S23,S24,S25,S26,S27,S28,S29,S30
636   0453            ON ILOOP - 30 GOTO S31,S32,S33,S34,S35,S36,S37,S38,S39
637   0454            GOTO AMSG
638   0455    S1:     STRNG0 = "PUMP AY-101 MCC OFF   "
639   0456            STRNG1 ="YS-AY-101"
640   0457            GOTO AMSG
641   0458    S2:     STRNG0 = "PUMP AY-102 MCC OFF   "
642   0459            STRNG1 = "YS-AY-102"
643   0460            GOTO AMSG
644   0461    S3:     STRNG0 = "PUMP AZ-101 MCC OFF   "
645   0462            STRNG1 = "YS-AZ-101"
646   0463            GOTO AMSG
647   0464    S4:     STRNG0 = "PUMP AZ-102 MCC OFF   "
648   0465            STRNG1 = "YS-AZ-102"
649   0466            GOTO AMSG
650   0467    S5:     STRNG0 = "AX-B SL-100 LINE LEAK "
651   0468            STRNG1 = "PSH-SL100"
652   0469            GOTO AMSG
653   0470    S6:     STRNG0 = "AX-A SL-101 LINE LEAK "
654   0471            STRNG1 = "PSH-SL101"
655   0472            GOTO AMSG
656   0473    S7:     STRNG0 = "A-B SL-113 LINE LEAK  "
657   0474            STRNG1 = "PSH-SL113"
658   0475            GOTO AMSG
659   0476    S8:     STRNG0 = "A-A SL-114 LINE LEAK  "
660   0477            STRNG1 = "PSH-SL114"
```

```
661   0478          GOTO AMSG
662   0479   S9:    STRNG0 = "AX-B SN-200/213 LEAK  "
663   0480          STRNG1 = "PSH200213"
664   0481          GOTO AMSG
665   0482   S10:   STRNG0 = "AX-A SN-201/214 LEAK  "
666   0483          STRNG1 = "PSH201214"
667   0484          GOTO AMSG
668   0485   S11:   STRNG0 = "AZ SL-500 LINE LEAK   "
669   0486          STRNG1 = "LDS-SL500"
670   0487          GOTO AMSG
671   0488   S12:   STRNG0 = "AZ SL-501 LINE LEAK   "
672   0489          STRNG1 = "LDS-SL501"
673   0490          GOTO AMSG
674   0491   S13:   STRNG0 = "AY SL-502 LINE LEAK   "
675   0492          STRNG1 = "LDS-SL502"
676   0493          GOTO AMSG
677   0494   S14:   STRNG0 = "AY SL-503 LINE LEAK   "
678   0495          STRNG1 = "LDS-SL503"
679   0496          GOTO AMSG
680   0497   S15:   STRNG0 = "AY SL-504 LINE LEAK   "
681   0498          STRNG1 = "LDS-SL504"
682   0499          GOTO AMSG
683   0500   S16:   STRNG0 = "AY SL-505 LINE LEAK   "
684   0501          STRNG1 = "LDS-SL505"
685   0502          GOTO AMSG
686   0503   S17:   STRNG0 = "AZ SN-600 LINE LEAK   "
687   0504          STRNG1 = "LDS-SN600"
688   0505          GOTO AMSG
689   0506   S18:   STRNG0 = "AZ SN-601 LINE LEAK   "
690   0507          STRNG1 = "LDS-SN601"
691   0508          GOTO AMSG
692   0509   S19:   STRNG0 = "AY DR-700 LINE LEAK   "
693   0510          STRNG1 = "LDS-DR700"
694   0511          GOTO AMSG
695   0512   S20:   STRNG0 = "AX-A/B SELPOT HI FAIL "
696   0513          STRNG1 = "LXS-AXPOT"
697   0514          GOTO AMSG
698   0515   S21:   STRNG0 = "AX-A/B SEAL POT HIGH  "
699   0516          STRNG1 = "LSH-AX-1"
700   0517          GOTO AMSG
701   0518   S22:   STRNG0 = "AX-A/B SEAL POT LOW   "
702   0519          STRNG1 = "LSL-AX-1"
703   0520          GOTO AMSG
704   0521   S23:   STRNG0 = "A/AX COB LEAK         "
705   0522          STRNG1 = "LDSCOBAAX"
706   0523          GOTO AMSG
707   0524   S24:   STRNG0 = "A/AX AREA RADN BYPASS "
708   0525          STRNG1 = "YS-AAXARM"
709   0526          GOTO AMSG
710   0527   S25:   STRNG0 = "A/AX MASTER SHUTDOWN  "
711   0528          STRNG1 = "YS-AAXMSD"
712   0529          GOTO AMSG
713   0530   S26:   STRNG0 = "A/AX FARM RADN HIGH   "
714   0531          STRNG1 = "RSHAAX3/4"
715   0532          GOTO AMSG
```

```
716   0533   S27:    STRNG0 = "A/AX/AY/AZ RD HI/FAIL "
717   0534           STRNG1 = "RSH-AXZY1"
718   0535           GOTO AMSG
719   0536   S28:    STRNG0 = "A/AX/AZ PPT AZ COB LK "
720   0537           STRNG1 = "LDSPPCOB1"
721   0538           GOTO AMSG
722   0539   S29:    STRNG0 = "A3/AX3/AX4 RAD MON FA "
723   0540           STRNG1 = "RXSAAX3/4"
724   0541           GOTO AMSG
725          S30:    ;       STRNG0 = "LEAK-DET CNSL GEN ALM "       -mct2+
726          ;       STRNG1 = "QSCONSOLE"
727          ;       GOTO AMSG
728   0542   S31:    STRNG0 = "PUMP AY-101 SHUTDOWN  "
729   0543           STRNG1 = "P-AY-101"
730   0544           GOTO AMSG
731   0545   S32:    STRNG0 = "PUMP AY-102 SHUTDOWN  "
732   0546           STRNG1 = "P-AY-102"
733   0547           GOTO AMSG
734   0548   S33:    STRNG0 = "PUMP AZ-101 SHUTDOWN  "
735   0549           STRNG1 = "P-AZ-101"
736   0550           GOTO AMSG
737   0551   S34:    STRNG0 = "PUMP AZ-102 SHUTDOWN  "
738   0552           STRNG1 = "P-AZ-102"
739   0553           GOTO AMSG
740   0554   S35:    STRNG0 = "A SN-215 LINE LEAK    "
741   0555           STRNG1 = "PSH-SN215"
742   0556           GOTO AMSG
743   0557   S36:    STRNG0 = "A SN-216 LINE LEAK    "
744   0558           STRNG1 = "PSH-SN216"
745   0559           GOTO AMSG
746   0560   S37:    STRNG0 = "XYZBAT PROGRAM START  "
747   0561           GOTO AMSG
748   0562   S38:    STRNG0 = "XYZBAT IN WRONG UNIT  "
749   0563           GOTO AMSG
750   0564   S39:    STRNG0 = "XYZBAT TERMINATING    "
751          ;
752   0565   AMSG:   IF (FPASS)THEN
753                     BEGIN
754   0566              PRINT ADI,"%p ALARM ACKNOWLEDGE",STRNG1
755   0567              FPASS = 0
756   0568              GOTO NXT
757                     END
758   0570            IF PTYPE = 4 THEN COLOR0 = 3
759   0571               ELSE COLOR0 = PTYPE
760   0572            EPNS0 = STRNG1
761   0573            IN0 = TIMSTR(SSNG0)
762   0575            IF IN0 != 8 THEN SSNG0 = "TIMEFAIL"
763   0576            IN0 = APPEND(LSG0,STRNG0,SSNG0)
764   0578            IF IN0 < 0 THEN LSG0 = "APPEND FAILED!!"
765   0579            IN0 = DATSTR(SDATE0)
766   0581            IF IN0 != 9 THEN SDATE0 = "DATE FAIL"
767   0583            IF PSKID = 1 THEN GRAFX0 = 272
768   0585            IF PSKID = 2 THEN GRAFX0 = 272
769   0587            IF PSKID = 3 THEN GRAFX0 = 273
770   0589            IF PSKID = 4 THEN GRAFX0 = 274
```

```
771   0591            IF PSKID = 5 THEN GRAFX0 = 270
772   0593            IF PSKID = 0 THEN GRAFX0 = 270
773   0595            IF PTYPE = 1 THEN PRINT ADI,"WHITE ALARM %p %p GRAPHICS %i",EPNS0,STRNG0,GRAFX0
774   0596  NXT:      IF FSTART = 1 THEN
775                      BEGIN
776   0597               COLOR0 = 0
777   0598               GRAFX0 = 0
778   0599               STRNG1 =" "
779   0600               FSTART = 0
780   0601               PRINT ADI,"%p",STRNG0
781                      END
782   0602          RETURN
783         ;
784   0603  ERRSUB:IF ERRN != ERRNUM() THEN
785                BEGIN
786   0604          LET ERRN = ERRNUM()
787   0605          LET ERRL = ERRLIN()
788   0606          PRINT ADI,"ERROR # %i AT LINE %i, ILOOP = %i",errn,errl,ILOOP
789                END
790   0607         ERSCLR
791   0608         GOTO AGAIN
792         ;
793         ; TERM:  LET FABORT = 0
794   0609  TERM:    LET STRNG0 = "XYZBAT TERMINATED    "
795   0610         LET ILOOP = 39
796   0611         LET FSTART = 1
797   0612         TIMErOFF STMR1
798   0613         TIMErCLR STMR1
799   0614         LET FLDALM = 0
800   0615         OPEN 'HSAXYZSIL'
801   0616         GOSUB CYCL
802                END

      Total Errors Detected      0     Total Warnings      0
           Sequence Program Size = 12032 Bytes, 752 Pages
                95 Forward Reference(s)
```

242BATCH

```
;       2 4 2 B A T C H . S E Q                                 VERSION 2
;
;       Function:       Sequence program for Lamp function,
;                       message display on Graphic number 023,
;                       and printer messages.
;
;       Written By      : R. N. Kersey
;       Date            : 05-31-92
;       Facility        : 242-A Evaporator Bldg.
;       Machine         : VAX 4104-A
;       Op. System      : VAX/VMS 5.5-2AH
;       D/3 Version     : 6.3-2 BASELINE
;       Compiler        : SIC
;       Build command   : @[320,20]242BATCH
;
;       UNIT = 5, SKDKEY = 47
;
;       Three AIs (RI-K1-1-1, RI-K1-1-2, and FI-K1-1) are found in unit ROOMS in PCM0A.
;
;       Assoc. UPF:     [300,1]_TNKFRM.___      TNKFRM.LST
;       Assoc. RPF:     [300,1]_FARM.___        FARM.LST
;       SDB:            [300,1]PCM1ASDB.___     PCM1ASDB.CMD
;
;       POINTER  SDB VAR.       FUNCTION
;       -------  ---------      ------------------------------------------
;       PSTAT    INTn           Current EPN alarm state is an integer tied to
;                               graphic flashing string of value:
;       .        (n=0-100)              0 = previously not alarmed
;       .                               1 = previously alarmed, not ack
;       .                               2 = previously alarmed, is ack
;       .                               3 = alarm is inhibited
;       .                               4 = input is off scan
;       .                               5 = input is bad
;       PDIN     'EPN'          ---
;       PTYPE    IALMn          alarm type 0 = none, 1 = white, 2 = yellow,
;                                  3 = yellow and yellow, 4 = red,
;                                  6 = yellow and red
;       PREV     FXn            Reverse acting (not fail safe) flag (1=set)
;       PSKID    SKIDn          Tracks which graphic has an alarm
;       PPASS    FPASn          If set (1), turn lamp off when alarm is ACKnowledged
;       PSN      FSNn           Flag for the graphic to ACKnowledge
;       ---      FHOLD          Flag to update the lamps
;       ---      FNING          Flag for DOG1A.C micro select
;       ---      ACK            Flag for C program to ACK system wide
;
; MODIFICATION LOG:
;
;       10-DEC-92       Modified STRNG0 length.                         M. Teats
;
;       15-DEC-92       Modified for Rad Systems Upgrade.               M. Teats
;
;       30-DEC-92       Fixed white alarm print message.        M. Teats
;
;       15-JUN-93       Added comments and baselined for        M. Teats
;                       D/3 version 3.3.  CCN# 93JUN01-01.
```

```
56          ;
57          ;       15-DEC-94       Revised header to indicate the new VMS          J. Berger
58          ;                       system. Rebaselined document. Removed
59          ;                       obsolite Select() statement.
60          ;
61          ;  END MOD LOG
62          ;
63          UNIT    BLDNG
64                  COMMON ROOMS
65          ;
66    0000          NOP
67    0001          LET STAR =  45                  ; Start message
68    0002          LET SKDKEY = 47         ; SKID Key
69    0003          LET UNITT = 5                   ; SEQ Unit number
70    0004          ASSIGN PDEV = 'BUILDING' ; Alarm device
71    0005          ASSIGN PTYPE = IALM0
72    0006          LET ADI = 0                     ; Alarm destination index
73    0007          LET FSN0 = 0                    ; No alarm yet
74    0008          TIMEROFF STMR0                  ; Alarm timer off
75    0009          TIMERCLR STMR0                  ; Alarm timer clear
76    0010          LET STRNG0 = " "                ; Blank 30 character string
77    0011          LET ERRN = 0                    ; Set error number to zero
78    0012          LET ERRL = 0                    ; Set error line to zero
79    0013          LET ACK = 0
80    0014          GETUNIT TRIP                    ; Which sequence unit is program running in
81    0015          IF TRIP != UNITT THEN           ; Abort if running in wrong seq. unit
82                    BEGIN
83    0016            FSTART = 1
84    0017            ILOOP = STAR + 1
85    0018            PTYPE = 0
86    0019            GOSUB ACKK
87    0020            GOTO TERM
88                    END
89          ;       IF !SELECT() THEN GOTO AGAIN
90          ;
91          ;       B E G I N   I N I T I A L I Z I N G :
92          ;
93          ;       0-27 are PCM1 DIN, 28-41 are PCM1 AIN, 42-44 are PCM0 AIN
94          ;
95    0021          LET SSNG0 = "INIT"
96    0022    INIT: LET FABORT = 0                  ; If FABORT = 1 then abort program
97    0023          LET FSTART = 0                  ; Program start flag = 0
98    0024          LAMP ADI,SKDKEY,0               ; Turn off the annunciator key
99    0025          LET LSG0 = " "                  ; Clear the string variables
100   0026          LET LSG1 = " "
101   0027          LET LSG2 = " "
102   0028          LET LSG3 = " "
103   0029          LET LSG4 = " "
104   0030          LET PDEV:DEV_TMP0 = 1                                   ; Set alarm device to non-alarm state
105   0031          IF AND(PDEV:DEV_ALMCT,16) THEN                          ; If alarm device is inha, remove the inha
106   0032             LET PDEV:DEV_ALMCT=XOR(PDEV:DEV_ALMCT,16)
107   0033          IF AND(PDEV:DEV_ALMCT,32) THEN                          ; If alarm device is ack, clear it
108   0034             LET PDEV:DEV_ALMCT = XOR(PDEV:DEV_ALMCT,32)
109         ;
110         ;       INITIALIZE PCM1 DINs AND AINs:
```

```
111             ;
112     0035            ASSIGN PDIN = 'RSH-ARAMU'
113     0036            ASSIGN PSTAT = INT0
114     0037            ASSIGN PTYPE = IALM0
115     0038            ASSIGN PREV = FX0
116     0039            ASSIGN PSKID = SKID0
117     0040            ASSIGN PPASS = FPAS0
118             ;
119     0041            FOR ILOOP = 0 TO 41                             ; 0-27 are PCM1 DINs, 28-41 are PCM1 AINs
120     0042            LET PPASS = 0
121     0043            LET PREV = 0
122     0045            IF PSTAT > 5 THEN PSTAT = 0                     ; Reset current status if undefined
123     0046            LET PSKID = 0
124     0047            LET PTYPE = 2                                   ; Default to yellow alarm
125     0048            IF ILOOP <= 27 THEN                             ; For PCM1 DINs
126                       BEGIN
127     0049              IF (AND(PDIN:DIN_DFLG,3)) THEN          ; Check for alarms (Must be red or yellow)
128                         BEGIN
129     0050                LET PTYPE = 2                               ; Default to yellow alarm
130     0052                IF AND(PDIN:DIN_DFLG,1) THEN LET PREV = 1           ; Reverse acting
131     0054                IF AND(PDIN:DIN_DFLG,4) THEN LET PTYPE = 4          ; Set to red critical
132                         END
133     0056              IF ILOOP < 27 THEN STEP PDIN BY 1             ; Point to next DIN
134                       END
135     0057            ELSE LET PTYPE = 4                              ; For PCM1 AINs
136     0058            STEP PTYPE BY 1
137     0059            STEP PREV BY 1
138     0060            STEP PSTAT BY 1
139     0061            STEP PSKID BY 1
140     0062            STEP PPASS BY 1
141     0063            NEXT ILOOP
142             ;
143             ;       DONE WITH PCM1 DINs AND AINs:
144             ;
145             ;       INITIALIZE PCM0 AINs:
146             ;
147     0064            LET IALM42 = 4          ; RI-K1-1-1
148     0065            LET SKID42 = 0
149     0066            LET IALM43 = 4          ; RI-K1-1-2
150     0067            LET SKID43 = 0
151     0068            LET IALM44 = 0          ; FI-K1-1
152     0069            LET SKID44 = 0
153             ;
154             ;       E N D   I N I T I A L I Z I N G
155             ;
156             ;       S E N D   P R O G R A M   S T A R T   M E S S A G E
157             ;
158     0070            LET PTYPE = 0
159     0071            LET ILOOP = STAR
160     0072            LET FSTART = 1
161     0073            GOSUB ACKK
162     0074            ON ERROR GOSUB ERRSUB
163             ;
164             ;       B E G I N   P R O G R A M   L O O P :
165             ;
```

```
166   0075   AGAIN:   MARK STEP "AGAIN",0,0
167          ;        IF (FABORT) THEN GOTO TERM       ; Terminate program if abort flag set
168   0076           SSNGO = "WAIT"
169          ; JH01   WAITUNTIL SELECT()               ; Wait until micro is selected
170   0077           WAIT 1
171   0078           IF (FSNO) THEN                    ; If alarm flag set
172   0079             IF STMRO > 60 THEN              ; If alarm flag set for 1 minute
173                        BEGIN                       ; Reset timer and turn off flag
174   0080                 TIMEROFF STMRO
175   0081                 TIMERCLR STMRO
176   0082                 FSNO = 0
177   0083                 ERRN = 0
178   0084                 ERRL = 0
179                        END
180          ;
181          ;        If BUILDING $T1 is alarmed and it's ALMCT indicates alarms are acknowledged:
182          ;
183   0085           IF AND(PDEV:DEV_TMPO,2) & AND(PDEV:DEV_ALMCT,32) THEN
184             BEGIN
185   0086        LET PDEV:DEV_TMPO = 1  ; Put BUILDING in ACK state
186   0087        LET FTRIPO = 1                  ; Set for Unit alarm acknowledgement, system to grahpic
187   0088        LET ACK = 1                     ; Set for PCMO Unit alarm acknowledgement, grahpic to system
188   0089        LET FLGACK = 1                  ; Set for PCM1 Unit alarm acknowledgement, grahpic to system
189   0090        CRIT
190   0091        LAMP ADI,SKDKEY,1               ; Turn lamp on
191             END
192          ;
193          ;        BEGIN PCM1 DINS:
194          ;
195   0092           ASSIGN PDIN = 'RSH-ARAMU'
196   0093           ASSIGN PTYPE = IALMO
197   0094           ASSIGN PREV = FXO
198   0095           ASSIGN PSTAT = INTO
199   0096           ASSIGN PPASS = FPASO
200   0097           LET CURENT = 0                  ; Used to set lamp status
201   0098           LET LAST = 0
202   0099           LET NXT = 0
203   0100           LET LMPVAR = 0
204   0101           LET BITVAL = 0
205          ;
206   0102   LOP:    FOR ILOOP = 0 TO 27
207          ;        IF !SELECT() THEN GOTO AGAIN
208   0103           IF ( PTYPE > 0 ) THEN                                    ; If some type of alarm
209                     BEGIN                                              ;be1
210   0105              IF AND(PDIN:DIN_ALMCT,16) THEN  LET PSTAT = 3  ; If EPN is inhibited
211   0107              IF AND(PDIN:DIN_ALMCT,4) THEN  LET PSTAT = 5   ; If EPN is bad input
212   0109              IF !AND(PDIN:DIN_FLG,8) THEN  LET PSTAT = 4            ; If EPN is not disabled
213   0110              IF ( PSTAT < 3 ) THEN                             ; If not inhibited
214                        BEGIN                                           ;be2
215   0111                 IF PSTAT = 1 THEN
216                           BEGIN                                        ;be3
217   0112                    IF PTYPE != 1 THEN
218   0114                       IF AND(PDIN:DIN_ALMCT,32) THEN GOSUB ACK1
219   0116                    IF (FTRIP4) THEN GOSUB ACK1
220                           END                                          ;en3
```

```
221  0119                IF PSTAT = 0 THEN IF(PDIN:DIN_VAL = PREV) THEN GOSUB ALM
222  0120                IF (PSTAT) THEN
223                         BEGIN                                          ;be4
224  0122                   IF PDIN:DIN_VAL != PREV THEN GOSUB CLR
225  0123                   NXT = PSTAT
226  0125                   IF (PPASS) & (PSTAT = 2 ) THEN NXT = 0
227  0126                   LAST = OR(LAST,NXT)
228                         END                                            ;en4
229                      END                                               ;en2
230  0128             ELSE IF !AND(PDIN:DIN_ALMCT,16)&!AND(PDIN:DIN_ALMCT,4)&AND(PDIN:DIN_FLG,8)THEN LET PSTAT = 0
231                END                                                     ;en1
232  0130          IF ILOOP < 27 THEN STEP PDIN BY 1
233  0131          STEP PTYPE BY 1
234  0132          STEP PREV BY 1
235  0133          STEP PSTAT BY 1
236  0134          STEP PPASS BY 1
237  0135          NEXT ILOOP
238          ;
239  0137          IF (INT5 + INT6) > 0 THEN LET FTRIP1 = 1  ; For K1-6-1 combined dp
240  0138            ELSE LET FTRIP1 = 0
241  0140          IF (INT7 + INT8) > 0 THEN LET FTRIP2 = 1  ; For K1-6-2 combined dp
242  0141            ELSE LET FTRIP2 = 0
243          ;
244          ;       END PCM1 DINS
245          ;
246          ;       BEGIN PCM1 ANALOGS:
247          ;
248  0142          ANALOG = 0
249  0143          SSNGO = "RUN"
250  0144          ASSIGN PSKID = SKID28
251  0145          ASSIGN PTYPE = IALM28
252  0146          ASSIGN PAI = 'RI-AR-AMU'
253  0147          FOR ILOOP = 28 TO 41
254          ;       IF !SELECT() THEN GOTO AGAIN
255  0148              IF(PTYPE) THEN
256                     BEGIN                                              ;be1
257  0149               BITVAL = PAI:AI_ALMCT
258  0150               IF !AND(BITVAL,16) THEN
259                       BEGIN                                            ;be2
260  0151                 IF AND(BITVAL,15) THEN
261                         BEGIN                                          ;be3
262  0153                   IF AND(BITVAL,32) THEN NXT = 2
263                         ELSE
264                           BEGIN                                        ;be4
265  0154                     IF PSKID != BITVAL THEN
266                             BEGIN                                      ;be5
267  0155                       LET PSKID = BITVAL
268  0157                       IF BITVAL < 3 THEN FALMON = 1
269  0158                       PDEV:DEV_TMP0 = 2
270  0159                       NXT = 1
271                             END                                        ;en5
272  0160                     ELSE NXT = 1
273                           END                                          ;en4
274                       END                                              ;en3
275                   ELSE
```

```
276                                 BEGIN
277    0161                         NXT = 0
278    0162                         LET PSKID = 0
279                                 END
280                              END                                        ;en2
281                           ELSE
282                              BEGIN
283    0163                      NXT = 0
284    0164                      LET PSKID = 0
285                              END
286    0165                      ANALOG = OR(ANALOG,NXT)
287                           END                                           ;en1
288    0166             STEP PSKID BY 1
289    0167             STEP PTYPE BY 1
290    0169             IF ILOOP < 41 THEN STEP PAI BY 1
291    0170             NEXT ILOOP
292             ;
293             ;       END PCM1 ANALOGS
294             ;
295             ;       BEGIN PCM0 ANALOGS:
296             ;
297    0171             ILOOP = 42
298    0172             ASSIGN PAI = 'RI-K1-1-1'          ; 42
299    0173             GOSUB ANLG1
300    0174             ASSIGN PAI = 'RI-K1-1-2'          ; 43
301    0175             GOSUB ANLG1
302    0176             ASSIGN PAI = 'FI-K1-1'                    ; 44
303    0177             GOSUB ANLG1
304             ;
305             ;       END PCM0 ANALOGS
306             ;
307    0178             NOCRIT
308    0180             IF LAST > 3 THEN PRINT ADI," LAMP VARIABLE IS GREATER THAN 3!"
309    0181             CURENT = OR(LAST,ANALOG)
310    0183             IF AND(CURENT,1) THEN LMPVAR = 2
311    0185               ELSE IF AND(CURENT,2) THEN LMPVAR = 1
312    0186                       ELSE LMPVAR = 0
313    0187             LAMP ADI,SKDKEY,LMPVAR
314    0188             MARK STEP "BOTTOM",0,0
315    0190             IF AND(BUILDING:DEV_FLG,64) THEN PUTMANL BUILDING
316    0191             LET ACK = 0
317    0192             GOTO AGAIN
318             ;
319             ;       E N D   P R O G R A M   L O O P
320             ;
321             ;       BEGIN PCM0 ANALOG SUBROUTINE:
322             ;
323    0193     ANLG1:  BITVAL = PAI:AI_ALMCT
324             ;       IF !SELECT() THEN GOTO AGAIN
325    0194             IF !AND(BITVAL,16) THEN
326                        BEGIN                                          ;be2
327    0195                IF AND(BITVAL,15) THEN
328                           BEGIN                                       ;be3
329    0197                   IF AND(BITVAL,32) THEN NXT = 2
330                           ELSE
```

```
331                              BEGIN                              ;be4
332    0198                      IF PSKID != BITVAL THEN
333                                 BEGIN                           ;be5
334    0199                         LET PSKID = BITVAL
335    0200                         LET PDEV:DEV_TMP0 = 2
336    0201                         NXT = 1
337                                 END                             ;en5
338    0202                      ELSE NXT = 1
339                              END                                ;en4
340                           END                                   ;en3
341                        ELSE
342                           BEGIN
343    0203                   NXT = 0
344    0204                   LET PSKID = 0
345                           END
346                        END                                      ;en2
347                     ELSE
348                        BEGIN
349    0205                NXT = 0
350    0206                LET PSKID = 0
351                        END
352    0207               ANALOG = OR(ANALOG,NXT)
353    0208               STEP PSKID BY 1
354    0209               STEP PTYPE BY 1
355    0210               ILOOP = ILOOP + 1
356    0211               RETURN
357             ;
358             ;         END PCM0 ANALOG SUBROUTINE
359             ;
360             ;         BEGIN CLEAR, ALARM, ACKNOWLEDGE, MESSAGE, ERROR, AND TERMINATE SUBROUTINES:
361             ;
362    0212     CLR:      LET PSTAT = 0
363    0215               IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
364    0216               LET FCLEAR = 1
365    0217               GOSUB ACKK
366    0218               RETURN
367             ;
368    0219     ALM:      LET PSTAT = 1
369    0222               IF PTYPE != 1 THEN IF AND(PDIN:DIN_ALMCT,32)THEN PDIN:DIN_ALMCT = XOR(PDIN:DIN_ALMCT,32)
370    0224               IF AND(PTYPE,3)THEN FALMON = 1
371    0225               LET PDEV:DEV_TMP0 = 2
372    0226               LAMP ADI,SKDKEY,2
373    0227               CRIT
374    0228               GOSUB ACKK
375    0229               RETURN
376             ;
377    0230     ACK1:     LET PSTAT = 2
378    0231               LET FACK = 1
379    0232               LAMP ADI,SKDKEY,1
380             ;
381             ;         ALARM TYPER STRINGS:
382             ;
383    0234     ACKK:     IF ILOOP = 0  THEN  STRNG0 = "AMU AREA RAD HI      "
384    0236               IF ILOOP = 1  THEN  STRNG0 = "AMU AIRSMP CAM RAD HI"
385    0238              IF ILOOP = 2  THEN  STRNG0 = "242A K1 ALPHA RAD HI "
```

```
386  0240            IF ILOOP = 3  THEN  STRNGO = "242A K1 B/GAMA RAD HI"
387  0242            IF ILOOP = 4  THEN  STRNGO = "242A K1 RAD MON TROUB"
388  0244            IF ILOOP = 5  THEN  STRNGO = "242A K1-6-1 FLT DP HI"
389  0246            IF ILOOP = 6  THEN  STRNGO = "242A K1-6-1 FLT DP LO"
390  0248            IF ILOOP = 7  THEN  STRNGO = "242A K1-6-2 FLT DP HI"
391  0250            IF ILOOP = 8  THEN  STRNGO = "242A K1-6-2 FLT DP LO"
392  0252            IF ILOOP = 9  THEN  STRNGO = "242A K1-6-3 FLT DP LO"
393  0254            IF ILOOP = 10 THEN  STRNGO = "242A K1-6-4 FLT DP LO"
394  0256            IF ILOOP = 11 THEN  STRNGO = "242A TURBINE FAN ON  "
395  0258            IF ILOOP = 12 THEN  STRNGO = "HPT AIRSMP CAM RAD HI"
396  0260            IF ILOOP = 13 THEN  STRNGO = "CND AIRSMP CAM RAD HI"
397  0262            IF ILOOP = 14 THEN  STRNGO = "HVAC AREA RAD HI     "
398  0264            IF ILOOP = 15 THEN  STRNGO = "CND LVL4 AREA RAD HI "
399  0266            IF ILOOP = 16 THEN  STRNGO = "CND BSMT AREA RAD HI "
400  0268            IF ILOOP = 17 THEN  STRNGO = "EVAP RM AREA RAD HI  "
401  0270            IF ILOOP = 18 THEN  STRNGO = "LOADOUT AREA RAD HI  "
402  0272            IF ILOOP = 19 THEN  STRNGO = "CND AIR SAMP CAM FAIL"
403  0274            IF ILOOP = 20 THEN  STRNGO = "AMU AIR SAMP CAM FAIL"
404  0276            IF ILOOP = 21 THEN  STRNGO = "HPT AIR SAMP CAM FAIL"
405  0278            IF ILOOP = 22 THEN  STRNGO = "EVAP RM AREA RAD FAIL"
406  0280            IF ILOOP = 23 THEN  STRNGO = "AMU AREA RAD FAIL    "
407  0282            IF ILOOP = 24 THEN  STRNGO = "HVAC AREA RAD FAIL   "
408  0284            IF ILOOP = 25 THEN  STRNGO = "LOADOUT AREA RAD FAIL"
409  0286            IF ILOOP = 26 THEN  STRNGO = "CND LV4 AREA RAD FAIL"
410  0288            IF ILOOP = 27 THEN  STRNGO = "CND BAS AREA RAD FAIL"
411  0290            IF ILOOP = 45 THEN  STRNGO = "242BATCH PGM START   "
412  0292            IF ILOOP = 46 THEN  STRNGO = "244BATCH WRONG UNIT  "
413  0294            IF ILOOP = 47 THEN  STRNGO = "244BATCH TERMINATING "
414          ;
415  0295    AMSG:   IF PTYPE = 1 THEN
416                     BEGIN
417  0297               IF (FACK = 0) & (FCLEAR = 0) & (FSTART = 0) THEN PRINT ADI,"WHITE ALARM %p",STRNGO
418  0298               IF (FACK) THEN
419                        BEGIN
420  0299                  PRINT ADI,"%p ALARM ACKNOWLEDGE",STRNGO
421  0300                  LET FACK = 0
422  0301                  GOTO NEX
423                        END
424                     END
425  0302            IF (FCLEAR) THEN
426                     BEGIN
427  0303               PRINT ADI,"%p ALARM CLEAR",STRNGO
428  0304               LET FCLEAR = 0
429  0305               GOTO NEX
430                     END
431  0306            IF (FSTART) THEN
432                     BEGIN
433  0307               LET FSTART = 0
434  0308               PRINT ADI,"%p",STRNGO
435  0309               GOTO NEX
436                     END
437  0310            COLORO = PTYPE
438  0311            INO = TIMSTR(SSNGO)
439  0313            IF INO != 8 THEN SSNGO = "TIMEFAIL"
440  0314            INO = APPEND(LSGO,STRNGO,SSNGO)
```

```
441   0316            IF INO < 0 THEN LSGO = "APPEND FAILED!!"
442   0317            INO = DATSTR(SDATEO)
443   0319            IF INO != 9 THEN SDATEO = "DATE FAIL"
444   0320    NEX:    RETURN
445           ;
446   0321    ERRSUB:  IF ERRN != ERRNUM() THEN
447                     BEGIN
448   0322              ERRN = ERRNUM()
449   0323              ERRL = ERRLIN()
450   0324              PRINT ADI,"ERROR # %i AT LINE %i, iloop = %i",errn,errl,iloop
451   0325              FSNO = 1
452   0326              TIMERON STMRO
453                     END
454   0327            ERSCLR
455   0329            IF ERRN = -187 THEN WAIT 5
456   0330            GOTO AGAIN
457           ;
458           ; TERM:  LET FABORT = 0
459   0331    TERM:   LET STRNGO = "242BATCH TERMINATED    "
460   0332            LET ILOOP = STAR + 2
461   0333            LET FSTART = 1
462   0334            LET SSNGO = "TERM"
463   0335            GOSUB ACKK
464                   END

      Total Errors Detected      0     Total Warnings      0
          Sequence Program Size = 6784 Bytes, 424 Pages
               16 Forward Reference(s)
```

PGM1A

Sequence Instruction Compiler Listing 15:36:22 24-Feb-95 Page 1

```
                ;       P G M 1 A . S E Q                                       VERSION 2
                ;
                ;
                ;       Program to monitor sequence programs running in PCM1
                ;       PGMSTAT sets bits in an integer to give the status of a program.
                ;       It indicates if the program is running, is critical, trapped etc.
                ;       This program fetches this value then, by using a bit mask, it
                ;       checks specific statuses and converts these to a single integer value
                ;       used by the string editor.  SED translates the integer to strings
                ;       in graphic 043, titled "PROGRAM STATUS".
                ;
                ;       Written By      : R. N. Kersey
                ;       Date            : 05-31-92
                ;       Facility        : 242-A Evaporator Bldg.
                ;       Machine         : VAX 4104-A
                ;       Op. System      : VAX/VMS 5.5-2AH
                ;       D/3 Version     : 6.3-2 BASELINE
                ;       Compiler        : SIC
                ;       Build command   : @[320,20]PGM1A
                ;
                ;       Assoc. UPF:     [300,1]_BLANKK.___      BLANKK.LST
                ;       Assoc. RPF:     [300,1]_EMTY.___        EMTY.LST
                ;       SDB:            [300,1]PCM1ASDB.CMD     PCM1ASDB.LST
                ;
                ;  MODIFICATION LOG:
                ;
                ;       29-JUL-93       Added comments and baselined for        M. Teats
                ;                       D/3 version 3.3.
                ;
                ;       15-DEC-94       Revised header to indicate th enew VMS          J. Berger
                ;                       system. Rebase line document.
                ;
                ;  END MOD LOG
                ;
                UNIT    BLANK1
                ;
                        PSTAT == PALARM
                        PLIST == PSPAR0
0000                    TIMERCLR STMR1
0001                    TIMERON STMR1
0002                    FABORT = 0
0003                    ASSIGN PLIST = INT0
0004                    FOR ILOOP = 0 TO 11
0005                    PLIST = 0
0006                    NEXT ILOOP
                ;
0007            AGAIN:  LSG6 = "WAIT"
0008                    ON ERROR GOSUB ERRSUB
0009                    IF (FSPAR0) THEN
0010                       IF STMR0 > 60 THEN
                              BEGIN
0011                          TIMEROFF STMR0
0012                          TIMERCLR STMR0
0013                          FSPAR5 = 0
0014                          ERRN = 0
```

```
 56    0015                ERRL = 0
 57                        END
 58    0016          WAIT 1
 59          ;       IF FABORT THEN GOTO TERM
 60    0017          LSG6 = "RUNNING"
 61    0018          WAIT 1
 62    0019          TIMERCLR STMR1
 63    0020          INT6 = PGMSTAT("WHITE1","WHOLE1")
 64    0021          INT7 = PGMSTAT("ANBATCH","ANFARM")
 65    0022          INT8 = PGMSTAT("APBATCH","APFARM")
 66    0023          INT9 = PGMSTAT("AWBATCH","AWFARM")
 67    0024          INT10 = PGMSTAT("XYZBATCH","XYZFRM")
 68    0025          INT11 = PGMSTAT("242BATCH","BLDNG")
 69    0026          ASSIGN PLIST = INT0
 70    0027          ASSIGN PSTAT = INT6
 71    0028          FTRIP1 = 0
 72    0029          TRIP = 0
 73    0030          FOR ILOOP = 0 TO 5
 74    0031          BVAL = 1
 75    0032          REMAIN = 0
 76    0033           FOR INDEX = 1 TO 4
 77    0035           IF AND(BVAL,PSTAT)THEN REMAIN = INDEX
 78    0036           BVAL = BVAL{1
 79    0037           NEXT INDEX
 80    0038          IF PSTAT = -1 THEN
 81                     BEGIN
 82    0039             PLIST = 6
 83    0040             FTRIP1 = 1
 84                     END
 85                     ELSE
 86                     BEGIN
 87    0041             PLIST = REMAIN
 88    0043             IF AND(4096,PSTAT)THEN PLIST = 5
 89                     END
 90    0044          TRIP = OR(TRIP,PSTAT)
 91    0045          STEP PLIST BY 1
 92    0046          STEP PSTAT BY 1
 93    0047          NEXT ILOOP
 94    0049          IF AND(TRIP,2) ¦ FTRIP1 THEN FTRIP0 = 1
 95    0050          ELSE FTRIP0 = 0
 96    0051          GOTO AGAIN
 97          ;
 98          ;=============================================================================
 99          ;                SUBROUTINES
100          ;
101    0052  ERRSUB:  TIMERON STMR1
102    0053          TIMERCLR STMR1
103    0054          FSPAR5 = 1
104          ;
105          ; errors are handled differently depending on their severity
106          ;
107          ;  19    PROGRAM ALREADY IN KICK SUBROUTINE (rekick)
108          ;  32    UNDEFINED COMMUNICATIONS PACKET TYPE (ignore)
109          ;  41   PCM DB HEADER NOT LOADED  (ignore)
110          ;  69   IPN NOT DEFINED (ignore)
```

```
111             ;  162  NO SUCH PROGRAM IN THIS PCM (ignore)
112             ;  165   SEQUENCE PROGRAM ALREADY RUNNING (ignore)
113             ;  170   SERIAL NUMBER MISMATCH (wait forever)
114             ;  179  ILLEGAL SUBROUTINE STACK CLEAR (ignore)
115             ;  187   BACKUP PCM IS DOWN (ignore)
116             ;  193   NO SUCH PROGRAM IN THIS PCM (ignore)
117             ;  198   ILLEGAL SUBROUTINE STACK NESTING ERROR (ignore)
118             ;  199   ILLEGAL RDF NAME (wait forever)
119             ;  200   RDF NOT LOADED IN MEMORY (wait forever)
120             ;  201  RECIPE LOAD NOT ENABLED (wait forever)
121             ;  222  BOARDS DEAD (ignore)
122             ;  226   BOARD ERROR E.G. broken skid panel etc...  (ignore)
123             ;  245   BOTH PCMS DOWN, SENT REDUNDANT HIGHWAY ACCESS (ignore)
124             ;
125     0055            ERRN = ABS(ERRNUM())     ; get error number
126     0056            ERRL = ERRLIN()          ; get error line number
127             ;
128             ; if error is not serious, clear and return to program
129             ;
130     0057            IF (ERRN=187) ¦ (ERRN=32) ¦ (ERRN=69) ¦ (ERRN=222) THEN
131     0058                ERSRET
132
133     0059            IF (ERRN=41) ¦ (ERRN=165) ¦ (ERRN=162) ¦ (ERRN=179) THEN
134     0060                ERSRET
135
136     0061            IF (ERRN=226) ¦ (ERRN=198) ¦ (ERRN=245) THEN
137     0062                ERSRET
138
139     0063            IF (ERRN=193) ¦ (ERRN=19) THEN
140     0064                ERSRET
141             ;
142             ; else report the error, and wait for corrective action ....
143     0065            PRINT 0,"NUMBER %i AT LINE %i ***",ERRN,ERRL
144             ;
145     0066            WAITUNTIL(0)                    ; wait for engineer corrective
146                                                     ; action
147     0067            GOTO AGAIN                      ; when corrective action taken,
148                                                     ; goto done and cleanup
149             ;
150             ;=========================================================================
151             ;
152             ;
153     0068    TERM:   ASSIGN PLIST = INTO
154     0069            FOR ILOOP = 0 TO 5
155     0070            PLIST = 6
156     0071            STEP PLIST BY 1
157     0072            NEXT ILOOP
158     0073            TIMEROFF STMR0
159     0074            TIMERCLR STMR0
160     0075            TIMEROFF STMR1
161     0076            TIMERCLR STMR1
162                     END

      Total Errors Detected      0     Total Warnings      0
          Sequence Program Size = 1456 Bytes, 91 Pages
               1 Forward Reference(s)
```

APPENDIX F - "C" PROGRAMS FOR ALARMS

DOGOA

```
/*      D O G O A . C                                              VERSION 2

        DOGOA (DOGOA.LTK) runs in micro PCM0A.

        DOGOA scans the FALMON and ACK flags on all sequence units in PCM0A.
        When FALMON is set by any sequence program, DOGOA turns on audiable
        alarm device AUDALARM0 and resets the FALMON flag to zero.  FALMON
        is set by the sequence programs if alarms found by sequence programs
        are yellow or white.

        When the flag ACK is set by any sequence program, DOGOA sets FSL0
        (Flag 168) in unit BLANK0, which is scanned every 2 seconds by
        WATCH.C (installed as task DOG running in the CDCM).  WATCH.C resets
        the FSL0 flag to zero and spawns task ACKFLG (in the CDCM).  Task
        ACKFLG acknowledges all alarms in that unit and resets the ACK flag
        to zero in the sequence program.
*/
/*  Written by : R. N. Kersey              */
/*  Date       : 12-May-92                 */
/*  Modified by: John Hayden T.I.:         6.3-2 Upgrade    */
/*  Facility   : 242-A Evaporator Bldg.            */
/*  Machine    : VAX 4105-A                        */
/*  Compiler   : DEC VAX C V3.3                    JH01     */
/*  VMS DCL .cmd use : SET DEF MICRO:[200,10]      JH01     */
/*  C .cmd use : BLD DOGOA DOGOA.C -L -MOV         JH01     */
/*  Op. System : VAX/VMS 5.5-2AH           JH01    */
/*  D/3 Version: 6.3-2                             JH01     */
/*                                                          */
/*  MODIFICATION LOG:                                                   */
/*                                                                      */
/*      16-Jul-92       Added description, format changes.       M. Teats */
/*                                                                      */
/*      04-Sep-92       Added/increased sleep to prevent M. Teats */
/*                      FALMON getting stuck on (mt1).                  */
/*                                                                      */
/*      19-JAN-93       Added 30 tic sleep before put_vseq.      R. Kersey*/
/*                                                                      */
/*      02-AUG-93       Baseline for D/3 version 3.3             M. Teats */
/*                                                                      */
/*      02-JUL-94       Baseline for D/3 version 6.3             J. Hayden */
/*                                                                      */
/*      01-SEP-94       Sleeps now seconds instead of ticks      J. Hayden */
/*                                                                      */
/*      02-SEP-94       %i now %d, #includes different           J. Hayden */
/*                                                                      */
/*      20-SEP-94       Paramaters for put_vseq calls changed   J. Hayden */
/*                                                                      */
/*      22-Sep-94       Inclused RSI fuction to put in infinite         */
/*                      loop                                     J. Hayden */
/*  END OF MOD LOG                                                      */
/*                                                                      */

#include <d3sys.h>
#include <dba.h>
#include <micstd.h>
#include <seqdba.h>
#include <conf.h>
#include <almmsg.h>
#include <err.h>
#include <comm.h>
#include <comcodes.h>
#include <mic.h>
#include <comstat.h>
#include <pcmii.h>


main()
{
char    net, unit, type, field, cmd, pcm ;
int     access, k, vprt, alm_on, flg_on, value, select,
```

```
        ipn, extra, zero, err, crid, ack, ackflg, index ;

access = UNITREL ; net = 0 ; zero = 0 ; cmd = 1 ;
vprt = 0 ; alm_on = 0 ; flg_on = 0 ; value = 0 ; select = 0 ;
ipn = 0 ; extra = 0 ; err = 0 ; ack = 0 ; ackflg = 0 ; index = 0 ;
checkerr(1, mopen()) ;

FOREVER
   {                                                                    /* Infinite Loop
        */
      RSI(100) ;
      ackflg  = 0 ;  alm_on = 0 ;
      for ( k = 1; k < 22; k++ )                /* 21 Units, skip 19 (ROOMS). 14 handled outside Loop
*/
         {
         index = 364 ;                                                  /* FALMON offset 204 for UPF
SHORTO   */
         if ( k < 5 ) index = 204 ;                                     /* FALMON offset 364 for UPF
TNKFRM   */
         if ( k != 19 )
            {
            checkerr ( 3, get_vseq ( net, 2, FG, access, k, index, &vprt )) ;          /* Get FALMON
Flag     */
            if ( vprt == 1 )
               {
               alm_on = 1 ;
               sleep (1) ;                                                              /*
added             */
               checkerr ( 4, put_vseq ( net, 2, FG, access, k, index, &zero ));       /* Reset FALMON
Flag     */
               sleep(1);                                                                /* added
(mt1)             */
               }
            checkerr ( 4, get_vseq ( net, 2, FG, access, k, index+3, &ack )) ;       /* Get ACK Flag
         */
            if ( ack == 1 ) ackflg = 1 ;
            sleep(1);
            }
         }                                                              /* End FOR Loop
        */

      /* If ACKFLG = 1 then set Flag 168 in sequence unit BLANK0 in PCM0A
*/
      /* If ALM_ON = 1 then turn on device AUDALARM0 in PCM0A
        */

      if ( ackflg == 1 )   checkerr ( 8, put_vseq ( net, 2, FG, access, 25, 168, &ackflg )) ;
      if ( alm_on == 1)    checkerr ( 9, devcmd ( 0, 2, DEV, 171, &cmd )) ;
      /* TAKE A REST
        */
      sleep (3) ;                                                       /* changed from 10 to 100 (mt1)
        */
   }                                            /* End FOREVER                                 */
}                                               /* End MAIN                                    */

checkerr(x,ierr)
int ierr,x ;
{
int r2,r3 ;
if (ierr < 0 )
  {
   r2 = lnkrg("CONFIG");
   r3 = lnkrg("ALMQUE");
   sndmsg ( 0, 0, r3, r2, 3, MSG_TEXT, CRIT," DOGOB.LTK err %d at %d, DOGOB IS TERMINATING!!!!!!\n",ierr,x)
;
   sndmsg ( 0, 0, r3, r2, 3, MSG_TEXT, CRIT," DOGOB.LTK err %d at %d, DOGOB IS TERMINATING!!!!!!\n",ierr,x)
;
   exit();
   }
}
```

DOGOB

```
/*      D O G O B . C                                          VERSION 2

        DOGOB (DOGOB.LTK) runs in micro PCM0A.

        DOGOB scans the FALMON and ACK flags on all sequence units in PCM0A.
        When FALMON is set by any sequence program, DOGOB turns on audiable
        alarm device AUDALARM0 and resets the FALMON flag to zero.  FALMON
        is set by the sequence programs if alarms found by sequence programs
        are yellow or white.

        When the flag ACK is set by any sequence program, DOGOB sets FSL0
        (Flag 168) in unit BLANK0, which is scanned every 2 seconds by
        WATCH.C (installed as task DOG running in the CDCM).  WATCH.C resets
        the FSL0 flag to zero and spawns task ACKFLG (in the CDCM).  Task
        ACKFLG acknowledges all alarms in that unit and resets the ACK flag
        to zero in the sequence program.
/*                                                              */
/*  Written by : R. N. Kersey                                   */
/*  Date       : 12-May-92                                      */
/*  Facility   : 242-A Evaporator Bldg. */
/*  Modified by: John Hayden T.I.          6.3-2 Upgrade    */
/*  Facility   : 242-A Evaporator Bldg.           */
/*  Machine    : VAX 4105-A                                     */
/*  Compiler   : DEC VAX C V3.3                        JH01     */
/*  VMS DCL .cmd use : SET DEF MICRO:[200,10]          JH01     */
/*  C .cmd use : BLD DOGOB DOGOB.C -L -MOV             JH01     */
/*  Op. System : VAX/VMS 5.5-2AH           JH01     */
/*  D/3 Version: 6.3-2                                 JH01     */
/*                                                              */
/*  MODIFICATION LOG:                                                   */
/*                                                                      */
/*      16-Jul-92       Added description, format changes.       M. Teats */
/*                                                                      */
/*      04-Sep-92       Added/increased sleep to prevent M. Teats */
/*                      FALMON getting stuck on (mt1).                  */
/*                                                                      */
/*      19-JAN-93       Added 30 tic sleep before put_vseq.      R. Kersey*/
/*                                                                      */
/*      02-AUG-93       Baseline for D/3 version 3.3             M. Teats */
/*                                                                      */
/*      02-JUL-94       Baseline for D/3 version 6.3             J Hayden  */
/*                                                                       */
/*      01-SEP-94       Sleeps now seconds instead of ticks      J. Hayden */
/*                                                                       */
/*      02-SEP-94       %i now %d, #includes different           J. Hayden */
/*                                                                       */
/*      20-SEP-94       Paramaters for put_vseq calls changed   J. Hayden */
/*                                                                       */
/*      22-Sep-94       Inclused RSI fuction to put in infinite          */
/*                      loop                                     J. Hayden */
/*  END OF MOD LOG                                                      */

#include <d3sys.h>
#include <dba.h>
#include <micstd.h>
#include <seqdba.h>
#include <conf.h>
#include <almmsg.h>
#include <err.h>
#include <comm.h>
#include <comcodes.h>
#include <mic.h>
#include <comstat.h>
#include <pcmii.h>


main()
{
char    net, unit, type, field, cmd ;
int     access, k, vprt, alm_on, flg_on, value, select,
```

```
        ipn, extra, zero, err, crid, ack, ackflg, index ;

access = UNITREL ; net = 0 ; zero = 0 ; cmd = 1 ;
vprt = 0 ; alm_on = 0 ; flg_on = 0 ; value = 0 ; select = 0 ;
ipn = 0 ; extra = 0 ; err = 0 ; ack = 0 ; ackflg = 0 ; index = 0 ;
checkerr(1, mopen()) ;

FOREVER
   {                                                                    /* Infinite Loop
        */
      RSI(100) ;
      ackflg  = 0 ;  alm_on = 0 ;
      for ( k = 1; k < 22; k++ )                /* 21 Units, skip 19 (ROOMS). 14 handled outside Loop
*/
         {
         index = 364 ;                                                  /* FALMON offset 204 for UPF
SHORTO   */
         if ( k < 5 ) index = 204 ;                                     /* FALMON offset 364 for UPF
TNKFRM   */
         if ( k != 19 )
            {
            checkerr ( 3, get_vseq ( net, 3, FG, access, k, index, &vprt )) ;          /* Get FALMON
Flag     */
            if ( vprt == 1 )
               {
               alm_on = 1 ;
               checkerr ( 4, put_vseq ( net, 3, FG, access, k, index, &zero ));        /* Reset FALMON
Flag     */
               sleep(1);                                                /* added
(mt1)           */
               }
            checkerr ( 4, get_vseq ( net, 3, FG, access, k, index+3, &ack )) ;       /* Get ACK Flag
         */
            if ( ack == 1 ) ackflg = 1 ;
            sleep(1);
            }
         }                                                              /* End FOR Loop
        */

      /* If ACKFLG = 1 then set Flag 168 in sequence unit BLANK0 in PCM0A
*/
      /* If ALM_ON = 1 then turn on device AUDALARM0 in PCM0A
        */

      if ( ackflg == 1 )   checkerr ( 8, put_vseq ( net, 3, FG, access, 25, 168, &ackflg )) ;
      if ( alm_on == 1)    checkerr ( 9, devcmd ( 0, 3, DEV, 171, &cmd )) ;

      /* TAKE A REST
        */

      sleep (3) ;                                                       /* changed from 10 to 100 (mt1)
        */
   }                                            /* End FOREVER                          */
}                                               /* End MAIN                             */

checkerr(x,ierr)
int ierr,x ;
{
int r2,r3 ;
if (ierr < 0 )
  {
   r2 = lnkrg("CONFIG");
   r3 = lnkrg("ALMQUE");
   sndmsg ( 0, 0, r3, r2, 3, MSG_TEXT, CRIT," DOGOB.LTK err %d at %d, DOGOB IS TERMINATING!!!!!!\n",ierr,x)
;
   sndmsg ( 0, 0, r3, r2, 3, MSG_TEXT, CRIT," DOGOB.LTK err %d at %d, DOGOB IS TERMINATING!!!!!!\n",ierr,x)
;
   exit();
   }
}
```

DOG1A

```
/*      D O G 1 A . C                                              VERSION 2

        DOG1A (DOG1A.LTK) runs in micro PCM1A.

        DOG1A scans the FALMON and FLGACK flags on all sequence units in PCM1A.
        When FALMON is set by any sequence program, DOG1A turns on audiable
        alarm device AUDALARM0 and resets the FALMON flag to zero.  FALMON
        is set by the sequence programs if alarms found by sequence programs
        are yellow or white.

        When the flag FLGACK is set by any sequence program, DOG1A sets FSL0
        (Flag 168) in unit BLANK1, which is scanned every 2 seconds by
        WATCH.C (installed as task DOG running in the CDCM).  WATCH.C resets
        the FSL0 flag to zero and spawns task ACKFLG (in the CDCM).  Task
        ACKFLG acknowledges all alarms in that unit and resets the FLGACK flag
        to zero in the sequence program.
 */
/*  Written by : R. N. Kersey              */
/*  Date       : 27-May-92                 */
/*  Modified by: John Hayden T.I.          6.3-2 Upgrade    */
/*  Facility   : 242-A Evaporator Bldg.            */
/*  Machine    : VAX 4104-A                                 */
/*  Compiler   : DEC VAX C V3.3                     JH01    */
/*  VMS DCL .cmd use : SET DEF MICRO:[200,10]       JH01    */
/*  C .cmd use : BLD DOG1A DOG1A.C -L -MOV          JH01    */
/*  Op. System : VAX/VMS 5.5-2AH           JH01    */
/*  D/3 Version: 6.3-2                              JH01    */
/*                                                          */
/*                                         */
/*  MODIFICATION LOG:                                                   */
/*                                                                      */
/*      13-Jul-92       Removed LNKRG('CONFIG') after mopen.      M. Teats */
/*                      Reformatted brackets.                           */
/*                                                                      */
/*      16-Jul-92       Added description, format changes.        M. Teats */
/*                                                                      */
/*      21-Jul-92       Implemented micro select feature.         M. Teats */
/*                      Changed 'err = ' to 'checkerr'.                 */
/*                      Added CHECKERR subroutine.                      */
/*                      Deleted CONTINUE if err < 0.                    */
/*                      Changed 'cmd' from int to char to allow         */
/*                         DEVCMD to turn on horn.                      */
/*                      Initialize 'cmd = 1' at program start and       */
/*                         removed 'cmd = 1' from IF #8 when            */
/*                         AUDALARM0 turned on.                         */
/*                      Rewrote last half of code.                      */
/*                                                                      */
/*      12-FEB-93       - Added sleep as noted before and         M. Teats */
/*                        after put_vseq                                */
/*                      - Removed phasing (& put_vseq) on               */
/*                        FHOLD (done by sequence programs)             */
/*                      These are to prevent program lock-up.           */
/*                                                                      */
/*      02-AUG-93       Baseline for D/3 version 3.3                    */
/*                                                                      */
/*      03-JUL-93       Baseline for D/3 version 6.3-2            J. Hayden */
/*                                                                      */
/*      01-SEP-94       Sleeps now seconds instead of ticks       J. Hayden */
/*                                                                      */
/*      02-SEP-94       %i now %d, #includes different            J. Hayden */
/*                                                                      */
/*      20-SEP-94       Paramaters for put_vseq calls changed   J. Hayden */
/*                                                                      */
/*      22-Sep-94       Inclused RSI fuction to put in infinite         */
/*                      loop                                      J. Hayden */
/*  END OF MOD LOG                                                      */
/*                                                                      */

#include <d3sys.h>
#include <dba.h>
```

```
#include <micstd.h>
#include <seqdba.h>
#include <conf.h>
#include <almmsg.h>
#include <err.h>
#include <comm.h>
#include <comcodes.h>
#include <mic.h>
#include <comstat.h>
#include <pcmii.h>

main()
{                                                                    /* be1 */
char    net, unit, type, field, cmd ;
int     access, k, vprt, alm_on, one, select,
        flg_on, value, ipn, extra, phase,
        zero, err, crid, total ;

access = UNITREL ; net = 0 ; zero = 0 ; phase = 0 ; one = 1 ; cmd = 1;
vprt = 0 ; alm_on = 0 ; select = 0 ; flg_on = 0 ; value = 0 ; ipn = 0 ;
extra = 0 ; err = 0 ; total = 0 ;
checkerr (1, mopen());

FOREVER
   {                                                                 /* be2 */
      RSI(100) ;
      alm_on = 0 ;
      total  = 0 ;

/*       A L A R M:                                    */
/*       Get FALMON Flag 492, then reset it to zero:   */
      for ( k = 1 ; k < 6 ; k++ )                                    /* For 6 Sequence Units  */
         {                                                           /* be4 */
         checkerr ( 3, get_vseq ( 0, 4, FG, access, k, 492, &vprt )) ;
         if (vprt==1)
            {                                                        /* be5 */
            alm_on = 1 ;
            checkerr ( 4, put_vseq ( 0, 4, FG, access, k, 492, &zero )) ;
            sleep (1) ;
/*          sndmsg(0,0,-1,crid,1,MSG_TEXT,NONCRIT,"D O G 1 A . C - FALMON FLAG SET ON UNIT %d\n",k) ;
*/
            }                                                        /* en5 */

/*       A C K N O W L E D G E:                        */
/*       Get FLGACK Flag 495, then reset it to zero:   */
         checkerr ( 5, get_vseq ( 0, 4, FG, access, k, 495, &vprt )) ;
         if (vprt == 1 )
            {                                                        /* be6 */
            total = 1 ;
            if ( k != 5 )
                {                                                    /* be7 */
                checkerr ( 6, put_vseq ( 0, 4, FG, access, k, 495, &zero )) ;
                sleep (1) ;                                          /* added                 */
                }                                                    /* en7 */
            }                                                        /* en6 */
         }                                                           /* en4, End FOR Loop     */

/*    If FALMON = 1, then turn on device AUDALARM0 in PCM0A:       */
      if (alm_on == 1 )
         {                                                           /* be8 */
         checkerr ( 7, devcmd ( 0, 2, DEV, 171, &cmd )) ;
         }                                                           /* en8 */

/*    If FLGACK = 1, then set Flag 168 (FSL0) in unit BLANK1 in PCM1A:     */
      if (total == 1 )   checkerr ( 8, put_vseq ( 0, 4, FG, access, 6, 168, &total )) ;

/*    Phase the FHOLD flag 479 to turn off Lamp:                                    */
/*      if (phase == 1)  put_vseq ( 0, 4, FG, access, 1, 479, &one ) ;     deleted  */
/*      if (phase == 7)  put_vseq ( 0, 4, FG, access, 2, 479, &one ) ;     deleted  */
/*      if (phase == 14) put_vseq ( 0, 4, FG, access, 3, 479, &one ) ;     deleted  */
```

```
/*      if (phase == 21) put_vseq ( 0, 4, FG, access, 4, 479, &one ) ;          deleted        */
/*      phase++ ; if (phase >= 22) phase = 1 ;                                  deleted        */

      sleep (3) ;
   }                                                                  /* en2, End FOREVER     */
}                                                                     /* en1, End main        */

/* CHECKERR SUBROUTINE:          */
checkerr(x,ierr)
int ierr,x ;
{
int r2,r3 ;
if (ierr < 0 )
  {
   r2 = lnkrg("CONFIG");
   r3 = lnkrg("ALMQUE");
   sndmsg ( 0, 0, r3, r2, 3, MSG_TEXT, CRIT," DOG1A.LTK err %d at %d\n",ierr,x) ;
   exit();
   }
}
```

DOG1B

```
/*      D O G 1 B . C                                          VERSION 2

        DOG1b (DOG1b.LTK) runs in micro PCM1b.

        DOG1b scans the FALMON and FLGACK flags on all sequence units in PCM1b.
        When FALMON is set by any sequence program, DOG1b turns on audiable
        alarm device AUDALARM0 and resets the FALMON flag to zero.  FALMON
        is set by the sequence programs if alarms found by sequence programs
        are yellow or white.

        When the flag FLGACK is set by any sequence program, DOG1b sets FSL0
        (Flag 168) in unit BLANK1, which is scanned every 2 seconds by
        WATCH.C (installed as task DOG running in the CDCM).  WATCH.C resets
        the FSL0 flag to zero and spawns task ACKFLG (in the CDCM).  Task
        ACKFLG acknowledges all alarms in that unit and resets the FLGACK flag
        to zero in the sequence program.
 */
/*  Written by : R. N. Kersey              */
/*  Date       : 27-May-92                 */
/*  Modified by: John Hayden T.I.          6.3-2 Upgrade    */
/*  Facility   : 242-A Evaporator Bldg.              */
/*  Machine    : VAX 4104-A                          */
/*  Compiler   : DEC VAX C V3.3                      JH01    */
/*  VMS COMMAND use : SET DEF MICRO:[200,10]         JH01    */
/*  C .cmd use : BLD DOG1B DOG1B.C -L -MOV           JH01    */
/*  Op. System : VAX/VMS 5.5-2AH           JH01      */
/*  D/3 Version: 6.3-2                               JH01    */
/*                                                   */
/*                                         */
/*  MODIFICATION LOG:                                                    */
/*                                                                       */
/*      13-Jul-92       Removed LNKRG('CONFIG') after mopen.     M. Teats */
/*                      Reformatted brackets.                             */
/*                                                                        */
/*      16-Jul-92       Added description, format changes.       M. Teats */
/*                                                                        */
/*      21-Jul-92       Implemented micro select feature.        M. Teats */
/*                      Changed 'err = ' to 'checkerr'.                   */
/*                      Added CHECKERR subroutine.                        */
/*                      Deleted CONTINUE if err < 0.                      */
/*                      Changed 'cmd' from int to char to allow           */
/*                         DEVCMD to turn on horn.                        */
/*                      Initialize 'cmd = 1' at program start and         */
/*                         removed 'cmd = 1' from IF #8 when              */
/*                         AUDALARM0 turned on.                           */
/*                      Rewrote last half of code.                        */
/*                                                                        */
/*      12-FEB-93       - Added sleep as noted before and        M. Teats */
/*                        after put_vseq                                  */
/*                      - Removed phasing (& put_vseq) on                 */
/*                        FHOLD (done by sequence programs)               */
/*                      These are to prevent program lock-up.             */
/*                                                                        */
/*      02-AUG-93       Baseline for D/3 version 3.3                      */
/*                                                                        */
/*      02-JUL-94       Baseline for D/3 version 6.3-2           J Hayden */
/*                                                                        */
/*      01-SEP-94       Sleeps now seconds instead of ticks      J. Hayden */
/*                                                                        */
/*      02-SEP-94       %i now %d, #includes different           J. Hayden */
/*                                                                        */
/*      20-SEP-94       Paramaters for put_vseq calls changed   J. Hayden */
/*                                                                        */
/*      22-Sep-94       Inclused RSI fuction to put in infinite           */
/*                      loop                                     J. Hayden */
/*                                                                        */
/*  END OF MOD LOG                                                       */
/*                                                                       */
#include <d3sys.h>
#include <dba.h>
```

```
#include <micstd.h>
#include <seqdba.h>
#include <conf.h>
#include <almmsg.h>
#include <err.h>
#include <comm.h>
#include <comcodes.h>
#include <mic.h>
#include <comstat.h>
#include <pcmii.h>

main()
{                                                                    /* be1 */
char    net, unit, type, field, cmd ;
int     access, k, vprt, alm_on, one, select,
        flg_on, value, ipn, extra, phase,
        zero, err, crid, total ;

access = UNITREL ; net = 0 ; zero = 0 ; phase = 0 ; one = 1 ; cmd = 1;
vprt = 0 ; alm_on = 0 ; select = 0 ; flg_on = 0 ; value = 0 ; ipn = 0 ;
extra = 0 ; phase = 0 ; err = 0 ; total = 0;
checkerr (1, mopen());

FOREVER
   {                                                                 /* be2 */
      RSI(100) ;
      alm_on = 0 ;
      total  = 0 ;

/*       A L A R M:                                        */
/*       Get FALMON Flag 492, then reset it to zero:       */
      for ( k = 1 ; k < 6 ; k++ )                                    /* For 6 Sequence Units  */
         {                                                           /* be4 */
         checkerr ( 3, get_vseq ( 0, 5, FG, access, k, 492, &vprt )) ;
         if (vprt==1)
            {                                                        /* be5 */
            alm_on = 1 ;
            checkerr ( 4, put_vseq ( 0, 5, FG, access, k, 492, &zero )) ;
            sleep (1) ;
/*          sndmsg(0,0,-1,crid,1,MSG_TEXT,NONCRIT,"D O G 1 A . C - FALMON FLAG SET ON UNIT %d\n",k) ;
*/
            }                                                        /* en5 */

/*       A C K N O W L E D G E:                            */
/*       Get FLGACK Flag 495, then reset it to zero:       */
         checkerr ( 5, get_vseq ( 0, 5, FG, access, k, 495, &vprt )) ;
         if (vprt == 1 )
            {                                                        /* be6 */
            total = 1 ;
            if ( k != 5 )
                {                                                    /* be7 */
                checkerr ( 6, put_vseq ( 0, 5, FG, access, k, 495, &zero )) ;
                sleep (1) ;                                          /* added               */
                }                                                    /* en7 */
            }                                                        /* en6 */
         }                                                           /* en4, End FOR Loop     */

/*    If FALMON = 1, then turn on device AUDALARM0 in PCM0A:       */
      if (alm_on == 1 )
         {                                                           /* be8 */
         checkerr ( 7, devcmd ( 0, 2, DEV, 171, &cmd )) ;
         }                                                           /* en8 */

/*    If FLGACK = 1, then set Flag 168 (FSL0) in unit BLANK1 in PCM1b:     */
      if (total == 1 )   checkerr ( 8, put_vseq ( 0, 5, FG, access, 6, 168, &total )) ;

/*    Phase the FHOLD flag 479 to turn off Lamp:                                    */
/*       if (phase == 1)  put_vseq ( 0, 4, FG, access, 1, 479, &one ) ;     deleted */
/*       if (phase == 7)  put_vseq ( 0, 4, FG, access, 2, 479, &one ) ;     deleted */
/*       if (phase == 14) put_vseq ( 0, 4, FG, access, 3, 479, &one ) ;     deleted */
```

```
/*      if (phase == 21) put_vseq ( 0, 4, FG, access, 4, 479, &one ) ;          deleted        */
/*      phase++ ; if (phase >= 22) phase = 1 ;                                  deleted        */

      sleep (3) ;
   }                                                                  /* en2, End FOREVER     */
}                                                                     /* en1, End main        */

/* CHECKERR SUBROUTINE:          */
checkerr(x,ierr)
int ierr,x ;
{
int r2,r3 ;
if (ierr < 0 )
  {
   r2 = lnkrg("CONFIG");
   r3 = lnkrg("ALMQUE");
   sndmsg ( 0, 0, r3, r2, 3, MSG_TEXT, CRIT," DOG1b.LTK err %d at %d\n",ierr,x) ;
   exit();
   }
}
```

WATCH

```
/*      W A T C H . C              VERSION 2                          */
/*      Written by : R. N. Kersey                                     */
/*      Date       : 12-May-92                                        */
/*      Facilty    : 242-A Evaporator Bldg.                           */
/*      Machine          : VAX 4104-A                                 */
/*      Op. System       : VAX/VMS 5.5-2AH                            */
/*      D/3 Version      : 6.3-1 BASELINE                             */
/*      Compiler   : VAX C                                            */
/*      C .cmd use : BLD -L -ALL -MOV -LIS WATCH.SCR                  */

/*      Program to acknowledge, from a graphic, all alarms in that graphic (and
 *      the corresponding sequence unit - since there is one sequence unit per
 *      graphic).  Resets flag FSL0 to zero in Sequence Units BLANK0 and
 *      BLANK1, and runs function ACKFLG().
 *
 *      How WATCH.C works:  WATCH.C is installed as task DOG in the CDCM.
 *      It works in conjunction with DOG0A and DOG1A running in the micros.
 *      These three programs communicate via sequence variable flags in the
 *      various sequence units to turn on device AUDALARM0 for yellow and
 *      white alarms and acknowledge (from a single graphic) all analog and
 *      digital alarms in that graphic.
 *
 *      Each sequence program sets flag FALMON to 1 when a new yellow or white
 *      alarm occurs.  DOG0A and DOG1A scan the sequence units for FALMON flags
 *      that are set to 1.  When the flag is found, DOG0A or DOG1A resets
 *      FALMON to zero.  Any FALMON flag set to 1 causes AUDALARM0 to be
 *      commanded on, which turns on the yellow/white audiable alarm horn.
 *
 *      Also, DOG0A looks at flag ACK (in PCM0 sequence units) and DOG1A looks
 *      at flag FLGACK (in PCM1 sequence units).  If ACK is set to 1, DOG0A
 *      sets flag FSL0 in unit BLANK0.  If FLGACK is set to 1, DOG1A sets flag
 *      FSL0 in unit BLANK1.  WATCH scans the FSL0 flags in these two units.
 *      If FSL0 is set to 1, WATCH resets them to zero and runs function
 *     ACKFLG() contained within the WHTCH.C program. This function looks
 *      at all the sequence units, finds the ACK and FLGACK flags, resets
 *      them to zero, and acknowledges all EPNs only in the unit corresponding
 *      to the currently displayed graphic.
 *
 *      NOTE: The redundant program for DOG0A in PCM0B is DOG0B.C, and
 *                                for DOG1A in PCM1B is DOG1B.C
 *
 *
 *                  .-------------------------------------------------.
 *                  |                Sequence Program Flags           |
 *                  |                                                 |
 *                  |    .-----------.          .------------------.  |
 *          .-------|--> |   FALMON  |          |  ACK (PCM0), or  |  | <-------.
 *          |       |    |___________|          |  FLGACK (PCM1)   |  |         |
 *          |       |          V               |__________________|  |         |
 *          |       |          |                        V            |         |
 *          |       |__________|_____________________   |   _________|         |
 *          |                  |                     |  |  |                   |
 *          |                  V                     V  V  V                   |
 *          |       .-------------------------------------------------.        |
 *          |       |      DOG0A.C, DOG0B.C, DOG1A.C, or DOG1B.C      |        |
 *          |       |.....................__.........................|        |
 *          |       |                     |  |  .------------------.  |        |
 *          |---<   |  * Resets FALMON    |  |  |  Sets Flag FSL0  |  |        |
 *          |       |                     |  |  |   (FLAG 168) in  |  | <------.  |
 *          |---<   |  * Turns on AUDALARM0  |  |  BLANK0, BLANK1  |  |      |  |
 *          |       |                     |  |  |__________________|  |      |  |
 *          |       |_____________________|  |_____ V _____________|  |      |  |
 *          |                                      |                   |      |  |
 *          |                                      V                   |      |  |
 *          |       .-----------------------.    .--------------------.      |  |
 *          |------>|  Press SILENCE Key    |    |  WATCH.C (Ins. DOG)|      |  |
 *          |       |                       |    |....................|      |  |
 *          |       |  * Turns off AUDALARM0|    |  * Resets FSL0     |>-----'  |
 *          |       |_______________________|    |                    |         |
 *          |                                    |  * Function ACKFLG |         |
 *          |                                    |____________________|         |
 *          |                                                                 |
 *          |                                                                 |
 *          |                                                                 
```

```
*                                          V
*                                          |
*                                          |
*                                 .------------------------.
*                                 |  FUNCTION ACKFLG()     |
*                                 |------------------------|
*                                 |  * Resets sequence     |\
*                                 |    program ACK or      | >------'
*                                 |    FLGACK Flags        |/
*                                 |                        |
*                                 |  * Acknowledges all    |
*                                 |    alarms in a single  |
*                                 |    sequence unit       |
*                                 '------------------------'
*
*
*
```

```
    if ( pcm0==1 || pcm1==1 ) ackflg();

    if (pcm0 == 1 ) err = put_vseq ( net, 2, FG, access, 25, 168, &zero ) ;

         /* Reset flags */
    if (pcm1 == 1 ) err = put_vseq ( net, 4, FG, access, 6, 168, &zero ) ;

    err = get_vseq ( net, 4, FG, access, 6, 181, &vprt ) ;
    err = put_vseq ( net, 2, FG, access, 25, 183, &vprt ) ;
                          /* Flag FTRIP2, Unit(25) Blank0  */
/*                                                        */
/*       ABANDONED CODE (may be used for testing) TO TURN ON AUDALARM0:
         */
/*
         */
/* err = get_vseq ( net, 2, FG, 25, access, 184, &extra ) ;          Flag FTRIP3, Unit (25) Blank0
         */
/* if ( extra == 1)
         */
/*    {                                                           Test for AUDALARM0 using FTRIP3
in SSD  */
/*    err = put_vseq ( net, 2, FG, 25, access, 184, &zero ) ;      Reset FTRIP3
         */
/*    }
         */
/*
         */
/*       END ABANDONED CODE
         */
/*
         */

   sleep (2) ;

   }                                                  /* End FOREVER */

}                                                     /* End Main */
void ackflg(void)
{
        struct s1
        {
            char pcm;
            int  sunit,
                 index,
                 bloks,
                 offset;
        };

        static struct s1  seqrec [] =
                {

/*  k           struct s1 seqrec []        seq unit name          */
/* --           ------------------         ------------           */
/*  0  */         2,  1, 207, 2,  0,   /* A244                    */
/*  1  */         2,  2, 207, 1,  2,   /* HVC244          */
/*  2  */         2,  3, 207, 2,  3,   /* WASTE           +mct3-  */
/*  3  */         2,  4, 207, 1,  5,   /* A350                    */
/*  4  */         2,  5, 367, 1,  6,   /* FEED                    */
/*  5  */         2,  6, 367, 1,  7,   /* CA1                     */
/*  6  */         2,  7, 367, 1,  8,   /* PB1                     */
/*  7  */         2,  8, 367, 1,  9,   /* PB2                     */
/*  8  */         2,  9, 367, 1, 10,   /* EA1                     */
/*  9  */         2, 10, 367, 1, 11,   /* EC1                     */
/* 10  */         2, 11, 367, 2, 12,   /* RWATER          */
/* 11  */         2, 12, 367, 1, 14,   /* URW                     */
/* 12  */         2, 13, 367, 2, 15,   /* VVENT                   */
/* 13  */         2, 14, 367, 1, 17,   /* PC                      */
/* 14  */         2, 15, 367, 1, 18,   /* IX                      */
/* 15  */         2, 16, 367, 1, 19,   /* SC                      */
```

```
/* 16  */          2, 17, 367, 1, 20,   /* BASINS          */
/* 17  */          2, 18, 367, 1, 21,   /* UTIL                    */
/* 18  */          2, 20, 367, 1, 22,   /* AMU                     */
/* 19  */          2, 21, 367, 1, 23,   /* LERF                    */
/* 20  */          4,  1, 447, 1, 24,   /* ANFARM          */
/* 21  */          4,  1, 450, 1, 25,   /* ANFARM - ANHVC  */
/* 22  */          4,  1, 449, 1, 26,   /* ANFARM - ANVLV  */
/* 23  */          4,  1, 451, 1, 27,   /* ANFARM - AN271  */
/* 24  */          4,  2, 447, 2, 28,   /* APFARM          */
/* 25  */          4,  2, 450, 1, 30,   /* APFARM - APHVC  */
/* 26  */          4,  2, 449, 3, 31,   /* APFARM - APVLV  */
/* 27  */          4,  2, 451, 2, 34,   /* APFARM - AP271  */
/* 28  */          4,  2, 452, 4, 36,   /* APFARM - AP102  */
/* 29  */          4,  3, 447, 1, 40,   /* AWFARM          */
/* 30  */          4,  3, 450, 2, 41,   /* AWFARM - AWHVC  */
/* 31  */          4,  3, 449, 2, 43,   /* AWFARM - AWVLV  */
/* 32  */          4,  3, 451, 3, 45,   /* AWFARM - AW271  */
/* 33  */          4,  4, 447, 1, 48,   /* XYZFRM          */
/* 34  */          4,  4, 448, 2, 49,   /* XYZFRM - AFRM           */
/* 35  */          4,  4, 449, 2, 51,   /* XYZFRM - AYFRM  */
/* 36  */          4,  4, 450, 2, 53,   /* XYZFRM - AY, AZFRM      */
/* 37  */          4,  5, 495, 2, 55    /* BLDNG           -mct3+  */
                  };

        struct s2
        {
            char pcm;
            int  fstDIN,
                 totDIN,
                 fstAI,
                 totAI,
                 fstDEV,                                    /* -mct2+ */
                 totDEV;                                    /* -mct2+ */
        };

        static struct s2  datblk [] =
                  {
/*   Added fstDEV and totDEV columns to struct s2 -          *//* -mct2+ */
/*                                                           */
/*  0 */          2,   0,  15,   0, 11,   0, 0,   /* A244    */
/*  1 */          2,  17,   3,   0,  0,   0, 0,   /* A244    */
/*  2 */          2,  20,   5,  11,  3,   0, 0,  /* HVC244 */
/*  3 */          2,  25,  18,   0,  0,   0, 0,  /* WASTE   */
/*  4 */          4, 243,   1,   0,  0,   0, 0,  /* WASTE   *//* -mct3+ */
/*  5 */          2,  44,   3,  14,  7,   0, 0,   /* A350    */
/*  6 */          2,  50,   7,  30, 10,   0, 0,  /* FEED    */
/*  7 */          2,  60,   4,  45, 30,   0, 0,  /* CA1     */
/*  8 */          2,  70,  12,  80, 16,   0, 0,  /* PB1     *//* -jfb3+ */
/*  9 */          2,  85,  17, 100, 15,   0, 0,  /* PB2     *//* -mct1+ */
/* 10 */          2, 105,   3, 120, 21,   0, 0,   /* EA1     *//* -mct1+ */
/* 11 */          2, 110,   5, 150, 24,   0, 0,  /* EC1     *//* -mct1+ */
/* 12 */          2, 120,   9, 180,  5,   0, 0,  /* RWATER */
/* 13 */          4, 242,   1,   0,  0,   0, 0,  /* RWATER */
/* 14 */          2, 130,   4, 190, 11, 102, 1,  /* URW     *//* -mct2+ */
/* 15 */          2, 140,  13, 205, 16,   0, 0,  /* VVENT   */
/* 16 */          2, 154,   3,   0,  0,   0, 0,  /* VVENT   */
/* 17 */          2, 160,   5, 225, 12,   0, 0,  /* PC      */
/* 18 */          2, 170,  14, 240, 21, 133, 1,   /* IX      *//* -mct2+ */
/* 19 */          2, 190,  13, 265, 11, 143, 1,   /* SC      *//* -mct2+ */
/* 20 */          2, 210,  17, 280,  2, 151, 1,   /* BASINS *//* -mct2+ */
/* 21 */          2, 230,  11, 290,  4,   0, 0,  /* UTIL    */
/* 22 */          2, 260,   6, 310, 14,   0, 0,  /* AMU     */
/* 23 */          2, 270,  16, 330,  7,   0, 0,   /* LERF    */
/* 24 */          4,   0,  21,   0,  0,   0, 0,  /* ANFARM          */
/* 25 */          4,  21,   8,   0,  0,   0, 0,  /* ANFARM - ANHVC */
/* 26 */          4,  29,   8,   0,  0,   0, 0,  /* ANFARM - ANVLV */
/* 27 */          4,  37,   8,   0,  0,   0, 0,  /* ANFARM - AN271 */
/* 28 */          4,  60,   3,   0,  0,   0, 0,  /* APFARM           */
/* 29 */          4,  64,  25,   0,  0,   0, 0,  /* APFARM           */
/* 30 */          4, 114,  13,   0,  0,   0, 0,  /* APFARM - APHVC */
```

```
/* 31 */          4, 111,   3,   0,  0,   0, 0,  /* APFARM - APVLV */
/* 32 */          4, 127,   5,   0,  0,   0, 0,  /* APFARM - APVLV */
/* 33 */          4, 136,   6,   0,  0,   0, 0,  /* APFARM - APVLV */
/* 34 */          4,  98,  13,   0,  0,   0, 0,  /* APFARM - AP271 */
/* 35 */          4, 134,   2,   0,  0,   0, 0,  /* APFARM - AP271 */
/* 36 */          4,  63,   1,   0,  0,   0, 0,  /* APFARM - AP102 */
/* 37 */          4,  89,   9,   0,  0,   0, 0,  /* APFARM - AP102 */
/* 38 */          4, 132,   2,   0,  0,   0, 0,  /* APFARM - AP102 */
/* 39 */          4, 142,   1,   0,  0,   0, 0,  /* APFARM - AP102 */
/* 40 */          4, 150,  17,   0,  0,   0, 0,  /* AWFARM          */
/* 41 */          4, 181,   1,   0,  0,   0, 0,  /* AWFARM - AWHVC */
/* 42 */          4, 186,   3,   0,  0,   0, 0,  /* AWFARM - AWHVC */
/* 43 */          4, 167,  11,   0,  0,   0, 0,  /* AWFARM - AWVLV */
/* 44 */          4, 179,   2,   0,  0,   0, 0,  /* AWFARM - AWVLV */
/* 45 */          4, 178,   1,   0,  0,   0, 0,   /* AWFARM - AW271 */
/* 46 */          4, 189,   5,   0,  0,   0, 0,   /* AWFARM - AW271 */
/* 47 */          4, 195,   2,   0,  0,   0, 0,  /* AWFARM - AW271 */
/* 48 */          4, 229,  10,   0,  0,   0, 0,  /* XYZFRM          *//* -mct4+ */
/* 49 */          4, 214,   6,   0,  0,   0, 0,  /* XYZFRM - AFRM  */
/* 50 */          2,  15,   2,   0,  0,   0, 0,  /* A244   - AFRM  */
/* 51 */          4, 222,   4,   0,  0,   0, 0,  /* XYZFRM - AYFRM */
/* 52 */          4, 228,   1,   0,  0,   0, 0,  /* XYZFRM - AYFRM */
/* 53 */          4, 220,   2,   0,  0,   0, 0,  /* XYZFRM - AY,AZ */
/* 54 */          4, 226,   2,   0,  0,   0, 0,  /* XYZFRM - AY,AZ */
/* 55 */          4, 250,  28,  80, 12,   0, 0,  /* BLDNG           */
/* 56 */          2,   0,   0, 300,  3,   0, 0   /* BLDNG           */
                 };

{
        char net, unit;
        int  access, almreg, conreg, dbreg, j, k, vprt, extra, value ;

       access = UNITREL;
       net = 0; almreg = 0; conreg = 0; dbreg = 0; j = 0;
       k = 0; vprt = 0; extra = 0; value = 0;
   /* opens network for communication */
       checkerr (1, mopen() );
       for (k = 0; k < 38; k++)
       {

      /* get value from Sequence Database to check if flag is set */
           checkipn ( k, 2, seqrec[k].pcm, get_vseq(net,seqrec[k].pcm,FG,access,
                                seqrec[k].sunit, seqrec[k].index,&vprt) );
           if(vprt==1)
           {
                      vprt = 0;
               checkerr (3, put_vseq(net,seqrec[k].pcm,FG,access,
                                  seqrec[k].sunit,seqrec[k].index,&vprt) );
               checkerr (4, dbreg =  lnkrg ("DBASE" ) );
               checkerr (5, almreg = lnkrg ("ALMQUE") );
               checkerr (6, conreg = lnkrg ("CONFIG") );

               for ( j = seqrec[k].offset;
                     j < seqrec[k].offset + seqrec[k].bloks;
                     j++ )
               {
                   if (datblk[j].totDIN > 0 )
                       acknow (net, datblk[j].pcm, datblk[j].fstDIN,
                              datblk[j].totDIN, DIN, DIN_ALMCT,
                              almreg, conreg, dbreg);
                   if (datblk[j].totAI > 0 )
                       acknow (net, datblk[j].pcm, datblk[j].fstAI,
                              datblk[j].totAI, AI, AI_ALMCT,
                              almreg, conreg, dbreg);

                   if (datblk[j].totDEV > 0 )
                       acknow (net, datblk[j].pcm, datblk[j].fstDEV,
                              datblk[j].totDEV, DEV, DEV_ALMCT,
                              almreg, conreg, dbreg);
```

```
            } /* end for */

            checkerr ( 11, conreg = unlrg ( conreg ) );
            checkerr ( 12, almreg = unlrg ( almreg ) );
            checkerr ( 13, dbreg =  unlrg ( dbreg ) );

          } /* end if */

       } /* end for loop */
            checkerr (14, get_vseq(net,4,FG,access, 6, 181, &vprt) );
            checkerr (15, put_vseq(net,2,FG,access,25, 183, &vprt) ); /* +mct2- */
            checkerr (16, get_vseq(net,2,FG,access,25, 184, &extra) );        /* +mct2- */
            if ( extra == 1)
               {
            value = 1 ;
            extra = 0 ;
            checkerr ( 4, devcmd( net, 2, DEV, 171, &value));         /* AUDALARM0, +mct2- */
            checkerr (17, put_vseq(net,2,FG,access,25, 184, &extra) );        /* +mct2- */
               }
} /* end main */
}
/*******************/
/* function acknow */
/*******************/
acknow (net, pcm, first, last, type, field, almreg, conreg, dbreg)

        char field, net, pcm, type;
        int  almreg, conreg, dbreg, first, last;
{
        char buff[MAX_DB_RECSIZ],
             dbuff[MAX_DB_RECSIZ],
             epn[EPNSIZ + 1];
        int  adi, devunt, ipn, tot_val, temp_val, err;

       devunt = 4;


   /* loop for ipns */
        for (ipn = first; ipn < first + last; ipn++)
        {

       /* transfers every field in one PCMDB into buffer */
            err = grb(net,pcm,type,ipn,buff);
           if ( err < 0 ) continue ;

       /* transfers one field from buffer into user location */
            checkerr (8, gfb(type,field,&tot_val,buff));

            temp_val = (tot_val & 15);
            if (temp_val > 0)
                {
                if ( ! ( tot_val & 32 ) )
                    {

                /* transfers an acknowledgement */
                     checkipn (9,ipn, pcm, dback(net,pcm,type,ipn));

                /* identifies epn */
                     checkerr (10, idepn(dbreg,epn,net,pcm,type,ipn));
                     sndmsg (net,devunt,almreg,conreg,adi,MSG_TEXT,
                             NONCRIT,"\t\t%s   ALARM ACKNOWLEDGED",epn);

                    } /* end if */

               } /* end if */

       } /* end for loop */

} /* end function acknow */
```

```
/*********************/
/* function checkerr */
/*********************/
checkerr(x,ierr)
        int ierr[],x ;
        {
        int r2,r3 ;
        if (ierr < 0 )
          {
           printf("\n checkerr problem on ACKFLG.C at %d, error = %d\n",x,ierr);
/*         perror(ierr); */
           r2 = lnkrg("CONFIG");
           r3 = lnkrg("ALMQUE");
/*         sndmsg(0,0,r3,r2,3,MSG_TEXT,NOCRIT,
"\n ALARM ACKNOWLEDGE PROGRAM --- ERROR ROUTINE FAILURE AT %d, ERROR %d\n",
                x,ierr); */
           exit();
           }
        }

checkipn(x,y,p,ierr)                    /* number, ipn, error */
        int x, y, p, ierr ;
        {
        int r2,r3 ;
        if (ierr < 0 )
          {
           printf("\n checkipn problem on ackflg.c at %d, error = %d, ipn = %d, pcm %d\n",x,ierr,y,p);
/*         perror(ierr); */
           r2 = lnkrg("CONFIG");
           r3 = lnkrg("ALMQUE");
/*         sndmsg(0,0,r3,r2,3,MSG_TEXT,NOCRIT,
"\n ALARM ACKNOWLEDGE PROGRAM --- ERROR ROUTINE FAILURE AT %d, ERROR %d\n",
                x,ierr); */
           exit();
           }
        }
```

INHA

```
/*              I N H A . C                              VERSION 2
 *
 *      Alarm Inhibit program for graphics 80, 81, 82, 83 and 84.  Scans
 *      EPNs and lists all inhibited EPNs and descriptors.  Batch variables
 *      FTRIP0, FTRIP1, FTRIP2, FTRIP3 display *MORE on GRA080, 81, 82, 83 when
 *      more than 20 found on the page.  FSTOP displays TOO MANY INHIBITED
 *      ALARMS!.... when more than 96 inhibited alarms are found.  FSTOP also
 *      terminates program.  INHA.C is to be run with sequence program
 *      TIMEINHA.SEQ in unit WHOLE0 which fetches the correct time and date.
 *      INHA.C is installed as task INHALM and runs from skid key 53.
 */

/*  Written by : R. N. Kersey            */
/*  Date       : 11-SEP-91               */
/*  Facility   : 242-A Evaporator Bldg. */
/*  Machine    : SMS 11/73               */
/*  D/3 Version: 3.3                     */
/*  Compiler   : WHITESMITH C            */
/*  C .cmd use : @[1,1]CC 'filename'     */
/*  Op. System : RSX 11                  */

/*  MODIFICATION LOG:                                                    */
/*                                                                       */
/*      20-JUL-92       Added description, header.            M. Teats   */
/*                                                                       */
/*      22-JUL-93       Baselined for D/3 Version 3.3.        M. Teats   */
/*                                                                       */
/*  END OF MOD LOG                                                       */

#include <emcstd.h>
#include <d3sys.h>
#include <dba.h>
#include <seqdba.h>
#include <almmsg.h>

        struct s1{
                int   pcm    ;
                char  type   ;
                char  field  ;
                int  tot_ipn ;
                 } ;

        /* number of analog, digital and device EPNS */
        /*      Tank farm signals are first          */

        static struct s1 point[] =
                {
                4,      AI,     AI_ALMCT,       93,
                4,      DIN,    DIN_ALMCT,      270,
                4,      DEV,    DEV_ALMCT,      88,
                2,      AI,     AI_ALMCT,       375,
                2,      DIN,    DIN_ALMCT,      348,
                2,      DEV,    DEV_ALMCT,      211
                        } ;

        static char fil[]="09       " ;
main()

{

   char variabl[EPNSIZ+3], epn[EPNSIZ+1];
   char buff [ MAX_DB_RECSIZ ], net;
   int descrip[(NUMDESC*DESCSIZ)+1] ; int seq_pcm = 2  ;
   int string[NUMDESC*DESCSIZ+1],  bottom[NUMDESC*DESCSIZ+3] ;
   int u_rel = UNITREL ; int sequnit = 0          ;
   int access, conreg, dbreg, i, j, index, jndex;
   int fstart, fstop, ftrip0, ftrip1, ftrip2, ftrip3, count ;
   int ipn, err, tot_val, sunit ;
   unsigned long epoch ;
```

```
   net = 0 ;
   access = UNITREL    ; sunit = 0 ;

   checkerr (1, mopen() );
   checkerr (2, dbreg =  lnkrg ("DBASE" ) );
   checkerr (3, conreg = lnkrg ("CONFIG") );

               /*      clear screen    */

for ( j = 0; j < 100 ; j++ )
        {
        index =  j ;
        checkerr ( 4,  put_vseq ( net, 2, SG, access, sunit, index, fil ) ) ;
        index = 100 + j ;
        checkerr ( 5,  put_vseq ( net, 2, SG, access, sunit, index, fil ) ) ;
        }

        /*              initialize batch variables              */

  fstart = 1 ;fstop = 0 ; ftrip0 = 0 ; ftrip1 = 0 , ftrip2 = 0, ftrip3 = 0;
  put_vseq ( net, 2, FG, access, sunit, 425, &fstart)  ;          /* sets  fstart */
  put_vseq ( net, 2, FG, access, sunit, 426, &fstop )  ;          /* clear fstop  */
  put_vseq ( net, 2, FG, access, sunit, 427, &ftrip0)  ;          /* clear ftrip0 */
  put_vseq ( net, 2, FG, access, sunit, 428, &ftrip1)  ;          /* clear ftrip1 */
  put_vseq ( net, 2, FG, access, sunit, 429, &ftrip1)  ;          /* clear ftrip2 */
  put_vseq ( net, 2, FG, access, sunit, 430, &ftrip1)  ;          /* clear ftrip3 */
  fstart = 0 ;fstop = 1 ; ftrip0 = 1 ; ftrip1 = 1 ; ftrip2 = 1; ftrip3 = 1;

  sleep(1) ;            /* give sequence program chance to run */

  index = 0 ;
  jndex = 100;
  count = 0 ;


for (i = 0; i < 6 ; i++)
{

   for (ipn = 0 ; ipn < point[i].tot_ipn ; ipn++ )
    {

        err = grb ( net, point[i].pcm, point[i].type, ipn, buff ) ;
        if ( err < 0 ) continue ;
        checkerr (6, gfb ( point[i].type, point[i].field, &tot_val, buff ) ) ;
        if ( tot_val & 16)                                                      /* if alm
inhibited...*/
          {

         err =  idepn ( dbreg, epn, net, point[i].pcm, point[i].type, ipn ) ;        /* get EPN and
descriptor */
          if (err < 0) continue ;
          cpystr(variabl,"09", epn);
          checkerr ( 8,  put_vseq ( net, seq_pcm, SG, u_rel, sequnit, index++, variabl)) ;
          sleep(1) ;
descrip[ squeeze ( descrip, desc ( dbreg, descrip, net, point[i].pcm, point[i].type, ipn), NULL)] = 0 ;
          cpystr ( bottom,"09", descrip) ;
          checkerr ( 10,  put_vseq ( net, seq_pcm, SG, u_rel, sequnit,jndex++, bottom));
          if (  count > 20 ) put_vseq ( net, 2, FG, access, sunit, 427, &ftrip0 )  ;       /* more on page
0 */
          if (  count > 40 ) put_vseq ( net, 2, FG, access, sunit, 428, &ftrip1 )  ;    /* more on page 1 */

          if (  count > 60 ) put_vseq ( net, 2, FG, access, sunit, 429, &ftrip2 )  ;    /* more on page 2 */
          if (  count > 80 ) put_vseq ( net, 2, FG, access, sunit, 430, &ftrip3 )  ;    /* more on poge 3 */
          if (++count > 99 )
             {                                                                  /* exceeds number of strings
*/
                put_vseq ( net, 2, FG, access, sunit, 425, &fstart )  ; /* clears  'running' message
*/
```

```
                put_vseq ( net, 2, FG, access, sunit, 426, &fstop )   ; /* lights 'too many inhibited
alarms' */
                exit();
             }          /* count   */
          }             /* tot_val */
  }                     /* ipn     */
}                       /* i       */

        put_vseq ( net, 2, FG, access, sunit, 425, &fstart )  ; /* clears  'running' message if normal term.
*/

}                       /* main  */


checkerr(x,ierr)
        int ierr,x ;
        {
        int r2,r3 ;
        if (ierr < 0 )
          {
           printf("\n checkerr problem on test.c at %d, error = %d\n",x,ierr);
           r2 = lnkrg("CONFIG");
           r3 = lnkrg("ALMQUE");
           sndmsg(0,0,r3,r2,3,MSG_TEXT,CRIT,
"\n ALARM INHIBIT TEST PROGRAM --- ERROR ROUTINE FAILURE AT %d, ERROR %d\n",
                  x,ierr);
           exit();
           }
        }

checkipn(x,y,ierr)
        int ierr,x,y ;
        {
        int r2,r3 ;
        if (ierr < 0 )
          {
           printf("\n checkerr problem on ackflg.c at %d, error = %d, ipn = %d\n",x,ierr,y);
           r2 = lnkrg("CONFIG");
           r3 = lnkrg("ALMQUE");
           sndmsg(0,0,r3,r2,3,MSG_TEXT,CRIT,
"\n ALARM ACKNOWLEDGE PROGRAM --- ERROR ROUTINE FAILURE AT %d, ERROR %d\n",
                  x,ierr);
           exit();
           }
        }
```

UKF34

```
/*              U K F 3 4 . C                                   VERSION 3
 *
 *
 *
 *  General Function 34 - To remove inhibited alarms for Plant startup.
 *                        Reversal of UKF43.  UKF34 is only loaded to the
 *                        CDCM, as are UKF43 (alarm inhibiting) and UKF14
 *                        (alarm regereration).
 *
 *
 *      Written by : M. C. Teats
 *      Date       : 03-08-93
 *      Facility   : 242-A Evaporator Bldg. */
/*  Modified by: John Hayden T.I.        6.3-2 Upgrade    */
/*  Facility   : 242-A Evaporator Bldg.          */
/*  Machine    : VAX 4104-A                                 */
/*  Compiler   : DEC VAX C V3.3                     JH01     */
/*  C .cmd use : SET DEF D3SOURCE::[261,10]         JH01     */
/*  C .cmd use : BLD UKF14 UKF14.C -L -MOV          JH01     */
/*  Op. System : VAX/VMS 5.5-2AH          JH01      */
/*  D/3 Version: 6.3-2   BASELINE                   JH01     */
/*  #INCLUDES CHANGED                               JH01     */
/*  PARAMATERS TO FUNCTIONS CHANGED                 JH01     */
/*  PRAGAMA FOR DLIST DECLARATIONS, INITIALIZE      JH01     */
/*  SLEEP NOW SECONDS INSTEAD OF TICKS              JH01     */
/*  %P NOT %S, PUTFMT-> PRINTF,%I->%D               JH01     */
/*
 *
 *  MODIFICATION LOG:
 *
 *      08-MAR-93       Define program for D/3 version 3.3              M. Teats
 *
 *      07-MAY-93       Modify ipns per CCN# 93MAR25-01 (search         M. Teats
 *                      MCT1).
 *
 *      05-JAN-95       Rebase lined the program for upgrade to         J. Berger
 *                      version 6.3 of the D/3 software. Added
 *                      additional EPNs to the inhibit list.
 *                      This will be documented on the rebaselined
 *                      documentation for the 6.3 upgrade.
 *
 *  END OF MOD LOG
 */

#include <emcstd.h>
#include <d3sys.h>
#include <unitdb.h>
#include <grinnell.h>
#include <dbatypes.h>
#include <dbaai.h>
#include <dbadin.h>
#include <dbadev.h>                     /*  jfb1  */
#include <digdef.h>
#include <conkey.h>
#include <conf.h>
#include <err.h>
#include <keyequ.h>
#include <func.h>
#include <seqdba.h>
#include <almmsg.h>

#define UDF34           34
#define DONE            1

#pragma nostandard
PUBLIC    int     dlrd    =0;           /* DLIST descriptor                 */
PUBLIC    struct dlist    dlist   =0;   /* DLIST structure                  */
PUBLIC    struct gcb      gcb     =0;   /* Struct for Grinnel Control Block */
PUBLIC    short  con      = 0;          /* console numbwer */
#pragma standard
```

```
/*
 *      DEFINE STRUCTURE FOR NODE STATUS:
 */
        struct unitstat ustat    =0;     /* For Micro Status determination        */
/*
 *      DEFINE STRUCTURES FOR EPN's TO REMOVE INHIBIT FOR:
 */
        struct s1
        {
            char pcm,                    /*  PCM unit number      */
                 dbt;                    /*  DBASE type           */
            int  sbi;                    /*  Scan block ipn       */
        };

        static struct s1 inha [] =
                {
/* k          pcm  dbt  sbi                 epn          */
/* --         ---  ---  ---                 ---------    */
/* 0  */        2,  AI,  45,          /*  pi-ca1-11    */
/* 1  */        2,  AI,  46,          /*  pic-ca1-7    */
/* 2  */        2,  AI,  53,          /*  lic-ca1-1    */
/* 3  */        2,  AI,  57,          /*  lic-ca1-2    */
/* 4  */        2,  AI,  63,          /*  li-ca1-3     */
/* 5  */        2,  AI,  68,          /*  fic-ca1-6    */
/* 6  */        2,  AI,  81,          /*  FI-CA1-7     */  /* JFB1 */
/* 7  */        2,  AI,  83,          /*  vi-pb1-2a    */
/* 8  */        2,  AI,  84,          /*  vi-pb1-1a    */
/* 9  */        2,  AI,  85,          /*  vi-pb1-3a    */
/* 10 */        2,  AI,  86,          /*  fi-ca1-1     */
/* 11 */        2,  AI,  87,          /*  pi-ca1-9     */
/* 12 */        2,  AI,  89,          /*  fi-ca1-3     */
/* 13 */        2,  AI, 102,          /*  sic-pb2-1    */  /* JFB1 */
/* 14 */        2,  AI, 103,          /*  vi-pb2-1a    */
/* 15 */        2,  AI, 106,          /*  fi-ca1-2     */
/* 16 */        2,  AI, 107,          /*  pi-ca1-10    */
/* 17 */        2,  AI, 110,          /*  fic-ca1-4    */
/* 18 */        2,  AI, 130,          /*  pi-ea1-14    */  /* JFB1 */
/* 19 */        2,  AI, 131,          /*  fic-ea1-1    */
/* 20 */        2,  AI, 153,          /*  pi-ec1-4     */  /* MCT1 */
/* 21 */        2,  AI, 154,          /*  pi-ec2-2     */  /* MCT1 */
/* 22 */        2,  AI, 162,          /*  fic-ec3-1    */  /* MCT1 */
/* 23 */        2,  AI, 197,          /*  pic-ec1-2    */
/* 24 */        2,  AI, 198,          /*  pi-ec1-9     */  /* JFB1 */
/* 25 */        2, DIN,  60,          /*  psh-ca111    */
/* 26 */        2, DIN,  87,          /*  TSH-PB2-1    */  /* JFB1 */
/* 27 */        2, DIN,  90,          /*  SXS-PB2-1    */  /* JFB1 */
/* 28 */        2, DIN,  91,          /*  YS-VFD-1     */  /* JFB1 */
/* 29 */        2, DIN,  92,          /*  SSL-PB2-1    */  /* JFB1 */
/* 30 */        2, DIN, 107,          /*  ZS-EA1-1C    */  /* JFB1 */
/* 31 */        2, DEV, 102,          /*  RC2-PIG      */  /* JFB1 */
/* 32 */        2, DEV, 133,          /*  RC3-PIG      */  /* JFB1 */
/* 33 */        2, DEV, 143,          /*  RC1-PIG      */  /* JFB1 */
                };
    static char type = 0;             /*  For PCM processor type, status              */

/*************************
 *                       *
 *  M A I N  ( UKF34 )   *
 *                       *
 *************************/

main( )
{
    char net,                         /*  Network number                              */
         uname[ UNAMSIZ+1 ],          /*  Unit name buffer                            */
         epn[ EPNSIZ+1 ],             /*  EPN buffer                                  */
         status[36],                  /*  Temp string for status                      */
         pcm_node[64],                /*  Table of PCM's completed or failed          */
         backup,                      /*  PCM backup unit number                      */
         anydone,                     /*  Any Nodes done ?                            */
```

```
        buff[ MAX_DB_RECSIZ ],          /*  Largest db block                    */
        field;                          /*  Alarm count field                   */
    int dbreg,                          /*  DBASE region id                     */
        creg,                           /*  Config region id                    */
        areg,                           /*  Alarm que region id                 */
        unit,                           /*  PCM unit number                     */
        npcm,                           /*  Maximum PCM unit number = 3         */
        value,                          /*  Alarm count field's bit value       */
        k,                              /*  De-Inhibit for-loop counter         */
        err,                            /*  Error code for dbase access calls   */
        i;                              /*  Miscellaneous variables             */
    net  = 0;
    npcm = 3;

        con = gtnum();
        if( con < 0 || con > 3 )
                {
                con = 0;
                printf("\n-- UKF34 - ALARM INHIBIT REMOVAL ( Invoked - Interactive Mode )\n" );
                }
        else                            /*  Else - Invoked by Gen Func 34 at console    */
                {
                printf("\n-- UKF34 - ALARM INHIBIT REMOVAL Invoked from Console %d \n", con );
                wrt_console( con );
                }                       /*  End else from CONSOLE                       */

/*
 *      FIRST, LINK TO THE CONFIG TABLE, DATABASE, AND ALARM QUE REGIONS:
 */
if( (creg = lnkrg("config")) < 0 )                              /*  Link to config      */
        {
        printf("\n -- UKF34 -- CAN'T LINK TO CONFIG - TASK EXITING --\n");
        exit();
        }
if( (dbreg = lnkrg("DBASE")) < 0 )                              /*  Link to database    */
        {
        printf("\n -- UKF34 -- CAN'T LINK TO DBASE - TASK EXITING --\n");
        unlrg( creg );
        exit();
        }
if( (areg = lnkrg("ALMQUE")) < 0 )                              /*  Link to alarm que   */
        {
        printf("\n -- UKF34 -- CAN'T LINK TO ALMQUE - TASK EXITING --\n");
        unlrg( creg );
        unlrg( dbreg );
        exit();
        }
/*
 *      OPEN A COMMUNICATION CHANNEL TO THE PCM:
 */
if ( net_saps(FALSE,TRUE) < 0 )
        {
        printf("\n -- UKF34 -- CAN'T OPEN COMMUNICATIONS, MOPEN() - TASK EXITING\n");
        unlrg( creg );
        unlrg( dbreg );
        unlrg( areg );
        exit();
        }
/*
 *      INITIALIZATION:
 */
    anydone = 0;
    unit = -1;
    for( i=0; i<sizeof(pcm_node); i++ )
        pcm_node[i] = 0;
/*
 *      MAIN LOOP -
 *
 *      WHILE PCM0A OR PCM0B (unit 2 or 3) ON NETWORK 0:
 */
```

```
while( ++unit <= npcm )
{
/*
 *  GET UNIT NAME FOR NEXT PCM:
 */
    for( type=0; type != UPCM; unit++ )
        {
        if( unit > npcm ) goto quit;
        fill( uname,sizeof(uname),'\0' );

        err = idpcm( creg, uname, net, unit );
        if( err < 0 )
            {
            if( err == ER_SND || err == ER_UNF || err == ER_UNL )
                continue;
            printf("UKF34 - CAN'T GET UNIT NAME FOR UNIT #%d, err=%d\n",unit,err) ;
            goto quit;
            }
        if( err = pcmid( creg, uname, &net, &unit, &type, &backup) < 0 )
            {
            printf("UKF34 - CAN'T GET UNIT TYPE FOR UNIT #%d, err=%d\n",unit,err);
            goto quit;
            }
        if( type == UPCM )
            {
            /*
             *   HAS THE PRIMARY UNIT ALREADY BEEN DONE ?
             */
            if( (unit>backup) && (backup!=-1) && (pcm_node[backup] == DONE) )
                {
                printf( "UKF34 - %s IS BACKUP UNIT (BYPASSED) \n", uname );
                type = 0;
                continue;
                }
            err = gsstat( creg, net, unit, &ustat );

            if( ustat.us_pro != STOK )
                type = 0;
            /*
             *      PROCESSOR STATUS:
             */
            switch( ustat.us_pro )
                    {
                    case( STOFF ):  cpystr( status, "offline", 0 );
                            break;
                    case( STLOD ):  cpystr( status, "load request", 0 );
                            break;
                    case( STIDL ):  cpystr( status, "idle", 0 );
                            break;
                    case( STOK ):   cpystr( status, "running", 0 );
                            break;
                    default:        cpystr( status, "UNKNOWN", 0 );
                            break;
                    }
            if( ustat.us_pro != STOK )
                 printf( "UKF34 - %s IS %s  (BYPASSED) \n", uname, status );
            else printf( "UKF34 - %s IS %s \n", uname, status );
            /*
             *      END PROCESSOR STATUS
             */
            }                           /*  End If  ( type UPCM )       */
    }                                   /*  End For ( unit++ ) loop     */
    unit--;
    anydone = 1;
    if( backup == -1 ) backup = unit;
    pcm_node[ unit ] = pcm_node[ backup ] = DONE;
/*
 *      EPN INHIBIT REMOVAL LOOP -
 *
 *      NOW WE ARE READY TO REMOVE INHIBITED ALARMS:
```

```
*               err = idepn (rid, &epn, net, unit, dbtype, ipn)
*               err = grb (net, unit, dbtype , ipn, &buff)
*               err = gfb (dbtype, field, &value, &buff)
*               err = pfb (dbtype, field, &value, &buff)
*               err = prb (net, unit, dbtype , ipn, &buff)
*/
        for (k = 0; k < 34; k++)          /* JFB1 */
        {
        field = AI_ALMCT;
        if (k > 24)                               /* JFB1 */
                {
                field = DIN_ALMCT;
                }
        if (k > 30)                               /* JFB1 */
                {                                 /* JFB1 */
                field = DEV_ALMCT;                /* JFB1 */
                }                                 /* JFB1 */
        err = idepn(dbreg, epn, net, inha[k].pcm, inha[k].dbt, inha[k].sbi);
                if(err < 0)
                {
                printf("\n --UKF34 IDEPN--ERROR = %d\n",err);
                perror(&err);
                unlrg( creg );
                unlrg( dbreg );
                unlrg( areg );
                exit();
                }
        err = grb(net, inha[k].pcm, inha[k].dbt, inha[k].sbi, buff);
                if(err < 0)
                {
                printf("\n --UKF34 %s GRB--ERROR = %d\n", epn, err);
                perror(&err);
                unlrg( creg );
                unlrg( dbreg );
                unlrg( areg );
                exit();
                }
        err = gfb(inha[k].dbt, field, &value, buff);
                if(err < 0)
                {
                printf("\n -- UKF34 %s GFB--ERROR = %d\n", epn, err);
                perror(&err);
                unlrg( creg );
                unlrg( dbreg );
                unlrg( areg );
                exit();
                }
        /*
         *      CHECK IF INHIBITED.  IF TRUE, REMOVE IT AND SEND MESSAGE!
         */
        if ( value = 16 )                                       /*  If inhibited        */
                {
                value = 32;                                     /*  Remove inhibit      */
                err = pfb(inha[k].dbt, field, &value, buff);
                        if(err < 0)
                        {
                        printf("\n -- UKF34 %s PFB--ERROR = %d\n", epn, err);
                        perror(&err);
                        unlrg( creg );
                        unlrg( dbreg );
                        unlrg( areg );
                        exit();
                        }
                sndmsg (0,0,areg,creg,3,MSG_TEXT,NONCRIT,"--- UKF34 - INHIBIT REMOVED FOR  %s",epn);
                sleep (2);
                }                                       /*  End If              */
        /*
         *      GET OUT AND GO TO NEXT EPN:
         */
        err = prb(net, inha[k].pcm, inha[k].dbt, inha[k].sbi, buff);
```

```
                if(err < 0)
                {
                printf("\n --UKF34 %s PRB--ERROR = %d\n", epn, err);
                perror(&err);
                unlrg( creg );
                unlrg( dbreg );
                unlrg( areg );
                exit();
                }
        sleep (1);                                  /*  Take a rest                 */
        }                                           /*  End removal for loop, k++   */
    }                                               /*  END while <= npcm           */
/*
 *      UNLINK REGIONS AND GET OUT:
 */
quit:
        unlrg( creg );
        unlrg( dbreg );
        unlrg( areg );
        if( anydone )
                {
                printf( "UKF34 - INHIBIT REMOVAL FOR 242-A STARTUP COMPLETED \n" );
                }
        else
                printf( "UKF34 - INHIBIT REMOVAL WARNING - ALL PCM NODES OFFLINE \n" );
        return(1);

}                                                   /*  End main  */

/*************************
 *                       *
 *    E N D   M A I N    *
 *                       *
 *************************/

/*
 *      W R T _ C O N S O L E (   )
 *
 *      Draw the General Function 34 - Display
 */
int wrt_console( con )
int con;
{
    int    kbfd;                            /*  Key board fd          */
    int    kboutfd;                         /*  Keyboard output fd    */
    int    err;                             /*  Error code            */
    static char dlname[] = "DLISTxx";

        dlname[5 + itob ( &dlname[5], con ,10 ) ] = 0;
        if((dlrd = lnkrg(dlname)) < 0)
            {
            printf("UKF34 - DLIST LINK FAILURE %d\n",dlrd);
            unlrg(dlrd);
            exit();
            }
        if((err = ginit(&gcb, con)) < 0 )
            {
            printf("UKF34 - CANT ACCESS GRINNELL, ERR %d\n",err );
            unlrg(dlrd);
            exit();
            }
        dlist.act = DMUSER + UDF34;
        putrg(dlrd, &dlist.act, OFFSET(dlist,act), sizeof(dlist.act) );
/*
 *      C L E A R  the  O C M :
 */
        gsetup(&gcb,BLACK);
        erase(&gcb);
        gflush(&gcb);
```

```
        /* O C M   G R A P H I C S :  */

        gsetup(&gcb,RED);
        cursor(&gcb,10,425);     /*      S C R E E N   F O R M A T    */
        horiz( &gcb,440,3);      /*  .______________________________. */
        cursor(&gcb,447,425);    /*  |   General Function  34       | */
        vert(  &gcb,-105,3);     /*  |------------------------------| */
        cursor(&gcb,449,320);    /*  |   Alarm Inhibit Removal      | */
        horiz( &gcb,-439,3);     /*  |                              | */
        cursor(&gcb,10,320);     /*  |   EPN's printing at typer.   | */
        vert(  &gcb,105,3);      /*  |______________________________| */
        cursor(&gcb,10,390);
        horiz( &gcb,440,3);

        /* O C M   T E X T :  */

        gsetup(&gcb, WHITE );
        dublchar(&gcb);
        cursor(&gcb,HCHAR*3,VCHAR*40);
        g_puts( &gcb, "GENERAL FUNCTION 34" );
        gflush(&gcb);
        gsetup( &gcb, YELLOW );
        widechar(&gcb);
        cursor(&gcb,HCHAR*3,VCHAR*36);
        g_puts( &gcb, "REMOVING INHA ALARMS FOR STARTUP   " );
        gflush(&gcb);
        cursor(&gcb,HCHAR*3,VCHAR*34);
        g_puts( &gcb, "EPN'S WILL BE PRINTED AT TYPER " );
        gflush(&gcb);
        gsetup(&gcb, WHITE );
        cursor(&gcb,HCHAR*12,VCHAR*31);
        g_puts( &gcb, "JH -- TEXAS INSTRUMENTS D/3 6.3-2 UPGRADE SPECIAL" );
        gflush(&gcb);
/*
 *      RELEASE THE CONSOLE AND DLIST REGION:
 */
        dlist.ukf = -1;
        dlist.act = 0;
        putrg(dlrd,&dlist.ukf,OFFSET(dlist,ukf),sizeof(dlist.ukf));
        putrg(dlrd,&dlist.act,OFFSET(dlist,act),sizeof(dlist.act));
        unlrg( dlrd );
        return(1);
}                                   /*  End wrt_console() */
```

UKF43

```
/*              U K F 4 3 . C                                   VERSION 3
 *
 *
 *
 *  General Function 43 - For standard alarm inhibiting during Plant
 *                        shutdown.  UKF43 is loaded only to the CDCM,
 *                        as are UKF34 (de-inhibiting), and UKF14 (alarm
 *                        regeneration).  Due to the way the ALMQMG task
 *                        works, the following user key functions must
 *                        be run after GF 43 in the order presented:
 *                        GF 44 (to clear the alarm summary pages),
 *                        GF 14 (to regenerate alarms).  Since UKF43
 *                        spawns the mcr command to reset the alarm
 *                        summary pages on the CDCM, GF 44 only needs
 *                        to be run at one of the DCM0 consoles.
 *
 *
 *      Written by : M. C. Teats
 *      Date       : 03-08-93
 *      Facility   : 242-A Evaporator Bldg.
 *      Run on Unit: CDCM only
 */
/*  Modified by: John Hayden T.I.        6.3-2 Upgrade    */
/*  Facility   : 242-A Evaporator Bldg.          */
/*  Machine    : VAX 4104-A                                */
/*  Compiler   : DEC VAX C V3.3                    JH01    */
/*  C .cmd use : SET DEF D3SOURCE::[261,10]        JH01    */
/*  C .cmd use : BLD UKF14 UKF14.C -L -MOV         JH01    */
/*  Op. System : VAX/VMS 5.5-2AH          JH01    */
/*  D/3 Version: 6.3-2   BASELINE                  JH01    */
/*  #INCLUDES CHANGED                              JH01    */
/*  PARAMATERS TO FUNCTIONS CHANGED                JH01    */
/*  PRAGAMA FOR DLIST DECLARATIONS, INITIALIZE     JH01    */
/*  SLEEP NOW SECONDS INSTEAD OF TICKS             JH01    */
/*  %P NOT %S, PUTFMT-> PRINTF,%I->%D              JH01    */
/*
 *
 *  MODIFICATION LOG:
 *
 *      08-MAR-93       Define program for D/3 version 3.3              M. Teats
 *
 *      07-MAY-93       Modify ipns per CCN# 93MAR25-01 (search         M. Teats
 *                      MCT1).
 *
 *      05-JAN-95       Rebaselined the program for version             J. Berger
 *                      6.3 upgrade of the D/3 software. Added
 *                      aditional EPN to the list of EPNs to be
 *                      inhibited. See rebaseline documents for
 *                      the 6.3 upgrade.
 *
 *  END OF MOD LOG
 */

#include <emcstd.h>
#include <d3sys.h>
#include <unitdb.h>
#include <grinnell.h>
#include <dbatypes.h>
#include <dbaai.h>
#include <dbadin.h>
#include <dbadev.h>                     /*  jfb1  */
#include <digdef.h>
#include <conkey.h>
#include <conf.h>
#include <err.h>
#include <keyequ.h>
#include <func.h>
#include <seqdba.h>
#include <almmsg.h>
```

```
#define UDF43           43
#define DONE            1

#pragma nostandard
PUBLIC    int     dlrd    = 0;                    /* DLIST descriptor                 */
PUBLIC    struct dlist    dlist   = 0;    /* DLIST structure                          */
PUBLIC    struct gcb      gcb     = 0;    /* Struct for Grinnel Control Block         */
PUBLIC  short con=0;
#pragma standard

/*
 *      DEFINE STRUCTURE FOR NODE STATUS:
 */
        struct unitstat ustat    =0;     /* For Micro Status determination           */
/*
 *      DEFINE STRUCTURES FOR EPN's TO INHIBIT:
 */
        struct s1
        {
            char pcm,                   /*  PCM unit number      */
                 dbt;                   /*  DBASE type           */
            int  sbi;                   /*  Scan block ipn       */
        };

        static struct s1 inha [] =
                {
/* k          pcm  dbt  sbi                epn          */
/* --         ---  ---  ---                ---------    */
/* 0  */        2,  AI,  45,              /*  pi-ca1-11   */
/* 1  */        2,  AI,  46,              /*  pic-ca1-7   */
/* 2  */        2,  AI,  53,              /*  lic-ca1-1   */
/* 3  */        2,  AI,  57,              /*  lic-ca1-2   */
/* 4  */        2,  AI,  63,              /*  li-ca1-3    */
/* 5  */        2,  AI,  68,              /*  fic-ca1-6   */
/* 6  */        2,  AI,  81,              /*  fi-ca1-7    */   /* jfb1 */
/* 7  */        2,  AI,  83,              /*  vi-pb1-2a   */
/* 8  */        2,  AI,  84,              /*  vi-pb1-1a   */
/* 9  */        2,  AI,  85,              /*  vi-pb1-3a   */
/* 10 */        2,  AI,  86,              /*  fi-ca1-1    */
/* 11 */        2,  AI,  87,              /*  pi-ca1-9    */
/* 12 */        2,  AI,  89,              /*  fi-ca1-3    */
/* 13 */        2,  AI, 102,              /*  sic-pb2-1   */  /* JFB1 */
/* 14 */        2,  AI, 103,              /*  vi-pb2-1a   */
/* 15 */        2,  AI, 106,              /*  fi-ca1-2    */
/* 16 */        2,  AI, 107,              /*  pi-ca1-10   */
/* 17 */        2,  AI, 110,              /*  fic-ca1-4   */
/* 18 */        2,  AI, 130,              /*  pi-ea1-14   */  /* JFB1 */
/* 19 */        2,  AI, 131,              /*  fic-ea1-1   */
/* 20 */        2,  AI, 153,              /*  pi-ec1-4    */  /* MCT1 */
/* 21 */        2,  AI, 154,              /*  pi-ec2-2    */  /* MCT1 */
/* 22 */        2,  AI, 162,              /*  fic-ec3-1   */  /* MCT1 */
/* 23 */        2,  AI, 197,              /*  pic-ec1-2   */
/* 24 */        2,  AI, 198,              /*  pi-ec1-9    */  /* JFB1 */
/* 25 */        2, DIN,  60,              /*  psh-ca111   */
/* 26 */        2, DIN,  87,              /*  TSH-PB2-1   */  /* JFB1 */
/* 27 */        2, DIN,  90,              /*  SXS-PB2-1   */  /* JFB1 */
/* 28 */        2, DIN,  91,              /*  YS-VFD-1    */  /* JFB1 */
/* 29 */        2, DIN,  92,              /*  SSL-PB2-1   */  /* JFB1 */
/* 30 */        2, DIN, 107,              /*  ZS-EA1-1C   */  /* JFB1 */
/* 31 */        2, DEV, 102,              /*  RC2-PIG     */  /* JFB1 */
/* 32 */        2, DEV, 133,              /*  RC3-PIG     */  /* JFB1 */
/* 33 */        2, DEV, 143,              /*  RC1-PIG     */  /* JFB1 */
                };
    static char type = 0;                 /*  For PCM processor type, status      */

/*************************
 *                       *
 *  M A I N  ( UKF43 )   *
 *                       *
 *************************/
```

```
main( )
{
    char net,                           /*  Network number                      */
         uname[ UNAMSIZ+1 ],            /*  Unit name buffer                    */
         epn[ EPNSIZ+1 ],               /*  EPN buffer                          */
         status[36],                    /*  Temp string for status              */
         pcm_node[64],                  /*  Table of PCM's completed or failed  */
         backup,                        /*  PCM backup unit number              */
         anydone,                       /*  Any Nodes done ?                    */
         buff[ MAX_DB_RECSIZ ],         /*  Largest db block                    */
         field;                         /*  Alarm count field                   */
    int  dbreg,                         /*  DBASE region id                     */
         creg,                          /*  Config region id                    */
         areg,                          /*  Alarm que region id                 */
         unit,                          /*  PCM unit number                     */
         npcm,                          /*  Maximum PCM unit number = 3         */
         value,                         /*  Alarm count field's bit value       */
         k,                             /*  For loop counter                    */
         err,                           /*  Error code for dbase access calls   */
         con,                           /*  Console number                      */
         i;                             /*  Miscellaneous variables             */
    net  = 0;
    npcm = 3;

        con = gtnum();
        if( con < 0 || con > 3 )
                {
                con = 0;
                printf("\n-- UKF43 - ALARM INHIBIT ( Invoked - Interactive Mode )\n" );
                }
        else                            /*  Else - Invoked by Gen Func 43 at console    */
                {
                printf("\n-- UKF43 - ALARM INHIBIT Invoked from Console %d \n", con );
                wrt_console( con );
                }                       /*  End else from CONSOLE                       */
/*
 *      SECOND, OPEN A COMMUNICATION CHANNEL TO THE PCM:
 */
if ( net_saps(FALSE,TRUE) < 0 )
        {
        printf("\n -- UKF43 -- CAN'T OPEN COMMUNICATIONS, net_saps- TASK EXITING\n");
        unlrg( creg );
        unlrg( dbreg );
        unlrg( areg );
        exit();
        }
/*
 *      FIRST, LINK TO THE CONFIG TABLE, DATABASE, AND ALARM QUE REGIONS:
 */
if( (creg = lnkrg("CONFIG")) < 0 )                          /*  Link to config      */
        {
        printf("\n -- UKF43 -- CAN'T LINK TO CONFIG - TASK EXITING --\n");
        exit();
        }
if( (dbreg = lnkrg("DBASE")) < 0 )                          /*  Link to database    */
        {
        printf("\n -- UKF43 -- CAN'T LINK TO DBASE - TASK EXITING --\n");
        unlrg( creg );
        exit();
        }
if( (areg = lnkrg("ALMQUE")) < 0 )                          /*  Link to alarm que   */
        {
        printf("\n -- UKF43 -- CAN'T LINK TO ALMQUE - TASK EXITING --\n");
        unlrg( creg );
        unlrg( dbreg );
        exit();
        }
/*
 *      INITIALIZATION:
 */
```

```
    anydone = 0;
    unit = -1;
    for( i=0; i<sizeof(pcm_node); i++ )
        pcm_node[i] = 0;
/*
 *      MAIN LOOP -
 *
 *      WHILE PCM0A OR PCM0B (unit 2 or 3) ON NETWORK 0:
 */
    while( ++unit <= npcm )
    {
    /*
     *  GET UNIT NAME FOR NEXT PCM:
     */
        for( type=0; type != UPCM; unit++ )
            {
            if( unit > npcm ) goto quit;
            fill( uname,sizeof(uname),'\0' );

            err = idpcm( creg, uname, net, unit );
            if( err < 0 )
                {
                if( err == ER_SND || err == ER_UNF || err == ER_UNL )
                    continue;
                printf("UKF43 - CAN'T GET UNIT NAME FOR UNIT %d, err=%d\n",unit,err) ;
                goto quit;
                }
            if( err = pcmid( creg, uname, &net, &unit, &type, &backup) < 0 )
                {
                printf("UKF43 - CAN'T GET UNIT TYPE FOR UNIT %d, err=%d\n",unit,err);
                goto quit;
                }
            if( type == UPCM )
                {
                /*
                 *   HAS THE PRIMARY UNIT ALREADY BEEN DONE ?
                 */
                if( (unit>backup) && (backup!=-1) && (pcm_node[backup] == DONE) )
                    {
                    printf( "UKF43 - %s IS BACKUP UNIT (BYPASSED) \n", uname );
                    type = 0;
                    continue;
                    }
                err = gsstat( creg, net, unit, &ustat );

                if( ustat.us_pro != STOK )
                    type = 0;
                /*
                 *      PROCESSOR STATUS:
                 */
                switch( ustat.us_pro )
                        {
                        case( STOFF ):  cpystr( status, "offline", 0 );
                                break;
                        case( STLOD ):  cpystr( status, "load request", 0 );
                                break;
                        case( STIDL ):  cpystr( status, "idle", 0 );
                                break;
                        case( STOK ):   cpystr( status, "running", 0 );
                                break;
                        default:        cpystr( status, "UNKNOWN", 0 );
                                break;
                        }
                if( ustat.us_pro != STOK )
                     printf( "UKF43 - %s IS %s  (BYPASSED) \n", uname, status );
                else printf( "UKF43 - %s IS %s \n", uname, status );
                /*
                 *      END PROCESSOR STATUS
                 */
                }                       /*  End If  ( type UPCM )       */
```

```
        }                               /*  End For ( unit++ ) loop      */
        unit--;
        anydone = 1;
        if( backup == -1 ) backup = unit;
        pcm_node[ unit ] = pcm_node[ backup ] = DONE;
/*
 *      EPN INHIBIT LOOP -
 *
 *      NOW WE ARE READY TO INHIBIT ALARMS:
 *              err = idepn (rid, &epn, net, unit, dbtype, ipn)
 *              err = grb (net, unit, dbtype , ipn, &buff)
 *              err = gfb (dbtype, field, &value, &buff)
 *              err = pfb (dbtype, field, &value, &buff)
 *              err = prb (net, unit, dbtype , ipn, &buff)
 */
        for (k = 0; k < 34; k++)        /* JFB1 */
        {
        field = AI_ALMCT;
        if (k > 24)                             /* JFB1 */
                {
                field = DIN_ALMCT;
                }
        if (k > 30)                             /* JFB1 */
                {                               /* JFB1 */
                field = DEV_ALMCT;              /* JFB1 */
                }                               /* JFB1 */
        err = idepn(dbreg, epn, net, inha[k].pcm, inha[k].dbt, inha[k].sbi);
                if(err < 0)
                {
                printf("\n --UKF43 IDEPN--ERROR = %d\n",err);

                unlrg( creg );
                unlrg( dbreg );
                unlrg( areg );
                exit();
                }
        err = grb(net, inha[k].pcm, inha[k].dbt, inha[k].sbi, buff);
                if(err < 0)
                {
                printf("\n --UKF43 %s GRB--ERROR = %d\n", epn, err);

                unlrg( creg );
                unlrg( dbreg );
                unlrg( areg );
                exit();
                }
        err = gfb(inha[k].dbt, field, &value, buff);
                if(err < 0)
                {
                printf("\n -- UKF43 %s GFB--ERROR = %d\n", epn, err);

                unlrg( creg );
                unlrg( dbreg );
                unlrg( areg );
                exit();
                }
        /*
         *      CHECK IF INHIBITED.  IF NOT, DO IT AND SEND MESSAGE!
         */
        if ( value != 16 )                                      /*  If not inhibited    */
                {
                value = 16;                                     /*  Set to inhibit      */
                err = pfb(inha[k].dbt, field, &value, buff);
                        if(err < 0)
                        {
                        printf("\n -- UKF43 %s PFB--ERROR = %d\n", epn, err);

                        unlrg( creg );
                        unlrg( dbreg );
                        unlrg( areg );
```

```
                        exit();
                        }
                sndmsg (0,0,areg,creg,3,MSG_TEXT,NONCRIT,"--- UKF43 INHIBITED EPN %s ",epn);
                sleep (.2);
                }                                       /*  End If              */
        /*
         *      GET OUT AND GO TO NEXT EPN:
         */
        err = prb(net, inha[k].pcm, inha[k].dbt, inha[k].sbi, buff);
                if(err < 0)
                {
                printf("\n --UKF43 %s PRB--ERROR = %d\n", epn, err);

                unlrg( creg );
                unlrg( dbreg );
                unlrg( areg );
                exit();
                }
        sleep (1);                                      /*  Take a rest                 */
        }                                               /*  End inhibit for-loop, k++   */
    }                                                   /*  END while <= npcm           */
/*
 *      UNLINK REGIONS AND GET OUT:
 */
quit:
        unlrg( creg );
        unlrg( dbreg );
        unlrg( areg );
        if( anydone )
                {
                sleep ( 5 );

        err = d3spawn("SUBMIT D3SOURCE:[261,10]RESET.COM",1,0);

                sleep ( 4 );
                printf( "UKF43 - ALARM INHIBIT COMPLETED \n" );
                }
        else
                printf( "UKF43 - ALARM INHIBIT WARNING - ALL PCM NODES OFFLINE \n" );
        return(1);

}                                                       /*  End main  */

/************************
 *                      *
 *    E N D   M A I N   *
 *                      *
 ************************/

/*
 *      W R T _ C O N S O L E (   )
 *
 *      Draw the General Function 43 - Display
 */
int wrt_console( con )
int con;
{
    int    kbfd;                        /*  Key board fd        */
    int    kboutfd;                     /*  Keyboard output fd  */
    int    err;                         /*  Error code          */
    static char dlname[] = "DLISTxx";

        dlname[5 + itob ( &dlname[5], con ,10 ) ] = 0;
        if((dlrd = lnkrg(dlname)) < 0)
            {
            printf("UKF43 - DLIST LINK FAILURE %d\n",dlrd);
            unlrg(dlrd);
            exit();
            }
        if((err = ginit(&gcb, con)) < 0 )
```

```
        {
        printf("UKF43 - CANT ACCESS GRINNELL, ERR %d\n",err );
        unlrg(dlrd);
        exit();
        }
    dlist.act = DMUSER + UDF43;
    putrg(dlrd, &dlist.act, OFFSET(dlist,act), sizeof(dlist.act) );
/*
 *      C L E A R  the  O C M :
 */
    gsetup(&gcb,BLACK);
    erase(&gcb);
    gflush(&gcb);

    /*  O C M    G R A P H I C S :  */

    gsetup(&gcb,RED);           /*      S C R E E N   F O R M A T    */
    cursor(&gcb,10,425);        /*  .________________________________. */
    horiz( &gcb,440,3);         /*  |   General Function  43         | */
    cursor(&gcb,447,425);       /*  |--------------------------------| */
    vert(  &gcb,-165,3);        /*  |   Inhibiting Alarms.           | */
    cursor(&gcb,449,260);       /*  |                                | */
    horiz( &gcb,-439,3);        /*  |   EPN's printing at typer.     | */
    cursor(&gcb,10,260);        /*  |                                | */
    vert(  &gcb,165,3);         /*  |   Clearing CDCM pages.         | */
    cursor(&gcb,10,390);        /*  |                                | */
    horiz( &gcb,440,3);         /*  |   Wait ( 1 minute ),           | */
                                /*  |                                | */
                                /*  |   Then run GF 44 on DCM0.      | */
                                /*  |________________________________| */

    gsetup(&gcb, WHITE );
    dublchar(&gcb);
    cursor(&gcb,HCHAR*3,VCHAR*40);
    g_puts( &gcb, "GENERAL FUNCTION 43" );
    gflush(&gcb);
    gsetup( &gcb, YELLOW );
    widechar(&gcb);
    cursor(&gcb,HCHAR*3,VCHAR*36);
    g_puts( &gcb, "INHIBITING ALARMS FOR 242A SHUTDOWN" );
    gflush(&gcb);
    cursor(&gcb,HCHAR*3,VCHAR*34);
    g_puts( &gcb, "EPN'S WILL BE PRINTED AT TYPER." );
    gflush(&gcb);
    cursor(&gcb,HCHAR*3,VCHAR*32);
    g_puts( &gcb, "CLEARING CDCM ALARM SUMMARY PAGES." );
    gflush(&gcb);
    gsetup( &gcb, RED );
    cursor(&gcb,HCHAR*3,VCHAR*30);
    g_puts( &gcb, "WAIT UNTIL DONE ( 1 MINUTE )," );
    gflush(&gcb);
    gsetup( &gcb, GREEN );
    cursor(&gcb,HCHAR*3,VCHAR*28);
    g_puts( &gcb, "THEN RUN GENL FCTN 44 AT ONE DCM0 CONSOLE." );
    gflush(&gcb);
    gsetup(&gcb, WHITE );
    cursor(&gcb,HCHAR*72,VCHAR*25);
    g_puts( &gcb, "MCT" );
    gflush(&gcb);
/*
 *      RELEASE THE CONSOLE AND DLIST REGION:
 */
    dlist.ukf = -1;
    dlist.act = 0;
    putrg(dlrd,&dlist.ukf,OFFSET(dlist,ukf),sizeof(dlist.ukf));
    putrg(dlrd,&dlist.act,OFFSET(dlist,act),sizeof(dlist.act));
    unlrg( dlrd );
    return(1);
}                                   /*  End wrt_console() */
```

UKF44

```
/*              U K F 4 4 . C                                   VERSION 2
 *
 *
 *
 *  General Function 44 - General Function 44 is to be used only on
 *                        DCM0 following GF 43 (alarm inhibiting).  GF 44
 *                        clears the alarm summary pages on DCM0.  The
 *                        alarm summary pages on the CDCM are cleared by
 *                        GF43, which spawns the mcr command file to reset
 *                        the summary pages.
 *
 *                        After GF 44 is run on DCM0, GF 14 is then run on
 *                        the CDCM to regenerate alarms.
 *
 *
 *      Written by : M. C. Teats
 *      Date       : 03-08-93
 *      Facility   : 242-A Evaporator Bldg. */
/*  Modified by: John Hayden T.I.          6.3-2 Upgrade    */
/*  Facility   : 242-A Evaporator Bldg.                     */
/*  Machine    : VAX 4104-A                                 */
/*  Compiler   : DEC VAX C V3.3                    JH01     */
/*  C .cmd use : SET DEF D3SOURCE::[261,10]        JH01     */
/*  C .cmd use : BLD UKF14 UKF14.C -L -MOV         JH01     */
/*  Op. System : VAX/VMS 5.5-2AH          JH01     */
/*  D/3 Version: 6.3-2   BASELINE                  JH01     */
/*  #INCLUDES CHANGED                              JH01     */
/*  PARAMATERS TO FUNCTIONS CHANGED                JH01     */
/*  PRAGAMA FOR DLIST DECLARATIONS, INITIALIZE     JH01     */
/*  SLEEP NOW SECONDS INSTEAD OF TICKS             JH01     */
/*  %P NOT %S, PUTFMT-> PRINTF,%I->%D              JH01     */
/*      Run on Unit: DCM0 only
 *
 *      Build command file : @[261,10]UKF44
 *
 *
 *  MODIFICATION LOG:
 *
 *      08-MAR-93       Define program for D/3 version 3.3              M. Teats
 *
 *      05-JAN-95       Rebaselined software for upgrade of             J. Berger
 *                      D/3 software to version 6.3.
 *
 *  END OF MOD LOG
 */

#include <emcstd.h>
#include <d3sys.h>
#include <unitdb.h>
#include <dbatypes.h>
#include <grinnell.h>
#include <dbaai.h>
#include <dbadin.h>
#include <digdef.h>
#include <conkey.h>
#include <conf.h>
#include <err.h>
#include <keyequ.h>
#include <func.h>
#include <almmsg.h>

#define UDF44           44
#define DONE            1
#pragma nostandard
PUBLIC    int     dlrd    =0;             /* DLIST descriptor                 */
PUBLIC    struct dlist    dlist   =0;     /* DLIST structure                  */
PUBLIC    struct gcb      gcb     =0;     /* Struct for Grinnel Control Block */
PUBLIC    short  con = 0;
#pragma standard
```

```
/*************************
 *                       *
 *  M A I N  ( UKF44 )   *
 *                       *
 *************************/

main( )
{
    int  err,                           /*  Error code                          */

        con = gtnum();
        if( con < 0 || con > 3 )
                {
                con = 0;
                printf("\n-- UKF44 - ALARM SUMMARY PAGE CLEARING ( Invoked - Interactive Mode )\n" );
                }
        else                            /*  Else - Invoked by Gen Func 44 at console    */
                {
                printf("\n-- UKF44 - ALARM SUMMARY PAGE CLEARING Invoked from Console %d \n", con );
                wrt_console( con );
                }                       /*  End else from CONSOLE                       */
/*
 *      SPAWN RESET.CMD TO CLEAR ALARM SUMMARY PAGES ON THIS DCM
 */


/*
                        sprintf(buffer, "@d3com:pcmlod %s all", node_name);
                        exit_code = d3spawn(buffer, 1, 0);
                        if (exit_code == ER_GEN)
                            printf("Error spawning PCMLOD\n");
                        sleep(3);
                        printf("%-80s", " ");

*/

        err = d3spawn("SUBMIT D3SOURCE:[261,10]RESET.COM",1,0);

        if(err == ER_GEN)
                printf("error spawning\n");

}

/*************************
 *                       *
 *   E N D   M A I N     *
 *                       *
 *************************/

/*
 *      W R T _ C O N S O L E (   )
 *
 *      Draw the General Function 44 - Display
 */
int wrt_console( con )
int con;
{
    int    kbfd;                        /*  Key board fd        */
    int    kboutfd;                     /*  Keyboard output fd  */
    int    err;                         /*  Error code          */
    static char dlname[] = "DLISTxx";

        dlname[5 + itob( &dlname[5], con ,10)] = 0;
        if((dlrd = lnkrg(dlname)) < 0)
            {
            printf("UKF44 - DLIST LINK FAILURE %d\n",dlrd);
            unlrg(dlrd);
            exit();
            }
        if((err = ginit(&gcb, con)) < 0 )
```

```
        {
        printf("UKF44 - CANT ACCESS GRINNELL, ERR %d\n",err );
        unlrg(dlrd);
        exit();
        }
    dlist.act = DMUSER + UDF44;
    putrg(dlrd, &dlist.act, OFFSET(dlist,act), sizeof(dlist.act) );
/*
 *      C L E A R  the  O C M :
 */
    gsetup(&gcb,BLACK);
    erase(&gcb);
    gflush(&gcb);

    /*  O C M    G R A P H I C S :  */

    gsetup(&gcb,RED);          /*      S C R E E N   F O R M A T    */
    cursor(&gcb,10,425);       /*  .-------------------------------. */
    horiz( &gcb,440,3);        /*  |  General Function  44         | */
    cursor(&gcb,447,425);      /*  |-------------------------------| */
    vert(  &gcb,-125,3);       /*  |  Clearing Alarm Pages.        | */
    cursor(&gcb,449,300);      /*  |                               | */
    horiz( &gcb,-439,3);       /*  |  Wait ( 1 minute ),           | */
    cursor(&gcb,10,300);       /*  |                               | */
    vert(  &gcb,125,3);        /*  |  Then run GF 14 on CDCM.      | */
    cursor(&gcb,10,390);       /*  |_______________________________| */
    horiz( &gcb,440,3);

    gsetup(&gcb, WHITE );
    dublchar(&gcb);
    cursor(&gcb,HCHAR*3,VCHAR*40);
    g_puts( &gcb, "GENERAL FUNCTION 44" );
    gflush(&gcb);
    gsetup( &gcb, YELLOW );
    widechar(&gcb);
    cursor(&gcb,HCHAR*3,VCHAR*36);
    g_puts( &gcb, "CLEARING ALARM PAGES FOR THIS UNIT." );
    gflush(&gcb);
    g_puts( &gcb, " " );
    gflush(&gcb);
    gsetup( &gcb, RED );
    cursor(&gcb,HCHAR*3,VCHAR*34);
    g_puts( &gcb, "WAIT UNTIL DONE ( 30 SECONDS )," );
    gflush(&gcb);
    gsetup( &gcb, GREEN );
    cursor(&gcb,HCHAR*3,VCHAR*32);
    g_puts( &gcb, "THEN RUN GENL FCTN 14 AT CDCM CONSOLE." );
    gflush(&gcb);
    gsetup(&gcb, WHITE );
    cursor(&gcb,HCHAR*72,VCHAR*29);.
    g_puts( &gcb, "JH" );
    gflush(&gcb);
/*
 *      RELEASE THE CONSOLE AND DLIST REGION:
 */
    dlist.ukf = -1;
    dlist.act = 0;
    putrg(dlrd,&dlist.ukf,OFFSET(dlist,ukf),sizeof(dlist.ukf));
    putrg(dlrd,&dlist.act,OFFSET(dlist,act),sizeof(dlist.act));
    unlrg( dlrd );
    return(1);
    exit();
}                                   /*  End wrt_console() */
```

APPENDIX G - STRING PACKET LISTING

Listing of string.out Created using SED from CDCM D3DATA:[100010]STRING.BIN

247

[0]

<0>
<1> ¦ALARM (UP)
<2> ALARM
<3> ¦ALARM (UP)
<4> ALARM
<5> ¦ALARM (UP)
<6> ALARM

[1]

<0>
<1> ¦LOW FLOW
<2> LOW FLOW
<3> LOW FLOW
<4> LOW FLOW
<5> LOW FLOW
<6> LOW FLOW

[2]

<0>
<1> ¦50 LB INSTRUMENT AIR LOW
<2> 50 LB INSTRUMENT AIR LOW
<3> 50 LB INSTRUMENT AIR LOW
<4> 50 LB INSTRUMENT AIR LOW
<5> 50 LB INSTRUMENT AIR LOW
<6> ALARM

[3]

<0>
<1> ¦ERROR
<2> ERROR
<3> ERROR
<4> ERROR
<5> ERROR
<6> ALARM

[4]

<0>
<1> ¦LOW
<2> LOW
<3> LOW
<4> LOW
<5> LOW
<6> LOW

[5]

<0>
<1> ¦ANNULUS EXH RAD HI/FAIL
<2> ANNULUS EXH RAD HI/FAIL
<3> ANN EXH RAD HI/FAIL INHA
<4> ¦ANN EXH RAD HI/FAIL OSCN
<5> ANN EXH RAD HI/FAIL IBAD

[6]

<0>
<1> ¦LEAK PIT RAD HI/FAIL
<2> LEAK PIT RAD HI/FAIL
<3> L-PIT RAD HI/FL INHA
<4> ¦L-PIT RAD HI/FL OSCN
<5> L-PIT RAD HI/FL IBAD

[7]

<0>
<1> ¦ALARM
<2> ALARM
<3> ¦ALARM
<4> ALARM
<5> ¦ALARM
<6> ALARM

[8]

<0>
<1> ¦SERVICE PIT RAD HI/FAIL
<2> SERVICE PIT RAD HI/FAIL
<3> SERPIT RAD DET INHA
<4> ¦SER PIT RAD DET OSCN
<5> SER PIT RAD DET IBAD

[9]

<0>
<1> ¦SERVICE PIT LEAK
<2> SERVICE PIT LEAK
<3> SERPIT LEAK INHA
<4> ¦SERPIT LEAK OSCN
<5> SERPIT LEAK IBAD

[10]

<0>
<1> ¦PROCESS PITS LEAK
<2> PROCESS PITS LEAK
<3> PRO PIT LEAK INHA
<4> ¦PRO PIT LEAK OSCN
<5> PRO PIT LEAK IBAD

[11]

<0>
<1> ¦SL-160 SL-168 LEAK
<2> SL-160 SL-168 LEAK
<3> SL-160 SL-168 INHA
<4> ¦SL160/168 OFF SCAN
<5> SL160/168 BAD INPUT

[12]

<0>
<1> ¦SN-260 SN-268 LEAK
<2> SN-260 SN-268 LEAK
<3> SN260/268 INHA
<4> ¦SN-260/268 OSCN
<5> SN260/268 BAD INPUT

[13]

<0>
<1> ¦CLEANOUT BOX LEAK
<2> CLEANOUT BOX LEAK
<3> COB LEAK INHA
<4> ¦COM LEAK DIS
<5> COB LEAK IBAD

[14]
<0>
<1> ¦SN FLUSH LINE PRESSURE HI
<2> SN FLUSH LINE PRESSURE HI
<3> SN FLUSH LINE PRES INHA
<4> ¦SN FLUSH LINE PRESS DIS
<5> SN FLUSH LINE PRESSURE IBAD

[15]
<0>
<1> ¦AN DR-368 ENCASE LEAK
<2> AN DR-368 ENCASE LEAK
<3> AN368 ENCASE INHA
<4> ¦AN368 ENCASE DIS
<5> AN368 ENCASE BAD INPUT

[16]
<0> 271 BUILDING POWER NORM
<1> ¦271 BUILDING POWER LOSS
<2> 271 BUILDING POWER LOSS
<3> BLD POWER LOSS ALM INHA
<4> ¦BLD POWER LOSS DIS
<5> BLD POW LOSS BAD INPUT

[17]
<0> 271 BUILDING RADIATION NORMAL
<1> ¦271 BUILDING RADIATION HI/FAIL
<2> 271 BUILDING RADIATION HI/FAIL
<3> 271 BUILDING RADIATION INHA
<4> ¦271 BUILDING RADIATION DIS
<5> BLD RAD ALARM BAD INPUT

[18]
<0> 241 AREA RADIATION NORMAL
<1> ¦241 AREA RADIATION HI/FAIL
<2> 241 RADIATION HI/FAIL
<3> 241 RAD HI/FAIL INHA
<4> ¦241 RAD HI/FAIL DIS
<5> 241 RAD HI/FAIL BAD INPUT

[19]
<0> GENERAL ALARM OFF
<1> ¦GENERAL ALARM ON
<2> GENERAL ALARM ON
<3> GENERAL ALM INHA
<4> ¦GENERAL ALM DIS
<5> GENERAL ALM IBAD

[20]
<0> RADIATION BYPASS OFF
<1> ¦AREA RADIATION BYPASS
<2> AREA RADIATION BYPASS
<3> AREA RAD BYPASS INHA
<4> ¦AREA RAD BYPASS DIS
<5> AREA RAD BYPASS IBAD

[21]
<0> MASTER SHUTDOWN OFF
<1> ¦FARM MASTER SHUTDOWN
<2> FARM MASTER SHUTDOWN
<3> MSTR SHUTDOWN INHA
<4> ¦MSTR SHUTDOWN DIS
<5> MSTR SHUTDN BAD INPUT

[22]
<0>
<1> ¦TANK EXH RAD HI
<2> TANK EXH RAD HI
<3> TANK EXHRD INHA
<4> ¦TANK EXHRD DIS
<5> TANK EXHRD IBAD

[23]
<0> STACK MONITOR SAMPLE PUMP ON
<1> ¦STACK MONITOR SAMPLE PUMP OFF
<2> STACK MONITOR SAMPLE PUMP OFF
<3> STACK MONITOR SAMPLE PMP INHA
<4> ¦STACK MONITOR SAMPLE PMP DIS
<5> STACK MONITOR SAMPLE PMP IBAD

[24]
<0> RECORD SAMPLE PUMP ON
<1> ¦RECORD SAMPLE PUMP OFF
<2> RECORD SAMPLE PUMP OFF
<3> RECORD SAMPLE PMP INHA
<4> ¦RECORD SAMPLE PMP DIS
<5> RECORD SAMPLE PMP IBAD

[25]
<0>
<1> ¦TANK EXH RAD MON FAIL
<2> TANK EXH RAD MON FAIL
<3> TANK EXH RAD MON INHA
<4> ¦TANK EXH RAD MON DIS
<5> TANK EXH RAD MON IBAD

[26]
<0> TANK EXH FANS ON
<1> ¦TANK EXH FANS OFF
<2> TANK EXH FANS OFF
<3> TANK EXH FAN INHA
<4> ¦TANK EXH FAN DIS
<5> TANK EXH FAN IBAD

[27]
<0> ANN EXH RAD HI/FAIL FAN NORM
<1> ¦ANN EXH RAD HI/FAIL FAN FAIL
<2> ANN EXH RAD HI/FAIL FAN FAIL
<3> ANN EXH RAD/FAN HI/FAIL INHA
<4> ¦ANN EXH RAD/FAN HI/FAIL DIS
<5> ANN EXH RAD/FAN HI/FAIL IBAD

[28]
<0> ANNULUS EXHAUST RADIATION MONITOR NORMAL
<1> ¦ANNULUS EXHAUST RADIATION MONITOR FAILURE
<2> ANNULUS EXHAUST RADIATION MONITOR FAILURE
<3> ANN EXHAUST RAD MONITOR FAILURE INHA
<4> ¦ANN EXHAUST RAD MONITOR FAILURE DIS
<5> ANN EXHAUST RAD MONITOR FAILURE BAD INPUT

[29]
<0> ANNULUS EXH RADIATION OK
<1> ¦ANNULUS EXH RADIATION HI
<2> ANNULUS EXH RADIATION HI
<3> ¦ANN EXH RAD MON FAILURE
<4> ANN EXH RAD MON FAILURE
<5> ANN EXH RAD MON INHIBIT
<6> ANN EXH RAD MON OFF SCAN
<7> ANN EXH RAD MON BAD INPT

[30]
<0>
<1> ¦LEAK PIT RADIATION HI
<2> LEAK PIT RADIATION HI
<3> LEAK PIT RAD MON INHA
<4> LEAK PIT RAD OFF SCAN
<5> LEAK PIT RAD BAD INPT
<6>
<7>

[31]
<0> DRAIN PIT LEAK
<1> ¦DRAIN PIT LEAK
<2> DRAIN PIT LEAK
<3> DRN PIT LK INHA
<4> ¦DRN PIT LK DIS
<5> D PIT LK BD IPT

[32]
<0> SERVICE PIT MONITOR NORMAL
<1> ¦SERVICE PIT MONITOR FAILURE
<2> SERVICE PIT MONITOR FAILURE
<3> SERVICE PIT MONITOR INHA
<4> ¦SERPIT MON FAILURE DIS
<5> SER PIT MON FAIL BAD INPUT

[33]
<0> RADIATION MONITOR NORMAL
<1> ¦RADIATION MONITOR FAILURE
<2> RADIATION MONITOR FAILURE
<3> RAD MON FAIL ALM INHIBIT
<4> RAD MON FAIL OFF SCAN
<5> RAD MON FAIL BAD INPUT

[34]
<0>
<1> ¦RADIATION HIGH
<2> RADIATION HIGH
<3> RADIATION HIGH
<4> RADIATION HIGH
<5> RADIATION HIGH

[35]
<0> SL-509/SL-510

[36]
<0>
<1> ¦SL-509/SL-510 LEAK DETECTED
<2> SL-509/SL-510 LEAK DETECTED
<3> SL-509/SL-510 LEAK DET INHA
<4> SL-509/SL-510 LEAK DET OSCN
<5> SL-509/SL-510 LEAK DET IBAD

[37]
<0>
<1> ¦SN-609/SN-610 LEAK DETECTED
<2> SN-609/SN-610 LEAK DETECTED
<3> SN-609/SN-610 LEAK DET INHA
<4> SN-609/SN-610 LEAK DET OSCN
<5> SN-609/SN-610 LEAK DET IBAD

[38]
<0>
<1> ¦SN-621 LEAK DETECTED
<2> SN-621 LEAK DETECTED
<3> SN-612 LEAK DET INHA
<4> SN-621 LEAK DET OSCN
<5> SN-621 LEAK DET IBAD

[39]
<0>
<1> ¦SN-650 ENCASEMENT LEAK DETECTED
<2> SN-650 ENCASEMENT LEAK DETECTED
<3> SN-650 ENCASEMENT LEAK DET INHA
<4> SN-650 ENCASEMENT LEAK DET OSCN
<5> SN-650 ENCASEMENT LEAK DET IBAD

[40]
<0> NORMAL
<1> ¦FLOW
<2> FLOW
<3> INHA
<4> OSCN
<5> IBAD

[41]
<0>
<1> ¦FLOW DETECTED
<2> FLOW DETECTED
<3> FLOW DET INHA
<4> FLOW DET OSCN
<5> FLOW DET IBAD

[42]
<0> CLOSED
<1> ¦OPEN
<2> OPEN
<3> INHA
<4> OSCN
<5> IBAD

[43]
<0>
<1> ¦HIGH PRESSURE
<2> HIGH PRESSURE
<3> PSH-APFL1 INHA
<4> PSH-APFL1 OSCN
<5> PSH-APFL1 IBAD

[44]
<0>
<1> ¦SHUTDOWN
<2> SHUTDOWN
<3>
<4>

[45]
<0>
<1> ¦LEAK DETECTED
<2> LEAK DETECTED
<3> LEAK DET INHA
<4> ¦LEAK DET DIS
<5> LEAK DET IBAD

[46]
<0> EXHAUST FANS ON
<1> ¦EXHAUST FANS OFF
<2> EXHAUST FANS OFF
<3> EXH FANS INHIBIT
<4> EXH FANS OFF SCAN
<5> EXH FANS BAD INPUT

[47]
<0> MONITOR SAMPLE FLO NORM
<1> ¦MONITOR SAMPLE LOW FLOW
<2> MONITOR SAMPLE LOW FLOW
<3> MON SMPL LOW FLOW INHA
<4> MON SMPL LOW FLOW OSCN
<5> MON SMPL LOW FLOW IBAD

[48]
<0> EXHAUST RADIATION NORM
<1> ¦EXHAUST RADIATION HIGH
<2> EXHAUST RADIATION HIGH
<3> EXH RAD ALARM INHIBITED
<4> EXH RAD ALARM OFF SCAN
<5> EXH RAD ALARM BAD INPUT

[49]
<0> RADIATION MONITOR NORMAL
<1> ¦RADIATION MONITOR FAILED
<2> RADIATION MONITOR FAILED
<3> RAD MON FAIL ALM-INHIBIT
<4> RAD MON FAIL ALARM OSCN
<5> RAD MON FAIL ALARM IBAD

[50]
<0> RECORD SAMPLE FLO NORM
<1> !RECORD SAMPLE LOW NORM
<2> RECORD SAMPLE LOW FLOW
<3> REC SAMPLE ALM INHIBIT
<4> REC SAMPLE ALARM OSCN
<5> REC SAMPLE ALARM IBAD

[51]
<0> STACK FLO NORM
<1> ¦STACK FLOW LOW
<2> STACK FLOW LOW
<3> STACK FLOW LOW
<4> STACK FLOW LOW
<5> STACK FLOW LOW

[52]
<0> SEAL-POT PIT NORMAL
<1> ¦SEAL POT PIT LEAK
<2> SEAL POT PIT LEAK
<3> SEAL POT PIT LEAK ALARM INHA
<4> SEAL POT PIT LEAK ALARM OSCN
<5> SEAL POT PIT LEAK ALARM IBAD

[53]
<0>
<1> ¦FLOW DETECT FAILED
<2> FLOW DETECT FAILED
<3> FLOW DET FAIL INHA
<4> FLOW DET FAIL OSCN
<5> FLOW DET FAIL IBAD
<6>

[54]
<0>
<1> ¦FAILURE
<2> FAILURE
<3> INHIBIT
<4> OFF SCN
<5> BAD INP

[55]
<0> ENCASEMENT
<1> ¦ENCASEMENT
<2> ENCASEMENT
<3> ENCASEMENT
<4> ENCASEMENT
<5> ENCASEMENT

[56]
<0>
<1> ¦LEAK
<2> LEAK
<3> INHA
<4> OSCN
<5> IBAD

[57]
<0>
<1> ¦VALVE PIT LEAK
<2> VALVE PIT LEAK
<3> VALVE PIT INHA
<4> VALVE PIT OSCN
<5> VALVE PIT IBAD

[58]
<0> OFF>
<1> ON > ¦HI
<2> ¦LO
<3> ¦HIHI
<4> ¦LOLO
<5> ¦ROC
<6> ¦LOOP
<7> ¦DEV
<8> ¦OBAD
<9> ¦RANGE
<10> ¦OVER
<11> ¦OC
<12> ¦IBAD
<13> ¦UNDR
<14> ¦TMR
<15> ¦CNTR
<16> HI
<17> LO
<18> HIHI
<19> LOLO
<20> ROC
<21> LOOP
<22> DEV
<23> OBAD
<24> RANG
<25> OVER

[59]
<0> HIGH TEMPERATURE
<1> ¦HIGH TEMPERATURE
<2> HIGH TEMPERATURE
<3> HI TEMP DET INHA
<4> HI TEMP DET OSCN
<5> HI TEMP DET IBAD

[60]
<0>
<1> ¦MOTOR HOUSING LEAK
<2> MOTOR HOUSING LEAK
<3> MOTOR LEAK INHA
<4> MOTOR LEAK OSCN
<5> MOTOR LEAK IBAD

[61]
<0>
<1> ¦HIGH VIBRATION
<2> HIGH VIBRATION
<3> HI VIB DET INHA
<4> HI VIB DET OSCN
<5> HI VIB DET IBAD

[62]
<0>
<1> ¦EXCESSIVE SHAFT STRAIN
<2> EXCESSIVE SHAFT STRAIN
<3> EXCESS SHAFT STRN INHA
<4> EXCESS SHAFT STRN OSCN
<5> EXCESS SHAFT STRN IBAD

[63]
<0>
<1> ¦DISCHARGE PRESS LOW
<2> DISCHARGE PRESS LOW
<3> DIS PRES LOW INHA
<4> DIS PRES LOW OSCN
<5> DIS PRES LOW IBAD

[64]
<0>
<1> ¦DISCHARGE PRESS HIGH
<2> DISCHARGE PRESS HIGH
<3> DIS PRESS HIGH INHA
<4> DIS PRESS HIGH OSCN
<5> DIS PRESS HIGH IBAD

[65]
<0>
<1> ¦HIGH TEMPERATURE
<2> HIGH TEMPERATURE
<3> HI TEMP ALM INHA
<4> HI TEMP ALM OSCN
<5> HI TEMP ALM IBAD

[66]
<0>
<1> ¦RADN HIGH/FAIL
<2> RADN HIGH/FAIL
<3> RAD HI/FA INHA
<4> RAD HI/FA OSCN
<5> RAD HI/FA IBAD

[67]
<0>
<1> ¦ANNULUS EXHAUST
<2> ANNULUS EXHAUST
<3> ANNULUS EXHAUST
<4> ANNULUS EXHAUST
<5> ANNULUS EXHAUST

[68]
<0>
<1> ¦LEAK PIT
<2> LEAK PIT
<3> LEAK PIT
<4> LEAK PIT
<5> LEAK PIT
<6>

[69]
<0>
<1> ¦ALARM (UP)
<2> ALARM
<3> ¦ALARM (UP)
<4> ALARM
<5> ¦ALARM (UP)
<6> ALARM
<7>
<8>
<9>
<10>
<11>

[70]
<0>
<1> ¦ALARM (DN)
<2> ALARM
<3> ¦ALARM (DN)
<4> ALARM
<5> ¦ALARM (DN)
<6> ALARM
<7>
<8>
<9>
<10>
<11>

[71]
<0>
<1> ¦INTERLOCK
<2> INTERLOCK
<3> INTERLOCK
<4> INTERLOCK
<5>
<6>
<7>

[72]
<0> LEAK DETECTION ALARM
<1> ¦LEAK DETECTION ALARM
<2> LEAK DETECTION ALARM
<3> LEAK DETECT INHIBITED
<4> LEAK DETECT OFF SCAN
<5> LEAK DETECT BAD INPUT
<6>
<7>

[73]
<0> SEAL POT HI ALM NORM
<1> ¦SEAL POT HI ALM FAIL
<2> SEAL POT HI ALM FAIL
<3> SEAL POT HI ALM INHA
<4> SEAL POT HI ALM OSCN
<5> SEAL POT HI ALM IBAD
<6>
<7>

[74]
<0> SEAL POT ALARM NORM
<1> ¦SEAL POT HIGH ALARM
<2> SEAL POT HIGH ALARM
<3> SEAL POT HI ALM INHA
<4> SEAL POT HI ALM OSCN
<5> SEAL POT HI ALM IBAD
<6>
<7>

[75]
<0> SEAL POT LOW ALM NORM
<1> ¦SEAL POT LOW LEVEL ALM
<2> SEAL POT LOW LEVEL ALM
<3> SEAL POT LOW ALM INHA
<4> SEAL POT LOW ALM OSCN
<5> SEAL POT LOW ALM IBAD
<6>
<7>

[76]
<0> NORMAL
<1> ¦GENERAL ALARM
<2> GENERAL ALARM
<3> GEN ALARM INHA
<4> GEN ALARM OSCN
<5> GEN ALARM IBAD
<6>
<7>

[77]
<0> RADIATION NORM
<1> ¦RADIATION HIGH
<2> RADIATION HIGH
<3> RAD DETECT INHA
<4> RAD DETECT OSCN
<5> RAD DETECT IBAD
<6>
<7>

[78]
<0>
<1> ¦LOLO
<2> LOLO
<3> ¦LO
<4> LO
<5> ¦HI
<6> HI
<7> ¦HIHI
<8> HIHI
<9> INHA
<10> OVER
<11> OC
<12> IBAD
<13> UNDR
<14> TMR
<15> CNTR
<16> ¦HI
<17> ¦LO
<18> ¦HIHI
<19> ¦LOLO
<20> ¦ROC
<21> ¦LOOP
<22> ¦DEV
<23> ¦OBAD
<24> ¦RANG
<25> ¦OVER
<26> ¦OC
<27> ¦IBAD
<28> ¦UNDR
<29> ¦TMR
<30> ¦CNTR
<31>
<32>

[79]
<0> LIFT STATION LEAK
<1> ¦LIFT STATION LEAK
<2> LIFT STATION LEAK
<3> LEAK DET INHIBIT
<4> LEAK DET OFFSCAN
<5> LEAK DET BADINPUT
<6>
<7>

[80]
<0> DIVERSION BOX LEAK
<1> ¦DIVERSION BOX LEAK
<2> DIVERSION BOX LEAK
<3> DIV BOX LEAK INHA
<4> DIV BOX LEAK OSCN
<5> DIV BOX LEAK IBAD
<6>
<7>

[81]
<0> VALVE LEAK
<1> ¦VALVE LEAK
<2> VALVE LEAK
<3> VLVLK INHA
<4> VLVLK OSCN
<5> VLVLK IBAD
<6>
<7>

[82]
<0>
<1> ¦HI PRESSURE
<2> HI PRESSURE
<3> HI PRES INHA
<4> HI PRES OSCN
<5> HI PRES IBAD
<6>
<7>

[83]
<0> HEAT TRACE TEMPERATURE NORM
<1> ¦HEAT TRACE OVERTEMPERATURE
<2> HEAT TRACE OVERTEMPERATURE
<3> HEAT TRACE TEMP MON INHIBIT
<4> HEAT TRACE TEMP MON OFFSCAN
<5> HEAT TRACE TEMP MON IBAD
<6>
<7>

[84]
<0> 241 AREA RADIATION NORM
<1> ¦241 AREA RADIATION HIGH
<2> 241 AREA RADIATION HIGH
<3> 241 AREA RAD MONIT INHA
<4> 241 AREA RAD MONIT OSCN
<5> 241 AREA RAD MONIT IBAD
<6>
<7>

[85]
<0> 271 BUILDING RADIATION NORM
<1> ¦271 BUILDING RADIATION HIGH
<2> 271 BUILDING RADIATION HIGH
<3> 271 BLDG RADIATION INHIBIT
<4> 271 BLDG RADIATION OFF SCAN
<5> 271 BLDG RADIATION BAD INPT
<6>
<7>

[86]
<0> CENTRAL PUMP PIT NORM
<1> ¦CENTRAL PUMP PIT LEAK
<2> CENTRAL PUMP PIT LEAK
<3> CENTRAL PUMP PIT INHA
<4> CENTRAL PUMP PIT OSCN
<5> CENTRAL PUMP PIT IBAD
<6>
<7>

[87]
<0>
<1> ¦PUMP PIT LEAK
<2> PUMP PIT LEAK
<3> LEAK DET INHA
<4> LEAK DET OSCN
<5> LEAK DET IBAD
<6>
<7>

[88]
<0> 271 CAM FLOW NORM
<1> ¦271 CAM FLOW LOW
<2> 271 CAM FLOW LOW
<3> 271 CAM FLOW INHA
<4> 271 CAM FLOW OSCN
<5> 271 CAM FLOW IBAD
<6>
<7>

[89]
<0> ANNULUS PUMP PIT NORM
<1> ¦ANNULUS PUMP PIT LEAK
<2> ANNULUS PUMP PIT LEAK
<3> ANNULUS PUMP PIT INHA
<4> ANNULUS PUMP PIT OSCN
<5> ANNULUS PUMP PIT IBAD
<6>
<7>

[90]
<0> ANNUNCIATOR POWER NORM
<1> ¦ANNUNCIATOR POWER LOSS
<2> ANNUNCIATOR POWER LOSS
<3> ANNUNCIATOR POWER INHA
<4> ANNUNCIATOR POWER OSCN
<5> ANNUNCIATOR POWER IBAD
<6>
<7>

[91]
<0> BUILDING RADIATION MONITOR NORM
<1> ¦BUILDING RADIATION MONITOR FAIL
<2> BUILDING RADIATION MONITOR FAIL
<3> BUILDING RADIATION MONITOR INHA
<4> BUILDING RADIATION MONITOR OSCN
<5> BUILDING RADIATION MONITOR IBAD
<6>
<7>

[92]
<0> 241 AREA RADIATION MONITOR NORM
<1> ¦241 AREA RADIATION MONITOR FAIL
<2> 241 AREA RADIATION MONITOR FAIL
<3> 241 AREA RADIATION MONITOR INHA
<4> 241 AREA RADIATION MONITOR OSCN
<5> 241 AREA RADIATION MONITOR IBAD
<6>
<7>

[93]
<0>
<1> ¦WF HIGH
<2> WF HIGH
<3> WF HIGH
<4> WF HIGH
<5> WF HIGH
<6>
<7>

[94]
<0>
<1> ¦DE-ENTRAINMENT
<2> DE-ENTRAINMENT
<3> ¦ LOWER PAD
<4> LOWER PAD
<5> ¦HIGH DELTA P
<6> HIGH DELTA P
<7> ¦HIGH DELTA P
<8>

[95]
<0> M
<1> A
<2>
<3>
<4>
<5>
<6>
<7>
<8>

[96]
<0>
<1> ¦REBOILER 10 LB STEAM PRESSURE LOW
<2> REBOILER 10 LB STEAM PRESSURE LOW
<3>
<4>
<5>
<6>
<7>

[97]
<0> O
<1> C
<2>
<3>
<4>
<5>
<6>
<7>

[98]
<0> DESUPERHEATER OUTLET TEMPERATURE HIGH
<1> ¦DESUPERHEATER OUTLET TEMPERATURE HIGH
<2> DESUPERHEATER OUTLET TEMPERATURE HIGH
<3>
<4>
<5>
<6>
<7>

[99]
<0> OPEN
<1> ¦CLOSED
<2> CLOSED
<3> INHA
<4> OSCN
<5> IBAD
<6>
<7>

[100]
<0>
<1> ¦SHUTDOWN
<2>
<3>
<4>
<5>
<6>
<7>

[101]
<0>
<1> ¦FAILED
<2> FAILED
<3> INHA
<4> OSCN
<5> IBAD
<6>
<7>

[102]
<0>
<1> ¦RADN HI
<2> RADN HI
<3> INHA
<4> OSCN
<5> IBAD
<6>
<7>

[103]
<0>
<1> ¦ANN EXH RAD HIGH
<2> ANN EXH RAD HIGH
<3> ANN EXH RAD INHA
<4> ANN EXH RAD OSCN
<5> ANN EXH RAD IBAD
<6>
<7>

[104]
<0>
<1> ¦ANN EXH RAD FAIL
<2> ANN EXH RAD FAIL
<3> ANN EXH RAD INHA
<4> ANN EXH RAD OSCN
<5> ANN EXH RAD IBAD
<6>
<7>

[105]
<0>
<1> ¦LEAK PIT RAD MON FAIL
<2> LEAK PIT RAD MON FAIL
<3> LEAK PIT RAD MON INHA
<4> LEAK PIT RAD MON OSCN
<5> LEAK PIT RAD MON IBAD
<6>
<7>

[106]
<0> LEAK DETECT PUMP PIT LEAK
<1> ¦LEAK DETECT PUMP PIT LEAK
<2> LEAK DETECT PUMP PIT LEAK
<3> LEAK DET PMP PIT LEAK INHA
<4> LEAK DET PMP PIT LEAK OSCN
<5> LEAK DET PMP PIT LEAK IBAD
<6>
<7>

[107]
<0> BYPASS OFF
<1> ¦BYPASS ON
<2> BYPASS ON
<3> BYPASS ON
<4> BYPASS ON
<5> BYPASS
<6>
<7>

[108]
<0>
<1> ¦ALARM
<2> ACKWD
<3>
<4>
<5>
<6>
<7>

[109]
<0>
<1> ¦ALARM (PG)
<2> ALARM
<3>
<4>
<5>
<6>
<7>

[110]
<0>
<1> ¦ALARM (LT)
<2> ALARM
<3>
<4>
<5>
<6>
<7>

[111]
<0>
<1> ¦ALARM (RT)
<2> ALARM
<3>
<4>
<5>
<6>
<7>

[112]
<0> EXCESS VACUUM ALM NORM
<1> ¦TANKS EXCESS VACUUM ALM
<2> TANKS EXCESS VACUUM ALM
<3> TANKS EXCS VAC ALM INHA
<4> TANKS EXCS VAC ALM OSCN
<5> TANKS EXCS VAC ALM IBAD
<6>
<7>

[113]
<0> VACUUM LOSS ALM NORMAL
<1> ¦TANKS VACUUM LOSS ALARM
<2> TANKS VACUUM LOSS ALARM
<3> TANKS VAC LOSS ALM INHA
<4> TANKS VACU LOSS ALM OSCN
<5> TANKS VACU LOSS ALM IBAD
<6>
<7>

[114]
<0>
<1> ¦TL-021 LINE LEAK
<2> TL-021 LINE LEAK
<3> TL-021 LEAK INHA
<4> TL-021 LEAK OSCN
<5> TL-021 LEAK IBAD
<6>
<7>

[115]
<0>
<1> ¦AW CIRCUIT TROUBLE/LEAK
<2> AW CIRCUIT TROUBLE/LEAK
<3> AW CIRCUIT TROUBL INHA
<4> AW CIRCUIT TROUBL OSCN
<5> AW CIRCUIT TROUBL IBAD
<6>
<7>

[116]
<0>
<1> ¦DRAIN ENCASEMENT LEAK
<2> DRAIN ENCASEMENT LEAK
<3> DRAIN ENCASE LK INHA
<4> DRAIN ENCASE LK OSCN
<5> DRAIN ENCASE LK IBAD
<6>
<7>

[117]
<0>
<1> ¦SLURRY ENCASEMENT LEAK
<2> SLURRY ENCASEMENT LEAK
<3> SLURRY ENCASE LK INHA
<4> SLURRY ENCASE LK OSCN
<5> SLURRY ENCASE LK IBAD
<6>
<7>

[118]
<0>
<1> ¦SUPERNATE ENCASEMENT LEAK
<2> SUPERNATE ENCASEMENT LEAK
<3> SUPERNATE ENCASE LK INHA
<4> SUPERNATE ENCASE LK OSCN
<5> SUPERNATE ENCASE LK IBAD
<6>
<7>

[119]
<0>
<1> ¦SN-269 LINE LEAK
<2> SN-269 LINE LEAK
<3> SN-269 LINE INHA
<4> SN-269 LINE OSCN
<5> SN-269 LINE IBAD
<6>
<7>

[120]
<0>
<1> ¦SN-270 LINE LEAK
<2> SN-270 LINE LEAK
<3> SN-270 LINE INHA
<4> SN-270 LINE OSCN
<5> SN-270 LINE IBAD
<6>
<7>

[121]
<0>
<1> ¦SN-FLUSH LINE PRESSURE HI
<2> SN-FLUSH LINE PRESSURE HI
<3> SN-FLUSH LINE PRESS INHA
<4> SN-FLUSH LINE PRESS OSCN
<5> SN-FLUSH LINE PRESS IBAD
<6>
<7>

[122]
<0>
<1> ¦FEEDLINE PRESSURE HI
<2> FEEDLINE PRESSURE HI
<3> FEEDLINE PRESS INHA
<4> FEEDLINE PRESS OSCN
<5> FEEDLINE PRESS IBAD
<6>
<7>

[123]
<0> CLEANOUT BOX NORM
<1> ¦CLEANOUT BOX LEAK
<2> CLEAKOUT BOX LEAK
<3> CLEANOUT BOX INHA
<4> CLEANOUT BOX OSCN
<5> CLEANOUT BOX IBAD
<6>
<7>

[124]
<0> RADIATION NORMAL
<1> ¦RADIATION HI/FAIL
<2> RADIATION HI/FAIL
<3> RAD HI/FAIL INHA
<4> RAD HI/FAIL OSCN
<5> RAD HI/FAIL IBAD
<6>
<7>

[125]
<0> CLEANOUT BOX NORM
<1> ¦CLEANOUT BOX LEAK
<2> CLEANOUT BOX LEAK
<3> CLEANOUT BOX INHA
<4> CLEANOUT BOX OSCN
<5> CLEANOUT BOX IBAD
<6>
<7>

[126]
<0>
<1> ¦MCC OFF
<2> MCC OFF
<3> MCC INHA
<4> MCC OSCN
<5> MCC IBAD
<6>
<7>

[127]
<0> ANN EXH RAD HI/FAIL NORM
<1> ¦ANN EXH RAD HI/FAIL FAIL
<2> ANN EXH RAD HI/FAIL FAIL
<3> ANN EXH RAD HI/FAIL INHA
<4> ANN EXH RAD HI/FAIL OSCN
<5> ANN EXH RAD HI/FAIL IBAD
<6>
<7>

[128]
<0>
<1> ¦TANK EXH FANS OFF
<2> TANK EXH FANS OFF
<3> TANK EXH FAN INHA
<4> TANK EXH FAN OSCN
<5> TANK EXH FAN IBAD
<6>
<7>

[129]
<0>
<1> ¦AX-B SL-100 LINE LEAK
<2> AX-B SL-100 LINE LEAK
<3> AX-B SL-100 LINE INHA
<4> AX-B SL-100 LINE OSCN
<5> AX-B SL-100 LINE IBAD
<6>
<7>

[130]
<0>
<1> ¦AX-B SN-200 SN-213 LINE LEAK
<2> AX-B SN-200 SN-213 LINE LEAK
<3> AX-B SN-200 SN-213 LEAK INHA
<4> AX-B SN-200 SN-213 LEAK OSCN
<5> AX-B SN-200 SN-213 LEAK IBAD
<6>
<7>

[131]
<0>
<1> ¦AX-A SN-201 SN-214 LINE LEAK
<2> AX-A SN-201 SN-214 LINE LEAK
<3> AX-A SN-201 SN-214 LEAK INHA
<4> AX-A SN-201 SN-214 LEAK OSCN
<5> AX-A SN-201 SN-214 LEAK IBAD
<6>
<7>

[132]
<0>
<1> ¦AX-A SL-101 LEAK
<2> AX-A SL-101 LEAK
<3> AX-A SL-101 INHA
<4> AX-A SL-101 OSCN
<5> AX-A SL-101 IBAD
<6>
<7>

[133]
<0>
<1> ¦A-B SL-113 LEAK
<2> A-B SL-113 LEAK
<3> A-B SL-113 INHA
<4> ¦A-B SL-113 OSCN
<5> A-B SL-113 IBAD
<6>
<7>

[134]
<0>
<1> ¦A-A SL-114 LEAK
<2> A-A SL-114 LEAK
<3> A-A SL-114 INHA
<4> ¦A-A SL-114 OSCN
<5> A-A SL-114 IBAD
<6>
<7>

[135]
<0>
<1> ¦HEPA DP FAN FAIL EXHAUST RADIATION
<2> HEPA DP FAN FAIL EXHAUST RADIATION
<3> HEPA DP FAN FAIL EXHAUST RAD INHA
<4> HEPA DP FAN FAIL EXHAUST RAD OSCN
<5> HEPA DP FAN FAIL EXHAUST RAD IBAD
<6>
<7>

[136]
<0>
<1> ¦HEPA DELTA PRESS HIGH
<2> HEPA DELTA PRESS HIGH
<3> HEPA DELTA PRESS INHA
<4> HEPA DELTA PRESS OSCN
<5> HEPA DELTA PRESS IBAD
<6>
<7>

[137]
<0>
<1> ¦STACK MONITOR SYSTEM FAILURE
<2> STACK MONITOR SYSTEM FAILURE
<3> STACK MONITOR SYS FAIL INHA
<4> STACK MONITOR SYS FAIL OSCN
<5> STACK MONITOR SYS FAIL IBAD
<6>
<7>

[138]
<0>
<1> ¦EXHAUST STACK RADIATION HIGH
<2> EXHAUST STACK RADIATION HIGH
<3> EXHAUST STACK RADIATION INHA
<4> EXHAUST STACK RADIATION OSCN
<5> EXHAUST STACK RADTATION IBAD
<6>
<7>

[139]
<0>
<1> ¦SERVICE PIT
<2> SERVICE PIT
<3> SERVICE PIT
<4> SERVICE PIT
<5> SERVICE PIT
<6>
<7>

[140]
<0>
<1> ¦FLUSH PIT
<2> FLUSH PIT
<3> FLUSH PIT
<4> FLUSH PIT
<5> FLUSH PIT
<6>
<7>

[141]
<0>
<1> ¦LOLO
<2> LOLO
<3> INHA
<4> OSCN
<5> IBAD
<6>
<7>

[142]
<0>
<1> ¦LO
<2> LO
<3> INHA
<4> OSCN
<5> IBAD
<6>
<7>

[143]
<0>
<1> ¦HI
<2> HI
<3> INHA
<4> OSCN
<5> IBAD
<6>
<7>

[144]
<0>
<1> ¦HIHI
<2> HIHI
<3> INHA
<4> OSCN
<5> IBAD
<6>
<7>

[145]
<0>
<1> ¦STACK FLOW HIGH
<2> STACK FLOW HIGH
<3> STACK FLOW INHA
<4> STACK FLOW OSCN
<5> STACK FLOW IBAD
<6>
<7>

[146]
<0>
<1> ¦PW-481 WASTE LINE LEAK
<2> PW-481 WASTE LINE LEAK
<3> PW-481 WASTE LINE INHA
<4> PW-481 WASTE LINE OSCN
<5> PW-481 WASTE LINE IBAD
<6>
<7>

[147]
<0>
<1> ¦LINE LEAK
<2> LINE LEAK
<3> LEAK INHA
<4> LEAK OSCN
<5> LEAK IBAD
<6>
<7>

[148]
<0>
<1> ¦BASIN LEVEL LOW
<2> BASIN LEVEL LOW
<3> BASIN LEVL INHA
<4> BASIN LEVL OSCN
<5> BASIN LEVL IBAD
<6>
<7>

[149]
<0>
<1> ¦BASIN LEVEL HIGH
<2> BASIN LEVEL HIGH
<3> BASIN LEVEL INHA
<4> BASIN LEVEL OSCN
<5> BASIN LEVEL IBAD
<6> BASIN LEVEL IBAD
<7>

[150]
<0>
<1> ¦PRESS LOW
<2> PRESS LOW
<3> PRSS INHA
<4> PRSS OSCN
<5> PRSS IBAD
<6>
<7>

[151]
<0>
<1> 24 VDC POWER SUPPLY NORM
<2> ¦24 VDC POWER SUPPLY FAIL
<3> 24 VDC POWER SUPPLY FAIL
<4> 24 VDC POWER SUPPLY INHA
<5> 24 VDC POWER SUPPLY DIS
<6> 24 VDC POWER SUPPLY IBAD
<7>
<8>

[152]
<0> DIESEL GENERATOR NORMAL
<1> ¦DIESEL GENERATOR TROUBLE
<2> DIESEL GENERATOR TROUBLE.
<3> DIESEL GENERATOR INHA
<4> DIESEL GENERATOR DIS .
<5> DIESEL GENERATOR IBAD
<6>
<7>

[153]
<0> DIESEL GENERATOR OFF
<1> ¦DIESEL GENERATOR RUNNING
<2> DIESEL GENERATOR RUNNING
<3> DIESEL GEN OFF/ON INHA
<4> DIESEL GEN OFF/ON DIS
<5> DIESEL GEN OFF/ON IBAD
<6>
<7>

[154]
<0> UPS SUMMARY NORML
<1> ¦UPS SUMMARY ALARM
<2> UPS SUMMARY ALARM
<3> UPS SUMMARY INHA
<4> UPS SUMMARY DIS
<5> UPS SUMMARY IBAD
<6> UPS SUMMARY ALARM
<7>

[155]
<0>
<1> ¦CF-ON
<2> CF-ON
<3>
<4>
<5>
<6>
<7>

[156]
<0> RAD MON NORM
<1> ¦RAD MON FAIL
<2> RAD MON FAIL
<3> RAD MON INHA
<4> RAD MON OSCN
<5> RAD MON IBAD

[157]
<0>
<1> ¦RAD HI/FAIL
<2> RAD HI/FAIL
<3> RAD INHA
<4> RAD OSCN
<5> RAD IBAD

[158]
<0>
<1> ¦ALPHA RADN HI
<2> ALPHA RADN HI
<3> ALPHA RAD INHA
<4> ALPHA RAD OSCN
<5> ALPHA RAD IBAD

[159]
<0>
<1> ¦BETA/GAMMA HI
<2> BETA/GAMMA HI
<3> BETA/GAM INHA
<4> BETA/GAM OSCN
<5> BETA/GAM IBAD
<6>

[160]
<0>
<1> ¦SYSTEM TROUBLE
<2> SYSTEM TROUBLE
<3> SYS TROUB INHA
<4> SYSTEM OSCN
<5> SYSTEM IBAD
<6>

[161]
<0> ¦RADIATION HI
<1> RADIATION HI
<2> RAD HI INHA
<3> RAD HI OSCN
<4> RAD HI IBAD

[162]
<0>
<1> ¦RAD HI/FAIL
<2> RAD HI/FAIL
<3> RAD H/F INHA
<4> RAD H/F OSCN
<5> RAD H/F IBAD

[163]
<0>
<1> ¦DP LOW
<2> DP LOW
<3>
<4>
<5>

[164]
<0>
<1> ¦DP HI
<2> DP HI
<3>
<4>
<5>

[165]
<0> REMOTE
<1> ¦LOCAL
<2> LOCAL
<3>
<4>
<5>

[166]
<0> PUMP ON
<1> ¦PUMP OFF
<2> PUMP OFF
<3>
<4>
<5>

[167]
<0>
<1> ¦OVERFLOW TRAP LEVEL LO
<2> OVERFLOW TRAP LEVEL LO
<3> OVERFLOW TRAP LEVEL LO
<4> OVERFLOW TRAP LEVEL LO
<5> OVERFLOW TRAP LEVEL LO

[168]
<0>
<1> ¦LOW
<2> LOW

[169]
<0>
<1> ¦LEVEL
<2> LEVEL

[170]
<0>
<1> ¦WATER SERVICE BLDG TEMP LO
<2> WATER SERVICE BLDG TEMP LO
<3> WATER SERVICE BLDG TEMP INHA
<4> WATER SERVICE BLDG TEMP OSCN
<5> WATER SERVICE BLDG TEMP IBAD
<6>
<7>

[171]
<0>
<1> ¦EVAP VESSEL PRESSURE HIGH
<2> EVAP VESSEL PRESSURE HIGH
<3> EVAP VESSEL PRESSURE INHA
<4> EVAP VESSEL PRESSURE OSCN
<5> EVAP VESSEL PRESSURE IBAD
<6>
<7>

[172]
<0>
<1> ¦LEAK
<2> LEAK
<3> LEAK
<4> LEAK
<5> LEAK

[173]
<0>
<1> ¦ON
<2>
<3>
<4>
<5>

[174]
<0>
<1> ¦OVERFLOW TRAP LEVEL LO
<2> OVERFLOW TRAP LEVEL LO
<3> OVERFLOW TRAP LVL INHA
<4> OVERFLOW TRAP LVL OSCN
<5> OVERFLOW TRAP LVL IBAD

[175]
<0>
<1> ¦LINE TRAP LEVEL LO
<2> LINE TRAP LEVEL LO
<3> LINE TRAP LVL INHA
<4> LINE TRAP LVL OSCN
<5> LINE TRAP LVL IBAD

[176]
<0>
<1> ¦TEMPERATURE HI
<2> TEMPERATURE HI
<3> TEMP HI INHA
<4> TEMP HI OSCN
<5> TEMP HI IBAD

[177]
<0>
<1> ¦BROKEN
<2> BROKEN
<3> INHA
<4> OSCN
<5> IBAD

[178]
<0>
<1> ¦DESUP OUTLET TEMP HIGH
<2> DESUP OUTLET TEMP HIGH
<3> DESUP OUTLET TEMP INHA
<4> DESUP OUTLET TEMP OSCN
<5> DESUP OUTLET TEMP IBAD

[179]
<0>
<1> ¦A SN-215 LEAK
<2> A SN-215 LEAK
<3> A SN-215 INHA
<4> ¦A SN-215 OSCN
<5> A SN-215 IBAD

[180]
<0>
<1> ¦A SN-216 LEAK
<2> A SN-216 LEAK
<3> A SN-216 INHA
<4> ¦A SN-216 OSCN
<5> A SN-216 IBAD

[181]
<0>
<1> ¦VALVE FAILED TO CLOSE
<2> VALVE FAILED TO CLOSE
<3>
<4>
<5>

[182]
<0>
<1> RUNNING
<2> ¦TRAPPED
<3> CRITICAL
<4> WAIT
<5> WAITUNTIL
<6> ¦NOT RUNNING
<7> ¦NO STATUS
<8>
<9>
<10>

[183]
<0>
<1> LOCAL

[184]
<0>
<1> ¦TURBIN FAN ON
<2> TURBIN FAN ON
<3> TURBN FN INHA
<4> TURBN FN OSCN
<5> TURBN FN IBAD

[185]
<0>
<1> ¦ERROR

[186]
<0> PERMISSIVE OFF
<1> PERMISSIVE ON

[187]
<0> PUMP OFF
<1> PUMP ON

[188]
<0> INTERLOCK
<1> NORMAL

[189]
<0> BYPASS OFF
<1> BYPASS ON

[190]
<0> NOT OPEN
<1> OPEN

[191]
<0> NOT CLOSED
<1> CLOSED

[192]
<0> LOCAL
<1> REMOTE

[193]
<0> OFF
<1> ON

[194]
<0> PRMSSV OFF
<1> PERMISSIVE

[195]
<0> STOP/RESET
<1> START

[196]
<0>
<1> ¦TIMEOUT
<2>

[197]
<0> TIMER OFF
<1> TIMER ON

[198]
<0> POSN-1
<1> NOT POS1

[199]
<0> POSN-2
<1> NOT POS2

[200]
<0> NOT DVRT
<1> DIVERTED

[201]
<0> NOT NORM
<1> TO WEIR

[202]
<0> NOT NORM
<1> TO BASIN

[203]
<0> DIVERTED
<1> NORMAL

[204]
<0>
<1> ¦SAMPLER LOW FLOW
<2> SAMPLER LOW FLOW
<3> SAMPLER LOW FLOW
<4> SAMPLER LOW FLOW
<5> SAMPLER LOW FLOW

[205]
<0>
<1> ¦RAD SYSTEM FAIL
<2> RAD SYSTEM FAIL
<3> RAD SYSTEM FAIL
<4> RAD SYSTEM FAIL
<5> RAD SYSTEM FAIL

[206]
<0>
<1> ¦PUMP ENCODER LOSS
<2> PUMP ENCODER LOSS
<3> PUMP ENCODER INHA
<4> PUMP ENCODER OSCN
<5> PUMP ENCODER IBAD

[207]
<0>
<1> ¦VAR FREQ DRIVE FAULT
<2> VAR FREQ DRIVE FAULT
<3> VAR FREQ DRIVE INHA
<4> VAR FREQ DRIVE OSCN
<5> VAR FREQ DRIVE IBAD

[208]
<0>
<1> ¦PUMP ZERO SPEED
<2> PUMP ZERO SPEED
<3> PUMP SPEED INHA
<4> PUMP SPEED OSCN
<5> PUMP SPEED IBAD

[209]
<0> EMERGENCY POWER TRANSFR SWITCH OFF
<1> ¦EMERGENCY POWER TRANSFER SWITCH ON
<2> EMERGENCY POWER TRANSFER SWITCH ON
<3> EMERGENCY POWER XFR SWITCH INHA
<4> EMERGENCY POWER XFR SWITCH DIS
<5> EMERGENCY POWER XFR SWITCH IBAD

[210]
<0>
<1> ¦LOW
<2> LOW
<3> INHA
<4> OSCN
<5> IBAD

[211]
<0>
<1> ¦LEVEL
<2> LEVEL
<3>
<4>
<5>

[212]
<0> SD-RESET

[213]
<0> SHUTDOWN

[214]
<0> INTERLOK

[215]
<0>
<1> ¦FAULT
<2> FAULT

[216]
<0> CF-OFF
<1> CF-ON
<2> SD-RESET
<3>
<4> SHUTDOWN
<5> INTERLOCK
<6> STOPPING
<7> STARTING

[217]
<0>
<1>
<2>
<3> INHA
<4> ¦DIS
<5> IBAD

[218]
<0>
<1> ¦LO LEVEL
<2> LO LEVEL
<3> LO INHA
<4> ¦LO DIS
<5> LO IBAD

[219]
<0>
<1> ¦SHUTDOWN
<2> SHUTDOWN
<3> SHUTDOWN
<4> SHUTDOWN
<5> SHUTDOWN

[220]
<0>
<1> ¦INLET PRESS HI
<2> INLET PRESS HI
<3> INLET PRESS HI
<4> INLET PRESS HI
<5> INLET PRESS HI

[221]
<0> TIMER OFF
<1> ¦TIMER ON

[222]
<0> MANUAL
<1> AUTO

[223]
<0> NOT SELECTED
<1> SELECTED
<2> ¦OFFLINE
<3>
<4>
<5>

[224]
<0> VESSEL VENT RADIATION SYSTEM NORMAL
<1> ¦VESSEL VENT RADIATION SYSTEM FAILURE
<2> VESSEL VENT RADIATION SYSTEM FAILURE
<3> VESSEL VENT RADIATION SYSTEM INHIBIT
<4> VESSEL VENT RADIATION SYSTEM OFF SCN
<5> VESSEL VENT RADIATION SYSTEM BAD INP

[225]
<0> CLOSED
<1> OPEN

[226]
<0>
<1> ¦242-A TO L.E.R.F. LEAK
<2> 242-A TO L.E.R.F. LEAK
<3> 242-A TO L.E.R.F. LEAK
<4> 242-A TO L.E.R.F. LEAK
<5> 242-A TO L.E.R.F. LEAK

[227]
<0> OFF
<1> ¦ON
<2> ON
<3>
<4>
<5>

[228]
<0>
<1> *
<2> *
<3>
<4>
<5>

[229]
<0> OFF
<1> ON
<2> ON
<3> INHA
<4> DIS
<5> IBAD

[230]
<0>
<1> ¦CF-OFF
<2> CF-OFF

[231]
<0> UPS AC INPUT NORM
<1> ¦UPS AC INPUT FAIL
<2> UPS AC INPUT FAIL
<3> UPS AC INPUT INHA
<4> UPS AC INPUT DIS
<5> UPS AC INPUT IBAD
<6>

[232]
<0> UPS BATTERY NORM
<1> ¦UPS BATTERY LOW
<2> UPS BATTERY LOW
<3> UPS BATTERY INHA
<4> UPS BATTERY DIS
<5> UPS BATTERY IBAD
<6>

[233]
<0> UPS BYP SWITCH NORMAL
<1> ¦UPS BYP SWITCH BYPASS
<2> UPS BYP SWITCH BYPASS
<3> UPS BYP SWITCH INHA
<4> UPS BYP SWITCH DIS
<5> UPS BYP SWITCH IBAD
<6>

[234]
<0> ON
<1> OFF
<2> OFF
<3> INHA
<4> DIS
<5> IBAD

[235]
<0>
<1> ¦AREA RAD MON FAIL
<2> AREA RAD MON FAIL
<3> AREA RAD MON INHA
<4> AREA RAD MON DIS
<5> AREA RAD MON IBAD

[236]
<0>
<1> ¦AIR SAMP MON FAIL
<2> AIR SAMP MON FAIL
<3> AIR SAMP MON INHA
<4> AIR SAMP MON DIS
<5> AIR SAMP MON IBAD

[237]
<0>
<1> ¦AS RADIATION HI
<2> AS RADIATION HI
<3> AS RADN HI INHA
<4> AS RADN HI DIS
<5> AS RADN HI IBAD

[238]
<0>
<1> ¦AREA RADIATION HI
<2> AREA RADIATION HI
<3> AREA RADN HI INHA
<4> AREA RADN HI DIS
<5> AREA RADN HI IBAD

[239]
<0> PRESSURE NORM
<1> ¦PRESSURE HIGH
<2> PRESSURE HIGH
<3> PRESS HI INHA
<4> PRESS HI DIS
<5> PRESS HI IBAD

[240]
<0>
<1> ¦LERF MODEM COMM FAIL
<2> LERF MODEM COMM FAIL
<3> LERF MODEM FAIL INHA
<4> LERF MODEM FAIL DIS
<5> LERF MODEM FAIL IBAD

[241]
<0> ON
<1> ¦OFF
<2> OFF
<3>
<4>
<5>

[242]
<0>
<1> ¦RW ON
<2> RW ON
<3>   INHA
<4>   DIS
<5>   IBAD

[243]
<0>
<1> ¦SHUTDOWN
<2>
<3> SHUTDOWN
<4> SHUTDOWN
<5> SHUTDOWN

[244]
<0>
<1> ¦ANNULUS EXH RADIATION HI
<2> ANNULUS EXH RADIATION HI
<3> ANNULUS EXH RADN INHA
<4> ANNULUS EXH RADN DIS
<5> ANNULUS EXH RADN IBAD

[245]
<0>
<1> ¦LEAK DETECT PUMP PIT LEAK
<2> LEAK DETECT PUMP PIT LEAK
<3> LEAK DETECT PUMP PIT INHA
<4> LEAK DETECT PUMP PIT DIS
<5> LEAK DETECT PUMP PIT IBAD

[246]
<0>
<1> ¦WATER SERVICE BLDG TEMP LO
<2> WATER SERVICE BLDG TEMP LO
<3> WATER SERVICE BLDG TEMP INHA
<4> WATER SERVICE BLDG TEMP OSCN
<5> WATER SERVICE BLDG TEMP IBAD
<6>
<7>

THIS PAGE LEFT BLANK INTENTIONALLY

APPENDIX H - REBUILD PROCEDURE

INTRODUCTION

The 242-A control room is run by a Distributive Controls System (DCS) designed by Texas Instruments (TI) now known as GSE Process Solutions Inc. (GSE). The system runs on two redundant VAX 4000-105A computers located in the control room. The operating system on the computers is VMS version 5.5-2AH. The TI software is called D3 and is version 6.3-2. The two computers are know as CDCM and DCM0. The disk on both machines is DKA300: and is organized as follows. The D3 source code is located in a directory called D3SOURCE: and the site specific files are located in the directory called D3DATA:. To properly rebuild the system a few VMS commands are good to know. First to change you place iOn the directory structure the command SET DEFAULT is used or SET DEF for short. To change to the directory D3DATA: you would type SET DEF D3DATA:[000000]. Since D3DATA is a logical pointing to the a specific place on the directory structure it will appear that now the drive is D3DATA:. If you want to see where you are located in the directory structure you would type SHOW DEF. You will see D3DATA:[000000]. The six zero imply that you are at the top of that directory. Within the D3DATA directory you will see many other directory by typing DIRECTORY or DIR for short. The directories will be 6 digit numbers. The information for building the Continuous Control Database is located in directory [100001] or [100,1]. To change to that directory you would type SET DEF [100001]. A SHOW DEF command you display D3DATA:[100001] for your location on the disk. Another useful way to use the DIR command is to search for a file on the disk when you have forgotten where it is located. If you go to the top of the directory structure by typing SET DEF DKA300:[000000] you can then look for a file anywhere on the disk by typing DIR [*...]file-name. If you would like to search all .COM files to see which file contains a phrase you are looking for you can use the SEARCH or SEA command. From the top of the directory structure you could type SEA [*...]*.COM PCM0A. This would search in every .COM file on the disk and report back every file that contains the word PCM0A. Other commands that may be useful are COPY, MOVE, RENAME, and DELETE. All these file work as they do for IBM DOS.

Command files are used to rebuild the system. They are files that end in .COM and can be found through out the directory structure. These file are described through out this document. For the most part the command files for rebuilding the system are in D3DATA:[001001].

SYSTEM REBUILD PROCEDURE

1. REBUILD ALL OR PORTIONS OF CONTINUOUS DATABASE:

   a. REBUILD A PORTION - for example redefine an EPN:

      1. Write command file to report out existing EPN you wish to redefine. The command file content should be as follows:

         KILL (prevents new version of UDB)
         NOCHECK (avoids rebuild of SDB, SEQ, CGR)
         REP <EPN> (do this for each EPN)
         EXIT (exits MDL utility)

      2. Run the command file with MODEL to report out the EPNs to be modified. The command format to run MODEL should be as follows:

         MDL PCM__ -I <infile name.CMD> -O <outfile name.OUT>

         where PCM__ is the name of the PCM database.

      3. Modify the output (.OUT) file. I suggest that the modified file be named with ".INP" as the file name extension. This will differentiate the old from the new and will result in concise records of both configurations. After modifying, make sure of the following:

         a. The .INP file uses the MODEL commands KILL, NOCHECK, and EXIT.

         b. All REPs replaced with DELs.

         c. Analogs are redefined before MDVs and DEVs, in case the analogs "forward reference" the DEVs.

         d. DEVs are deleted first followed by MDVs, and that the MDVs are redefined before the DEVs.

         e. Some MDVs are associated with more then one DEV. So make sure all DEVs associated with a common MDV are redefined.

4. Run local MODEL with KILL on for error checking and debugging. With KILL on, a new database is not created. If MDL is run without KILL on and errors are generated, then the new database will be corrupted. I suggest writing the output file of this session to a file name having the .ERR extension, denoting error output file. For example:

   MDL PCM__ -I <infile name.INP> -O <outfile name.ERR>

   Pay attention to the CRT as MODEL runs as not all errors get written to the error output file, although most do. View the error file for errors. If no errors are indicated at the 'end of file', then proceed as long as the CRT did not display any errors.

5. EDT the .INP file to remove the KILL command.

6. Run Remote MODEL (CAUTION !!!) with KILL off again using the .ERR output file specification:

   MDL -R PCM__ -I <infile name.INP> -O <outfile name.ERR>

7. View the error file. If all looks well and this is a permanent change, modify the source input file for the modified database. The source input file for PCM0A PCM0B are in D3DATA:[100001]. They have .INP file extensions. All pertinent PCM0A text files are also in D3DATA:[100001] Otherwise just copy an EPN to create a new one.

8. With NOCHECK on, the sequence database, sequence programs, graphics, other displays, and c-code do not need to be rebuilt. But the display database may need to be reloaded to the DCMs (use D3DATA:[001001]DBL.COM, or a select part of it).

b. REBUILD ALL - that is, rebuild the system.

1. Use the latest PCM__.INP input files in D3DATA:[100001]. These use the CREATE model command. Do not use remote model. Do not error check with KILL on - MODEL ignores KILL when CREATE is used. If at first you don't succeed, edit the input file and try again. Run local model, which only creates a new disk version:

   MDL PCM__ -I <infile name.INP> -O <outfile name.ERR>

You can go to D3DATA:[001001] and use command files if creating a new database. The command @BLDALLPCMS will build both PCM0A and PCM1A databases using PCM0A.INP and PCM1A.INP files. It will also create the .ERR files and locate them in D3DATA:[100001]. Command file @BLDPCM0A and @BLDPCM1A will build just the respective databases. These command files remove all display databases from the CDCM only. Then runs MDL with PCM0A and PCM1A as input and creates the .ERR files.

2. After model is done, load the display database to the Host only! This is necessary to rebuild SABL, graphics, and other displays on the Host. Do not load to DCM0 yet to minimize impact to the control room. Use the following DBL commands. From now on CDCM OCM displays may be out to lunch, but the DCM0 will be unaffected. Or use command file @LOADDISDBCDCM located in D3DATA:[001001].

```
DBL -H -RD PCM0A
DBL -H -D PCM0A
DBL -H -RD PCM1A
DBL -H -D PCM1A
```

3. Rebuild sequence database:

   a. In D3DATA:[300001], if just rebuilding PCM0A OR PCM1A copy (NOT rename) SDBMASTER.XRF and SDBSHORT.XRF to the same file name but with .OLD extension. When SDB utility runs it overwrites the .XRF files (the old XRF files will be lost).

   b. Rebuild the Sequence databases in D3DATA:[001001]:

      For PCM0A: SDB P0ASDB (runs P0ASDB.COM)
      For PCM1A: SDB P1ASDB (runs P1ASDB.COM)

      [Note - to build all sdb files use D3DATA:[001001]BLDRMS. To build just the sdb you may use D3DATA:[001001]BLDSDB. For all PCM0, you may use D3DATA:[001001]P0ASDB. For all PCM1, you may use D3DATA:[001001]P1ASDB. There also is a D3DATA:[001001]BLDSYS file. Look at these and use what meets your needs at the time.]

      These command files will copy the SDBMANSTER.XRF and SDBSHORT.XRF to *.OLD. These command files use input files located in D3DATA:[300001] called 242A.CMD, ALL0.CMD, ALL1.CMD, BANKK.CMD, EMTY.CMD, FARM.CMD, OVRVEW.CMD, PCM0A.CMD, PCM1A.CMD, PROESS.CMD, SHORT0.CMD, TNKFRM.CMD, TOTAL0.CMD, and TATLA1.CMD as necessary.

4. Rebuild sequence programs in D3DATA:[001001]:

   a. For PCM0A: @SICPCM0APROGS
   b. For PCM1A: @SICPCM1APROGS
   c. For BOTH: @SICALLPROGS

5. Rebuild graphics in D3DATA:[001001]:

   a. For PCM0A: @COMPILE_PCM0A
   b. For PCM1A: @COMPILE_PCM1A
   c. For BOTH: @COMPILE_GRAPHICS

6. Rebuild Group Displays (POV, EPN, TRN, HIS) in D3DATA:[001001]:

   a. To build: @BLD_GROUPS

   This command file first stops the current trends and stops the historical trends.

7. Now you are ready to load the PCMs. Crash the PCMs by pressing the RESET switch on the PCMs. Reset and load only one redundant PCM pair at a time (PCM0 OR PCM1).

   D3DATA:[001001]LOADPCM0A, or
   D3DATA:[001001]LOADPCM1A

   Answer the prompts and then everything is automatically loaded to the PCMs, including sequence programs, and runs them. Don't try to do anything else. Just watch the screen to make sure everything loads and runs properly. Repeat for the other redundant pcm pair, if necessary. The two command files above start command procedure PCMLOD.COM located in D3SOURCE:[001003]. This command procedure runs four different command procedures located in D3SOURCE:[001001] which load and run all the sequence programs. These command procedures are SEQLD0A.COM, SEQLD0B.COM, SEQLD1A.COM, and SEQLD1B.COM.

8. At this point, the PCMs are loaded and the Host has the new display database loaded to it. Load the files to the DCM0 by using command file D3DATA:[001001]LOADDCM. But nothing else is loaded to the Host. So finish loading the Host by re-booting it as follows:

   a. Enter D3DOWN and answer the prompts.
   b. Enter SHUTDOWN to stop the system.
   c. Press the halt button on the back of the CDCM. Turn the power witch off on the Q buss for the CDCM then turn the power off on the CDCM.
   d. Turn the power switch on the Q buss first then the power switch on the back of the CDCM.
   e. When the power comes on the CDCM will reboot. When the prompt >>> comes on the CDCM alarm typer enter a "b" and hit enter.
   f. When the system is up you can log back on a CDCM terminal.
   g. After logging on type D3UP to start the D3 software.

9. Now the CDCM displays, for the most part, should be functional. There may be a few straggler displays that need to be re-compiled, but don't worry about these at the moment. Also, the DCM0 displays are now out to lunch. Notice how the operator for the most part has had at least one console with completely functional displays.

10. Reboot the DCM0 as follows:

    a. Enter D3DOWN and answer the prompts.
    b. Enter SHUTDOWN to stop the system.
    c. Press the halt button on the back of the DCM0. Turn the power switch off on the Q buss for the DCM0 then turn the power off on the DCM0..
    d. Turn the power switch on the Q buss first then the power switch on the back of the DCM0.
    e. When the power comes on the CDCM will reboot. When the prompt >>> comes on the DCM0 alarm typer enter a "b" and hit enter.
    f. When the system is up you can log back on a DCM0 terminal.
    g. After logging on type D3UP to start the D3 software.

    Now DCM0 is back to normal. All that is left is some miscellaneous rebuilds and cleanup.

11. Rebuild C-code, even if IPNs have not changed. The source code for the C-code is located in D3SOURCE:[261010] or [MICRO.200010]. The *.SCR files will place the executable files in the appropriate directories such as D3SOURCE:[003055], [004055], and [005055], The LTK files loaded to the micros are in D3SOURCE:[001055]. To build an individual C-program, set default to D3SOURCE:[261010] or [MICRO.200010] and type the following:

    BLD FILE.SCR -L -ALL -MOV -LIS

    D3SOURCE:[261010]:
    WATCH, FQIRESET, INHA, BALANCE, TFBALANCE, HISTDAT, UKF14, UKF34, UKF43, UKF44, etc)

    [MICRO.200010]:
    DOG0A, DOG0B, DOG1A, and DOG1B.

    To build all the C-code type the use the following command file located in D3DATA:[001001]:BLDALLC

12. If IPNs for DEVs or MDVs have changed, then make sure the key task (D3SOURCE:[160010]KEY.C, KEY.TSK) are correct for:

    AUDALARM0 (pcm0a dev ipn 171)
    HS-AB1ACK (pcm0a dev ipn 208)
    HS-AB2ACK (pcm1a dev ipn 15)

13. Reload C-code to PCMs (MICRO:[377001]DOG0, DOG1) if necessary. Use command file located in D3DATA:[001001]DOGSWAP.

14. Reinstall (reboot if you like) C-code to DCMs if necessary.

15. Rebuild the Report Writer reports on each DCM: (NOTE: report writer is no longer used)

CDCM	DCM0	
ROUTINEG1	ROUTINED1	ROUTINES1
ROUTINEG2	ROUTINED2	ROUTINES2
ROUTINEG3	ROUTINED3	ROUTINES3
ROUTINEG4	ROUTINED4	ROUTINES4
TOTALIZRS		

    This is done by recompiling these using RGE in [244,2].

16. Purge the old files away as required:

```
D3DATA:[100001]*.UDB, *.REF
D3DATA:[300001]PCM*SDB.LST
D3DATA:[300001]*.SEQ, *.LST, *.LTQ
D3DATA:[100010]GRA*.BIN, *.POV, *.EPN, *.TRN, *.HIS, *.COL
D3DATA:[100010]GRA*.CGR
D3SOURCE:[261010]name.C
[MICRO.200010]*.C
```

To purge all these directories above use command procedure in D3DATA:[001001]PGALL.

17. The String Editor can now only handle an input file of 50 string packets. Since there are about 274 string packets on this system, the input file had to be split into five separate files located in D3DATA:[100010]. They are STRING1.TXT, STRING2.TXT, STRING3.TXT, STRING4.TXT, and STRING5.TXT. Command procedure STRING_BUILD.COM is located in D3DATA:[100010] and will build one string file called STRING.BIN. After making any changes to the string any graphics that use the string changed will have to be recompiled and copied to the DCM0.

18. If all input files have been proporly changed, the way to build all inputs and reload everything is as follows:

```
SET DEF D3DATA:[001001]
@BLDALLPCMS                                    3:30
@LOADDISDBCDCM                                  :10
@BLDRMS                                        6:15
@SICALLPROGS                                   7:30
@COMPILE_GRAPHICS                              8:18
@BLD_GROUPS                                    2:30
CRASH BOTH PCMS AND TURN THEM BACK ON.         0:20
(Wait until the pcms are in load request)      1:30
@LOADPCM0A                                     2:30
@LOADPCM1A                                     2:30
@DCMLOAD                                       3:30
DO A D3DOWN AND D3UP ON BOTH COMPUTERS         5:00
                                              39:273
```

APPENDIX I - SKID KEY CONFIGURATION

Listing of D3DATA:[100010]CDCMSK0.CMD on the CDCM
(Other SKID command files are identical)

KEY: 0
LABEL: AN:271
VERIFICATION: N
ECHO: AN-271
DISPLAY TYPE: GR
DISPLAY NUMBER: 274
LED MODE: NONE
FUNCTION:
KEY: 1
LABEL: AN:VALVE:PIT
VERIFICATION: N
ECHO: AN VALVE PIT
DISPLAY TYPE: GR
DISPLAY NUMBER: 272
LED MODE: NONE
FUNCTION:
KEY: 2
LABEL: AN:HVAC
VERIFICATION: N
ECHO: AN HVAC
DISPLAY TYPE: GR
DISPLAY NUMBER: 273
LED MODE: NONE
FUNCTION:
KEY: 3
LABEL: AN:FARM
VERIFICATION: N
ECHO: AN FARM
DISPLAY TYPE: GR
DISPLAY NUMBER: 270
LED MODE: NONE
FUNCTION:
KEY: 4
LABEL: AP:271
VERIFICATION: N
ECHO: AP-271
DISPLAY TYPE: GR
DISPLAY NUMBER: 295
LED MODE: NONE
FUNCTION:
KEY: 5
LABEL: AP:VALVE:PIT
VERIFICATION: N
ECHO: AP VALVE PIT
DISPLAY TYPE: GR
DISPLAY NUMBER: 292
LED MODE: NONE
FUNCTION:
KEY: 6
LABEL: AP:HVAC
VERIFICATION: N
ECHO: AP HVAC
DISPLAY TYPE: GR
DISPLAY NUMBER: 294
LED MODE: NONE
FUNCTION:
KEY: 7
LABEL: AP:FARM
VERIFICATION: N
ECHO: AP FARM
DISPLAY TYPE: GR
DISPLAY NUMBER: 290
LED MODE: NONE
FUNCTION:
KEY: 8
LABEL: AW:271
VERIFICATION: N
ECHO: AW-271
DISPLAY TYPE: GR
DISPLAY NUMBER: 302
LED MODE: NONE
FUNCTION:
KEY: 9
LABEL: AW:VALVE:PIT
VERIFICATION: N
ECHO: AW VALVE PIT
DISPLAY TYPE: GR
DISPLAY NUMBER: 303
LED MODE: NONE
FUNCTION:
KEY: 10
LABEL: AW:HVAC
VERIFICATION: N
ECHO: AW HVAC
DISPLAY TYPE: GR
DISPLAY NUMBER: 304
LED MODE: NONE
FUNCTION:
KEY: 11
LABEL: AW:FARM
VERIFICATION: N
ECHO: AW FARM
DISPLAY TYPE: GR
DISPLAY NUMBER: 300
LED MODE: NONE
FUNCTION:
KEY: 12
LABEL: AZ:FARM
VERIFICATION: N
ECHO: AZ FARM

DISPLAY TYPE: GR
DISPLAY NUMBER: 323
LED MODE: NONE
FUNCTION:
KEY: 13
LABEL: AY:FARM
VERIFICATION: N
ECHO: AY FARM
DISPLAY TYPE: GR
DISPLAY NUMBER: 322
LED MODE: NONE
FUNCTION:
KEY: 14
LABEL: A:FARM
VERIFICATION: N
ECHO: A FARM
DISPLAY TYPE: GR
DISPLAY NUMBER: 321
LED MODE: NONE
FUNCTION:
KEY: 15
LABEL: A AX:AY AZ;FARMS
VERIFICATION: N
ECHO: A AX AY AZ FARMS
DISPLAY TYPE: GR
DISPLAY NUMBER: 320
LED MODE: NONE
FUNCTION:
KEY: 16
LABEL: WASTE:XFR
VERIFICATION: N
ECHO: WASTE TRANSFER
DISPLAY TYPE: GR
DISPLAY NUMBER: 344
LED MODE: NONE
FUNCTION:
KEY: 17
LABEL: A-350
VERIFICATION: N
ECHO: A-350
DISPLAY TYPE: GR
DISPLAY NUMBER: 343
LED MODE: NONE
FUNCTION:
KEY: 18
LABEL: 244-A:HVAC
VERIFICATION: N
ECHO: 244-A HVAC
DISPLAY TYPE: GR
DISPLAY NUMBER: 341
LED MODE: NONE
FUNCTION:
KEY: 19
LABEL: 244-A
VERIFICATION: N
ECHO: 244-A
DISPLAY TYPE: GR
DISPLAY NUMBER: 340
LED MODE: NONE
FUNCTION:
KEY: 20
LABEL:
VERIFICATION: N
ECHO:
DISPLAY TYPE: NONE
LED MODE: NONE
FUNCTION:
KEY: 21
LABEL:
VERIFICATION: N
ECHO:
DISPLAY TYPE: NONE
LED MODE: NONE
FUNCTION:
KEY: 22
LABEL: AP-102:GROUT
VERIFICATION: N
ECHO: AP-102 GROUT TANK
DISPLAY TYPE: GR
DISPLAY NUMBER: 291
LED MODE: NONE
FUNCTION:
KEY: 23
LABEL: 204AR:& MISC
VERIFICATION: N
ECHO: 204-AR ALARMS
DISPLAY TYPE: FP
DISPLAY NUMBER: 81
LED MODE: NONE
FUNCTION:
KEY: 24
LABEL:
VERIFICATION: N
ECHO:
DISPLAY TYPE: NONE
LED MODE: NONE
FUNCTION:
KEY: 25
LABEL:
VERIFICATION: N

ECHO:
DISPLAY TYPE: NONE
LED MODE: NONE
FUNCTION:
KEY: 26
LABEL:
VERIFICATION: N
ECHO:
DISPLAY TYPE: NONE
LED MODE: NONE
FUNCTION:
KEY: 27
LABEL:
VERIFICATION: N
ECHO:
DISPLAY TYPE: NONE
LED MODE: NONE
FUNCTION:
KEY: 28
LABEL: SC:BASIN
VERIFICATION: N
ECHO: STEAM CONDENSATE BASINS
DISPLAY TYPE: GR
DISPLAY NUMBER: 22
LED MODE: NONE
FUNCTION:
KEY: 29
LABEL: STEAM:CNDS8
VERIFICATION: N
ECHO: STEAM CONDENSATE
DISPLAY TYPE: GR
DISPLAY NUMBER: 14
LED MODE: NONE
FUNCTION:
KEY: 30
LABEL: PB-1
VERIFICATION: N
ECHO: PB-1 PUMP
DISPLAY TYPE: GR
DISPLAY NUMBER: 12
LED MODE: NONE
FUNCTION:
KEY: 31
LABEL: PUMP:AW-102
VERIFICATION: N
ECHO: AW-102 FEED PUMP
DISPLAY TYPE: GR
DISPLAY NUMBER: 301
LED MODE: NONE
FUNCTION:
KEY: 32
LABEL:
VERIFICATION: N
ECHO:
DISPLAY TYPE: NONE
LED MODE: NONE
FUNCTION:
KEY: 33
LABEL: E-A-1
VERIFICATION: N
ECHO: E-A-1 REBOILER
DISPLAY TYPE: GR
DISPLAY NUMBER: 13
LED MODE: NONE
FUNCTION:
KEY: 34
LABEL: C-A-1
VERIFICATION: N
ECHO: C-A-1 EVAPORATOR
DISPLAY TYPE: GR
DISPLAY NUMBER: 10
LED MODE: NONE
FUNCTION:
KEY: 35
LABEL: PB-2
VERIFICATION: N
ECHO: PB-2 PUMP
DISPLAY TYPE: GR
DISPLAY NUMBER: 15
LED MODE: NONE
FUNCTION:
KEY: 36
LABEL: IX-D-1
VERIFICATION: N
ECHO: IX-D-1 / RC3
DISPLAY TYPE: GR
DISPLAY NUMBER: 44
LED MODE: NONE
FUNCTION:
KEY: 37
LABEL: TK:C-100
VERIFICATION: N
ECHO: TK-C-100
DISPLAY TYPE: GR
DISPLAY NUMBER: 18
LED MODE: NONE
FUNCTION:
KEY: 38
LABEL: E-C-1:COND
VERIFICATION: N

ECHO: E-C-1 VACUUM CONDENSER
DISPLAY TYPE: GR
DISPLAY NUMBER: 16
LED MODE: NONE
FUNCTION:
KEY: 39
LABEL: VESSL:VENT
VERIFICATION: N
ECHO: VESSEL VENT
DISPLAY TYPE: GR
DISPLAY NUMBER: 20
LED MODE: NONE
FUNCTION:
KEY: 40
LABEL: LERF
VERIFICATION: N
ECHO: L.E.R.F.
DISPLAY TYPE: GR
DISPLAY NUMBER: 24
LED MODE: NONE
FUNCTION:
KEY: 41
LABEL:
VERIFICATION: N
ECHO:
DISPLAY TYPE: NONE
LED MODE: NONE
FUNCTION:
KEY: 42
LABEL: USED:RAW:WATER
VERIFICATION: N
ECHO: USED RAW WATER
DISPLAY TYPE: GR
DISPLAY NUMBER: 17
LED MODE: NONE
FUNCTION:
KEY: 43
LABEL: RAW:WATER
VERIFICATION: N
ECHO: RAW WATER SUPPLY
DISPLAY TYPE: GR
DISPLAY NUMBER: 11
LED MODE: NONE
FUNCTION:
KEY: 44
LABEL: CASS
VERIFICATION: N
ECHO: CASS ALARMS
DISPLAY TYPE: GR
DISPLAY NUMBER: 90
LED MODE: NONE
FUNCTION:
KEY: 45
LABEL: AMU
VERIFICATION: N
ECHO: AQUEOUS MAKE UP
DISPLAY TYPE: GR
DISPLAY NUMBER: 19
LED MODE: NONE
FUNCTION:
KEY: 46
LABEL: UTILS
VERIFICATION: N
ECHO: UTILITIES
DISPLAY TYPE: GR
DISPLAY NUMBER: 21
LED MODE: NONE
FUNCTION:
KEY: 47
LABEL: 242A:RADS
VERIFICATION: N
ECHO: 242-A RADIATION / VENT
DISPLAY TYPE: GR
DISPLAY NUMBER: 23
LED MODE: NONE
FUNCTION:
KEY: 48
LABEL: PROGRM:STATUS
VERIFICATION: N
ECHO: SEQUENCE PROGRAM STATUS
DISPLAY TYPE: GR
DISPLAY NUMBER: 43
LED MODE: NONE
FUNCTION:
KEY: 49
LABEL: CNDS8:OVER-:VIEW
VERIFICATION: N
ECHO: CONDENSATE OVERVIEW
DISPLAY TYPE: GR
DISPLAY NUMBER: 42
LED MODE: NONE
FUNCTION:
KEY: 50
LABEL: VACUM:OVER-:VIEW
VERIFICATION: N
ECHO: VACUUM CONDENSR OVERVIEW
DISPLAY TYPE: GR
DISPLAY NUMBER: 41
LED MODE: NONE
FUNCTION:

KEY: 51
LABEL: EVAP:OVER-:VIEW
VERIFICATION: N
ECHO: EVAPORATOR OVERVIEW
DISPLAY TYPE: GR
DISPLAY NUMBER: 40
LED MODE: NONE
FUNCTION:
KEY: 52
LABEL:
VERIFICATION: N
ECHO:
DISPLAY TYPE: NONE
LED MODE: NONE
FUNCTION:
KEY: 53
LABEL: INHA
VERIFICATION: N
ECHO: INHIBITED ALARMS
DISPLAY TYPE: GR
DISPLAY NUMBER: 80
LED MODE: NONE
FUNCTION: ES
UNIT NAME: WHOLE0
PROGRAM NAME: TIMEINHA
FUNCTION: EL
NAME: INHALM
FUNCTION:
KEY: 54
LABEL: HISTL:TREND:MENU
VERIFICATION: N
ECHO: HISTORICAL TREND MENU
DISPLAY TYPE: GR
DISPLAY NUMBER: 9
LED MODE: NONE
FUNCTION:
KEY: 55
LABEL: CURR:TREND:MENU
VERIFICATION: N
ECHO: CURRENT TREND MENU
DISPLAY TYPE: GR
DISPLAY NUMBER: 8
LED MODE: NONE
FUNCTION:
KEY: 56
LABEL:
VERIFICATION: N
ECHO:
DISPLAY TYPE: NONE
LED MODE: NONE
FUNCTION:
KEY: 57
LABEL:
VERIFICATION: N
ECHO:
DISPLAY TYPE: NONE
LED MODE: NONE
FUNCTION:
KEY: 58
LABEL:
VERIFICATION: N
ECHO:
DISPLAY TYPE: NONE
LED MODE: NONE
FUNCTION:
KEY: 59
LABEL: ROUNDS:MENU
VERIFICATION: N
ECHO: ROUNDS MENU
DISPLAY TYPE: GR
DISPLAY NUMBER: 501
LED MODE: NONE
FUNCTION:
KEY: 60
LABEL: WHITE:ALARM
VERIFICATION: N
ECHO: WHITE ALARMS
DISPLAY TYPE: GR
DISPLAY NUMBER: 100
LED MODE: NONE
FUNCTION: MS
UNIT NAME: WHOLE1
TYPE: S
VARIABLE NAME: FTRIP0
STATUS: ON
FUNCTION:
KEY: 61
LABEL:
VERIFICATION: N
ECHO: GRAPHICS NOTES
DISPLAY TYPE: GF
DISPLAY NUMBER: 4
LED MODE: NONE
FUNCTION:
KEY: 62
LABEL: EPN:ON/OFF
VERIFICATION: N
ECHO: EPN ON/OFF
DISPLAY TYPE: NONE
LED MODE: NONE

FUNCTION: ES
UNIT NAME: WHOLEO
PROGRAM NAME: TAG
FUNCTION:
KEY: 63
LABEL:
VERIFICATION: N
ECHO:
DISPLAY TYPE: NONE
LED MODE: NONE
FUNCTION:
KEY: