DOE / EV-0016

Subroutines for CAMAC*

Prepared and Adopted by

U.S. NIM Committee
(National Instrumentation Methods)

and ESONE Committee of European Laboratories
(European Standards on Nuclear Electronics)

June 1978

U.S. Department of Energy
Assistant Secretary for Environment
Division of Biomedical and Environmental Research

*Computer Automated Measurement and Control

DISCLAIMER

DOE/EV-0016

UC-37

Subroutines for CAMAC *

Prepared and Adopted by

U.S. NIM Committee
(National Instrumentation Methods)

and ESONE Committee of European Laboratories
(European Standards on Nuclear Electronics)

June 1978

U.S. Department of Energy
Assistant Secretary for Environment
Division of Biomedical and Environmental Research
Washington, D.C. 20545

* Computer Automated Measurement and Control

NOTICE

Available from:

National Technical Information Service (NTIS)
U.S. Department of Commerce
5285 Port Royal Road
Springfield, Virginia 22161

Price: Printed Copy: $ 4.50
Microfiche: $ 3.00

For sale by the Superintendent of Documents, U.S. Government Printing Office
Washington, D.C. 20402

Stock No. 061-000-00104-2

ABSTRACT and FOREWORD

This report† presents a recommended set of software subroutines prepared by the Software Working Group of the U.S. Department of Energy NIM Committee* and the ESONE Committee** of European Laboratories for use with the CAMAC modular instrumentation and interface system of IEEE Standard 583-1975 (EUR 4100e). These subroutines provide a general capability for communicating with CAMAC systems. They will be of primary interest to those who wish to write their own data-processing programs in a high-level programming language, such as Fortran. The achievable data transfer rate is, of course, dependent on a number of factors, including the language used, the operating system, the compiler, the method and level of subroutine implementation and the computer.

†Questions regarding this document should be directed to

R. F. Thomas, Jr., Lawrence Berkeley Laboratory, University of California, Berkeley, CA 94720, U.S.A.
W. K. Dawson, Nuclear Research Center, University of Alberta, Edmonton, Alberta, CANADA T6G 2N5
A. J. Cox, University of Oxford, Department of Engineering Science, Parks Road, OXFORD OX1 3PJ ENGLAND
P. Clout, European Microbiology Laboratory, DESY Notkestrasse 85, D-2000 Hamburg 52, GERMANY

KEY WORDS

CAMAC
Fortran
Software
Standard

THIS PAGE

WAS INTENTIONALLY

LEFT BLANK

CAMAC SPECIFICATIONS AND REPORTS

Title	IEEE, ANSI Std No.	IEC No.	DOE No.	EURATOM (EUR) No. or ESONE No.
Modular Instrumentation and Digital Interface System (CAMAC)	IEEE 583-1975* ANSI/IEEE 583-1975*	516	**	EUR 4100e
Serial Highway Interface System (CAMAC)	IEEE 595-1976* ANSI/IEEE 595-1976*	†	**	EUR 6100e
Parallel Highway Interface System (CAMAC)	IEEE 596-1976*	552	**	EUR 4600e
Block Transfers in CAMAC Systems	IEEE 683-1976	†	**	EUR 4100 suppl
CAMAC Instrumentation and Interface Standards***	SH06437*** (Library of Congress No. 76-39660)	-	-	-
Amplitude Analogue Signals within a 50Ω System	-	-	TID-26614	EUR 5100e
The Definition of IML A Language for use in CAMAC Systems	-	-	TID-26615	ESONE/IML/01
Multiple Controllers in a CAMAC Crate	†	-	DOE/EV-0007	EUR 6500e
CAMAC Tutorial Articles	-	-	TID-26618	-
Real-Time BASIC for CAMAC	-	-	TID-26619	ESONE/RTB/02
Recommendations for CAMAC Serial Highway Drivers and LAM Graders for the SCC-L2	-	-	DOE/EV-0006	ESONE/SD/02
Supplement to CAMAC Standards and Reports	†	-	DOE/EV-0009	-

*Includes supplementary information
**Superseded by corresponding IEEE Standard listed
***This is a hard cover book that contains IEEE Stds 583-1975, 595-1976, 596-1976 and 683-1976 plus introductory material
†In preparation

AVAILABILITY OF DOCUMENTS

IEEE - IEEE Service Center, 445 Hoes Lane, Piscataway, New Jersey 08854, U.S.A.

IEC - International Electrotechnical Commission, 1, rue de Varembe, CH-1211 Geneva 20, Switzerland

DOE & TID Reports - National Bureau of Standards, Washington, D.C. 20234, U.S.A., Attn: L. Costrell

EURATOM - Office of Official Publications of the European Communities, P. O. Box 1003, Luxembourg

ESONE - Commission of the European Communities, CGR-BCMN, B-2440 GEEL, Belgium, Attn: ESONE Secretariat, H. Meyer

CONTENTS

1. Introduction

This document describes a set of standard subroutines to provide access to CAMAC facilities in a variety of computer-programming languages. It is specifically intended that the subroutines be suitable for use with FORTRAN, although they are not restricted to that language. Appendix B describes a recommended implementation of the subroutines explicitly for FORTRAN.

The present approach is based largely on IML and IML-M1 (Ref. 1). A distinction is made between "declarations", which are used to name and specify computer and CAMAC entities, and "actions", which are used to implement the various data movements and condition tests which make up the CAMAC-related portion of a program. As far as possible, the nomenclature of IML-M1 has been followed in order to take advantage of existing familiarity with that system and to provide as uniform a terminology and style as possible among the various CAMAC software documents.

Because of the widespread use of CAMAC on computers with a word length of less than 24 bits and because of the great differences in computer and operating-system features, special, system-dependent features are often required to provide greater efficiency or to make appropriate use of the features of particular systems. The main body of this document describes subroutines which, at the user interface, depend only on the features of CAMAC and therefore, when implemented in any standard procedural language, should be computer-independent. Appendix A describes subroutines which depend not only on CAMAC, but to some extent on individual computers. They cannot be made independent of the system on which they are implemented, and the user should take special precautions when it is necessary to incorporate them into a program. Such a program may not be transportable from one computer to another without modification.

The subroutines have been grouped into three subsets in order to provide different standard levels of implementation. The lowest level requires only two subroutines, but nevertheless gives access to most of the facilities which can be found in CAMAC systems. In higher levels of implementation subroutines are added which permit procedures to be written in more mnemonic terminology, provide better handling of LAM's, permit procedures to be independent of the type of CAMAC highway used, and provide efficient block-transfer capability.

2. Functional Specifications

This section introduces and describes in detail all the recommended subroutines. Since many implementations will not require the complete set, the subroutines are grouped into three subsets corresponding to the recommended implementation levels. Level A, the simplest, requires only the subroutines from section 2.1. Level B, an intermediate level, requires the subroutines from sections 2.1 and 2.2. Level C is the highest level and requires implementation of the subroutines from sections 2.1, 2.2, and 2.3.

Two CAMAC facilities are not available through the use of these subroutines: the X response from an action and the BZ command. These facilities are available through the use of system-dependent subroutines described in Appendix A.

Naming conventions, compatible with the requirements of many computer languages, have been adopted. In order to make it simple for a user to avoid name conflicts, the name of each recommended subroutine begins with the letter "C". The second letter of the subroutine name is coded to indicate the general function of the subroutine. Six letters have been used for this purpose:

"C" indicates that the subroutine performs a control function;

"D" indicates that the subroutine is a declaration of a CAMAC entity;

"F" indicates that the subroutine transfers full-length (24-bit) data words;

"G" indicates that the subroutine analyzes a named CAMAC entity into its address components;

"S" indicates that the subroutine transfers short (less than 24-bit) data words;

"T" indicates that the subroutine tests the state of a signal or status indication.

The remaining letters of each subroutine name (to a maximum of six) are chosen for their mnemonic value in identifying which function the subroutine performs.

Since no particular language is assumed in the body of the document, no syntax can be defined for a subroutine call. The subroutines are described in terms of a subroutine name and an ordered sequence of parameters. As far as possible every implementation should retain the designated name and the order of the parameters. For the same reason no specific form can be defined for the subroutine parameters. CAMAC data words are bit strings with a length of twenty-four. An implementation must have the capacity to represent such strings. Other subroutine parameters are represented either as integer values or as the logical values "true" or "false". In this document the parameters of subroutines are described as variables or arrays of types CAMAC word, integer, or logical. For each implementation appropriate storage units and data formats must be chosen for each of these generalized entities.

2.1 Primary Subroutines

These two subroutines, which are required in all implementations, make up level A. The first provides the capability to define the address of a CAMAC entity and to access it. The second is used to perform CAMAC operations on the defined entities. In principle any CAMAC entity for which

there is a defined standard mode of access can be accessed through the use of these two subroutines. In practice some systems may contain restrictions on the use of crate controllers or other system modules.

2.1.1 Declare CAMAC Register

Name: CDREG

Parameters:
- ext (external address, see section 3.1)
- b (branch number, see section 3.2)
- c (crate number, see section 3.3)
- n (station number, see section 3.4)
- a (subaddress, see section 3.5)

Function: CDREG combines the branch number b, the crate number c, the station number n, and the subaddress a into a convenient system-dependent form and stores the result in ext. Since the method of encoding depends on the implementation, the contents of ext should not be modified by the program. Some subroutines (see section 2.2) require only a crate address; if the parameters n and a are both zero, CDREG encodes a crate address and stores it in ext.

2.1.2 Perform Single CAMAC Action

Name: CFSA

Parameters:
- f (function code, see section 3.6)
- ext (external address, see section 3.1)
- int (CAMAC data word, see section 3.7)
- q (Q response, see section 3.8)

Function: CFSA causes the CAMAC action specified by the function code f to be performed at the CAMAC address specified by ext. If f contains a read or write code, a twenty-four-bit data transfer occurs between the CAMAC register addressed by ext and the computer storage location int. Otherwise int is ignored. The state of Q resulting from the operation is stored in q, "true" if Q=1, "false" if Q=0.

2.2 Single-Action Subroutines

These subroutines, together with those described in section 2.1, form the level B implementation, which provides a complete facility for specifying single CAMAC actions in a way which is mnemonic, compact, and independent of the type of highway or crate controller. Facilities are provided for declaring LAM's and performing LAM actions using constructions which are independent of the LAM access mode (i.e., subaddress or register access)

2.2.1 Generate Dataway Initialize

Name: CCCZ

Parameter: ext (external address, see section 3.1)

Function: CCCZ causes Dataway Initialize (Z) to be generated in the crate specified by ext.

2.2.2 Generate Crate Clear

Name: CCCC

Parameter: ext (external address, see section 3.1)

Function: CCCC causes Dataway Clear (C) to be generated in the crate specified by ext.

2.2.3 Set or Clear Dataway Inhibit

Name: CCCI

Parameters: ext (external address, see section 3.1)
l (logical truth value, see section 3.9)

Function: CCCI causes Dataway Inhibit (I) to be set in the crate specified by ext if the value of l is "true" and to be reset if the value of l is "false".

2.2.4 Test Dataway Inhibit

Name: CTCI

Parameters: ext (external address, see section 3.1)
l (logical truth value, see section 3.9)

Function: CTCI sets the value of l to "true" if Dataway Inhibit is set in the crate specified by ext and sets the value of l to "false" if Dataway Inhibit is not set.

2.2.5 Enable or Disable Crate Demand

Name: CCCD

Parameters: ext (external address, see section 3.1)
l (logical truth value, see section 3.9)

Function: CCCD causes Crate Demand to be enabled in the crate specified by ext if the value of l is "true" and causes Crate Demand to be disabled if the value of l is "false".

2.2.6 Test Crate Demand Enabled

Name: CTCD

Parameters: ext (external address, see section 3.1)
l (logical truth value, see section 3.9)

Function: CTCD sets the value of l to "true" if Crate Demand is enabled in the crate specified by ext and sets the value of l to "false" if Crate Demand is disabled.

2.2.7 Test Crate Demand Present

Name: CTGL

Parameters: ext (external address, see section 3.1)
l (logical truth value, see section 3.9)

Function: CTGL sets the value of l to "true" if any demand is present in the crate specified by ext and sets the value of l to "false" if no demand is present.

2.2.8 Declare LAM

Name: CDLAM

Parameters: lam (LAM identifier, see section 3.10)
b (branch number, see section 3.2)
c (crate number, see section 3.3)
n (station number, see section 3.4)
m (LAM access specifier, see section 3.11)
inta (integer array, see section 3.12)

Function: CDLAM encodes the branch number b, the crate number c, the station number n and all other necessary information concerning a LAM into a system-dependent, integer form and stores the result in lam. The parameter m is interpreted as a subaddress if its value is greater than or equal to zero. It is interpreted as the negative of a bit position if its value is less than zero. This information is encoded in the value assigned to lam and used by other subroutines to determine whether the LAM is accessed via special functions for LAM access or via reading and writing group 2 registers. The information contained in the array inta is completely implementation-dependent. It contains any information which may be required by the computer system or the CAMAC interface to enable the program to access the LAM. The use of inta by CDLAM is not required, but inta must appear in the parameter string regardless.

2.2.9 Enable or Disable LAM

Name: CCLM

Parameters: lam (LAM identifier, see section 3.10)
l (logical truth value, see section 3.9)

Function: CCLM causes the LAM specified by lam to be enabled if the value of l is "true" and causes it to be disabled if the value of l is "false".

2.2.10 Clear LAM

Name: CCLC

Parameter: lam (LAM identifier, see section 3.10)

Function: CCLC causes the LAM specified by lam to be cleared.

2.2.11 Test LAM

Name: CTLM

Parameters: lam (LAM identifier, see section 3.10)
l (logical truth value, see section 3.9)

Function: CTLM sets l to the value "true" if the LAM specified by lam is asserted, and sets l to the value "false" if the LAM is not asserted.

2.2.12 Link LAM to Service Procedure

Name: CCLNK

Parameters: lam (LAM identifier, see section 3.10)
label (entry point identifier, see section 3.13)

Function: CCLNK performs an association between the LAM specified by lam and a procedure identified by the parameter label. As a result of this association the procedure will be executed whenever the LAM is recognized by the system.

2.3 Block Transfers, Multiple Actions, and Inverse Declarations

These subroutines, together with those described in sections 2.1 and 2.2, form a comprehensive CAMAC support system, level C, with very general features and a potential for efficient execution of block transfers and multiple actions. (See Ref. 2)

All the action subroutines in this section employ a "control block", cb, which is an integer array containing four elements as follows:

element 1: Repeat count,
element 2: Tally,
element 3: LAM identification,
element 4: Channel identification.

The repeat count specifies the number of CAMAC actions or the maximum number of data words to be transferred. The tally is returned by the subroutine and indicates the number of actions actually executed or the number of data words actually transferred. Thus the calling program can detect and analyze a premature termination of an activity. The LAM identification is a coded identifier of the same form and interpretation as that returned by the subroutine CDLAM. The channel identification is a system-dependent or implementation-dependent parameter which may not be required in all implementations. It is used to identify any computer-related facilities necessary for the execution of the specified CAMAC action. Examples of facilities which might be required to be identified include computer input/output channels and operating system facilities such as unit numbers.

2.3.1 General Multiple Action

Name: CFGA

Parameters:
- fa (function codes, see section 3.14)
- exta (external addresses, see section 3.15)
- intc (CAMAC data array, see section 3.16)
- qa (Q responses, see section 3.17)
- cb (control block, see section 3.18)

Function: CFGA causes a sequence of CAMAC functions specified in successive elements of fa to be performed at a corresponding sequence of CAMAC addresses specified in successive elements of exta. Any read or write function in fa causes a CAMAC data word to be transferred between the corresponding element of intc and the specified CAMAC register. The Q response for each individual CAMAC action is stored in the corresponding element of qa. The number of actions performed and the number of elements required in the arrays fa, exta, intc, and qa is given by the value contained in the first element of cb. If the third element of cb contains the value zero, the specified sequence of actions is executed immediately; if it contains a LAM identification, then the sequence of actions is not initiated until the LAM is recognized.

2.3.2 Address Scan

Name: CFMAD

Parameters:
- f (function code, see section 3.6)
- extb (external addresses, see section 3.19)
- intc (CAMAC data array, see section 3.16)
- cb (control block, see section 3.18)

Function: CFMAD causes a single CAMAC function specified by the value of f to be executed at a succession of addresses computed using the Address Scan algorithm described in Ref. 3, section 5.4.3.1 and Ref. 2, section 3.2. In the Address Scan mode the specified function is executed first at the address given by the first element of extb. Then if the Q response is 1, the subaddress is incremented by 1 and the index into the data array intc incremented by 1, and the function is executed at this new subaddress. If the subaddress is incremented beyond 15, it is set to zero, and the station number is incremented.

If the Q response is 0, the subaddress is set to zero and the station number is incremented, but the index into intc is not changed. Execution of the specified function is attempted at the resulting new address, and the process is repeated until either the requested number of actions (given in the first element of cb) have been performed or the address computed by the procedure described above exceeds the address contained in the second element of extb. If the third element of cb contains the value 0, execution of the CAMAC actions is begun immediately. If it contains a LAM identifier, execution does not begin until the specified LAM is recognized. Ref. 3 defines Address Scan within a single crate only. For the purpose of this document Address Scan is extended to other address components as follows. If the station number is incremented beyond the number of stations in a crate, it is set to 1, and the crate number is incremented. If the crate number is incremented beyond the number of crates in the branch, the crate number is set to 1, and the branch number is incremented.

2.3.3 Controller-Synchronized Block Transfer

Name: CFUBC

Parameters:
- f (function code, see section 3.6)
- ext (external address, see section 3.1)
- intc (CAMAC data array, see section 3.16)
- cb (control block, see section 3.18)

FUNCTION: CFUBC causes the single CAMAC function given by the value of f to be executed at the CAMAC address specified by the value of ext. In this mode the CAMAC address is never changed, but the single register is expected to supply or accept many words of data. It is assumed able to supply or accept a data word whenever the controller addresses it until the block is exhausted or the controller terminates the process because the number of data transfers exceeds the limit given by the first element of cb. If the third element of cb contains the value 0, execution of the CAMAC actions is begun immediately. If it contains a LAM identifier, execution does not begin until the specified LAM is recognized.

The module indicates that the block is exhausted by its Q response. Two methods for doing this are described in Ref. 2-- i.e. Stop Mode (section 3.1) and Stop-on-Word Mode (section 4.1). In general both methods are not implemented in a given hardware system and no confusion should occur. However, if both methods are permissible, then the fourth element of cb which contains system dependent facilities must be used to specify for each case which method the module is using.

2.3.4 LAM-Synchronized Block Transfer

Name: CFUBL

Parameters:
- f (function code, see section 3.6)
- ext (external address, see section 3.1)
- intc (CAMAC data array, see section 3.16)
- cb (control block, see section 3.18)

Function: CFUBL causes the single CAMAC function specified by the contents of f to be executed at the CAMAC address specified by the contents of ext, with the usage of the Q response and the LAM signal defined as in Ref. 2, section 4.2 for the ULS mode. In the ULS mode the CAMAC address is not changed, but the single address is expected to supply or accept many words of data. In the ULS mode the module asserts a LAM whenever it is ready to participate in a data transfer, and the controller responds with the appropriate command to effect the transfer. Any data words transferred are placed into or taken from the array intc. The controller terminates the process if the number of executions exceeds the limit given by the contents of the first element of cb, or the module may terminate the process by responding with Q=0. The response Q=1 indicates that the specified function was properly executed. The response Q=0 indicates that the attempted data transfer was not completed and that the process should be terminated. The LAM which synchronizes the process is specified by the contents of the third element of cb. Upon termination the number of Q=1 responses is stored in the second element of cb.

2.3.5 Repeat Mode Block Transfer

Name: CFUBR

Parameters:
f	(function code, see section 3.6)
ext	(external address, see section 3.1)
inta	(CAMAC data array, see section 3.16)
cb	(control block, see section 3.18)

Function: CFUBR causes the single CAMAC function specified by the contents of f to be executed at the CAMAC address specified by the contents of ext with the usage of the Q response defined as in Ref. 3, section 5.4.3.2 and Ref. 2, section 3.3 for the Repeat mode. In the Repeat mode the CAMAC address is never changed, but the single address is expected to supply or accept many words of data. Q is used as a timing signal. Q=1 indicates that the previously executed function succeeded; Q=0 indicates that the module was not ready to execute the function and that the controller should try again. Any data words transferred are placed into or taken from the array intc. If the response is Q=0, no transfer took place, and the index into the array intc is not changed. The number of Q=1 responses expected is given by the contents of the first element of cb. If the third element of cb contains zero, the process is initiated immediately; if it contains a LAM identifier, the process is initiated only when the specified LAM is recognized.

2.3.6 Analyze LAM Identifier

Name: CGLAM

Parameters:
lam	(LAM identifier, see section 3.10)
b	(branch number, see section 3.2)

c	(crate number, see section 3.3)
n	(station number, see section 3.4)
m	(LAM access identifier, see section 3.11)
inta	(integer array, see section 3.12)

Function: CGLAM decodes the LAM identifier lam into its component parts, consisting of the branch number b, the crate number c, the station number n, the subaddress or bit-position m, and the system-dependent information inta. This subroutine exactly reverses the process performed by CDLAM, and all parameters have the same interpretation and form.

2.3.7 Analyze Register Identifier

Name: CGREG

Parameters:	ext	(external address, see section 3.1)
	b	(branch number, see section 3.2)
	c	(crate number, see section 3.3)
	n	(station number, see section 3.4)
	a	(subaddress, see section 3.5)

Function: CGREG decodes the CAMAC address identifier ext into its component parts, consisting of the branch number b, the crate number c, the station number n, and the subaddress a. This subroutine exactly reverses the process performed by CDREG, and all parameters have the same interpretation and form.

3. Definitions of Parameters

The meanings of the parameters used in the subroutine definitions are given briefly in the subroutine function descriptions. Since similar parameters are used in many subroutines, the various symbols are defined and discussed more completely in the subsections which follow. This information is intended particularly as a guide to implementors in the hope that it will result in the greatest possible degree of uniformity among different implementations.

3.1 ext (external address)

The symbol ext represents an integer which is used as an identifier of an external CAMAC address. The address may represent a register which can be read or written, a complete CAMAC address which can be accessed by control or test functions, or a crate address. The value of ext is explicitly defined to be an integer. Normally it can be expected to be an encoded version of the address components, in which the coding has been selected for the most efficient execution of CAMAC actions on the interface to which the implementation applies. Other possibilities are allowed, however. For example ext may be an index or a pointer into a data structure in which the actual CAMAC address components are stored.

3.2 b (branch number)

The symbol b represents an integer which is the branch number component of a CAMAC address. It may represent a physical highway number in multiple highway systems, or it may represent sets of crates grouped together for functional or other reasons. In some systems it may be ignored, although it must be included in the parameter list for the sake of compatibility.

3.3 c (crate number)

The symbol c represents an integer which is the crate number component of a CAMAC address. "Crate number" in this context can be either the physical crate number or it can be an integer symbol which is interpreted by the computer system software to produce appropriate hardware access information.

3.4 n (station number)

The symbol n represents an integer which is the station number component of a CAMAC address.

3.5 a (subaddress)

The symbol a represents an integer which is the subaddress component of a CAMAC address.

3.6 f (function code)

The symbol f represents an integer which is the function code for a CAMAC action.

3.7 int (CAMAC data word)

The symbol int represents a CAMAC data word stored in computer memory. The form is not specified, but the word must be stored in an addressable storage entity capable of containing twenty-four bits. In a computer or programming system which does not have an addressable unit of storage which can contain twenty-four bits, multiple units must be used.

3.8 q (Q response)

The symbol q represents a logical truth value which corresponds to the CAMAC Q response. It is set to "true" if the Q response is 1, to "false" if the Q response is 0.

3.9 l (logical truth value)

The symbol l represents a logical truth value which can be either "true" or "false".

3.10 lam (LAM identifier)

The symbol lam represents an integer which is used as the identifier of a CAMAC LAM signal. The information associated with the identifier must include not only the CAMAC address but also information about the means of accessing and controlling the LAM, i.e., whether it is accessed via dataless functions at a subaddress or via read/write functions in group 2 registers.

The value of lam is explicitly defined to be a non-zero integer; whether it is an encoded representation of the information required to describe the LAM or simply provides a key for accessing the information in a system data structure is an implementation decision. The value 0 is used to indicate, where appropriate, that no LAM is being specified.

3.11 m (LAM access specifier)

The symbol m represents an integer which is used to indicate the mode of access of a LAM and the lowest-order address component for LAM addressing. If m is zero or positive, it is interpreted as a subaddress and the LAM is assumed to be accessed via dataless functions at this subaddress. If m is negative, it is interpreted as the negative of a bit position for a LAM which is accessed via reading, setting, or clearing bits in the group 2 registers at subadresses 12, 13, or 14.

3.12 inta (integer array)

The symbol inta represents an integer array, the length and contents of which are not defined in this document. It is intended to contain system-dependent or implementation-dependent information associated with the definition of a LAM. If no such information is required, the array need not be used. This information can include parameters necessary for interrupt linkage, event specification, etc. The documentation for an implementation must describe the requirements for any parameters contained in this array.

3.13 label (entry point identifier)

The symbol label represents an entry point into a programmed procedure. Such a procedure will typically be executed in response to the recognition of a LAM, and it may interrupt the process being executed at the time of recognition of the LAM. Under these circumstances the procedure must be capable of saving and restoring the state of the computer so that the interrupted process can be resumed. At least one value of labels should identify a system error procedure which deals with LAM's not linked to user processes.

3.14 fa (function codes)

The symbol fa represents an array of integers, each of which is the function code for a CAMAC action. The length of fa is given by the value of the first element of cb (see section 3.18) at the time the subroutine is executed.

3.15 exta (CAMAC external address)

The symbol exta represents an array of integers, each of which is a CAMAC register address. The form and information content of each element of exta must be identical to the form and information content of the quantity ext (see section 3.1). The length of exta is given by the value of the first element of cb (see section 3.18) at the time the subroutine is executed.

3.16 intc (CAMAC data array)

The symbol intc represents an array of CAMAC data words. Each element of intc has the same form as the CAMAC data word variable int (see section

3.7). The length of intc is given by the value of the first element of cb (see section 3.18) at the time the subroutine is executed.

3.17 qa (Q responses)

The symbol qa represents an array of Q response values. Each element of qa has the same form and can have the same values as the parameter q (see section 3.8). The length of qa is given by the value of the first element of cb (see section 3.18) at the time the subroutine is executed.

3.18 cb (control block)

The symbol cb represents an integer array having four elements. The contents of these elements are:

element 1: Repeat Count,
element 2: Tally,
element 3: LAM identification,
element 4: Channel identification.

The repeat count specifies the number of individual CAMAC actions or the maximum number of data words to be transferred. Some multiple action and block transfer subroutines permit termination of the sequence upon a signal from the addressed module. In such cases the repeat count represents an upper limit. The tally is the number of actions actually performed or the number of CAMAC data words actually transferred. If the block transfer or multiple action is terminated by the controller due to exhaustion of the repeat count, the tally will be equal to the repeat count; otherwise it may be less. The LAM identification is an integer value having the same form and information content as the variable lam (see section 3.10). The channel identification is an integer value which identifies system-dependent facilities which may be necessary to perform the block transfer or multiple action. This number, if it is required, has the same form and content as the parameter chan (see Appendix A, section A6.2) and can be created by the subroutine CDCHN (see Appendix A, section A3.1).

3.19 extb (external addresses)

The symbol extb represents an array of integers containing external CAMAC addresses. The array has two elements; the first contains the starting address for an Address Scan multiple action. The second contains the final address which can be permitted to participate in the Address Scan sequence. Each element has the same form and information content as the parameter ext (see section 3.1).

4. References

1. CAMAC, The Definition of IML, TID-22615.
2. Block Transfers in CAMAC Systems, IEEE Std 683-1976.
3. IEEE Standard Modular Instrumentation and Digital Interface System (CAMAC), IEEE Std 583-1975.

4. IEEE Standard Parallel Highway Interface System (CAMAC), IEEE Std 596-1976.
5. IEEE Standard Serial Highway Interface System (CAMAC), IEEE Std 595-1976.

APPENDIX A

SYSTEM-DEPENDENT SUBROUTINES

A1. Introduction

It is often useful, even necessary, to utilize subroutines which depend strongly on an individual computer or operating system architecture. These subroutines cannot be made completely transportable even in an implementation for a highly standardized language. The functions they must perform are usually very similar, however, and may be identical within restricted classes of systems. This appendix describes a number of such system-dependent subroutines. They are not required to be in any level of implementation, but any of them may be supplied in an implementation at the discretion of the implementor. The documentation for a given implementation must indicate which ones are supplied and must describe completely their system-dependent features and parameters.

A2. Access to Special Signals

These subroutines give access to features of CAMAC systems for which a standard access method is not defined or which are not available in all systems.

A2.1 Branch Initialize

Name: CCINIT

Parameter: b (branch number, see section 3.2)

Function: CCINIT causes the branch initialize signal (BZ) to be asserted on the branch identified by the parameter b. If b identifies a group of crates not interfaced via a parallel branch, the function may be simulated by causing Crate Initialize signals to be asserted in the crates. The user of a system should be warned that the response of many modules to the initialize signal is such as to place the system in undersirable states; thus the feature should be used with caution.

A2.2 Test Status of Preceding Action

Name: CTSTAT

Parameter: k (status code, see section A6.1)

Function: CTSTAT stores an integer status code into the parameter k. The status code stored reflects the results of the last action executed in the program; the code has a value given by the expression 4e+d, where e and d are non-negative integers. The meaning of d is given by the following table:

value	meaning
0	Q=1, X=1
1	Q=0, X=1
2	Q=1, X=0
3	Q=0, X=0

If e is zero, there are no errors reported; if e is greater than zero, then some error or exception is indicated. The exact meanings of e are system-dependent and must be defined in the documentation for each implementation.

A3. Channel Identifier

It may be necessary in some computers or in some operating systems to utilize special I/O channel controllers or operating system features such as logical unit numbers and device codes to access CAMAC systems. These two subroutines provide a means to specify or interrogate such information.

A3.1 Declare Channel

Name: CDCHN

Parameters: chan (channel identifier, see section A6.2)
additional parameters, depending on implementation

Function: CDCHN encodes system-dependent parameters into a convenient form and returns the results as the value of chan. It specifies the computer hardware and/or software I/O facilities to be used and possibly additional information relating to their use. The additional parameters must be completely described in the documentation for each implementation.

A3.2 Analyze Channel Declaration

Name: CGCHN

Parameters: chan (channel identifier, see section A6.2)
additional parameters, depending on implementation

Function: CGCHN extracts the system-dependent parameters from the channel identifier chan. It reverses the process performed by CDCHN.

A4. Short Data-Word Transfers

In many systems and applications it is inconvenient and inefficient to utilize the full twenty-four bit length of the CAMAC data word and sufficient to use a truncated data word. The appropriate degree of truncation is dependent on the system, but the computer word length is the most common choice. The leftmost bits of the CAMAC data word are discarded, while the rightmost bits are retained. The length of a truncated CAMAC data word must be specified in the documentation. These subroutines duplicate functions performed by subroutines described in section 2 of the main body of this

document, except that truncated CAMAC data words are used.

A4.1 Perform Single CAMAC Action

Name: CSSA

Parameters:
- f (function code, see section 3.6)
- ext (external address, see section 3.1)
- ints (truncated CAMAC data word, see section A6.3)
- q (Q response, see section 3.8)

Function: CSSA performs the same function as CFSA (see section 2.1.2) except that ints contains a truncated CAMAC data word.

A4.2 General Multiple Action

Name: CSGA

Parameters:
- fa (function codes, see section 3.14)
- exta (external addresses, see section 3.15)
- intt (truncated CAMAC data array, see section A6.4)
- qa (Q responses, see section 3.17)
- cb (control block, see section 3.18)

Function: CSGA performs the same function as CFGA (see section 2.3.1) except that the elements of intt are truncated CAMAC data words.

A4.3 Address Scan

Name: CSMAD

Parameters:
- f (function code, see section 3.6)
- extb (external addresses, see section 3.19)
- intt (truncated CAMAC data array, see section A6.4)
- cb (control block, see section 3.18)

Function: CSMAD performs the same function as CFMAD (see section 2.3.2) except that the elements of intt are truncated CAMAC data words.

A4.4 Controller-Synchronized Block Transfer

Name: CSUBC

Parameters:
- f (function code, see section 3.6)
- ext (external addresses, see section 3.1)
- intt (truncated CAMAC data array, see section A6.4)
- cb (control block, see section 3.18)

Function: CSUBC performs the same function as CFUBC (see section 2.3.3) except that the elements of intt are truncated CAMAC data words.

A4.5 LAM-Synchronized Block Transfer

Name: CSUBL

Parameters:
- f (function code, see section 3.6)
- ext (external address, see section 3.1)
- intt (truncated CAMAC data array, see section A6.4)
- cb (control block, see section 3.18)

Function: CSUBL performs the same function as CFUBL (see section 2.3.4) except that the elements of intt are truncated CAMAC data words.

A4.6 Repeat Mode Block Transfer

Name: CSUBR

Parameters:
- f (function code, see section 3.6)
- ext (external address, see section 3.1)
- intt (truncated CAMAC data array, see section A6.4)
- cb (control block, see section 3.18)

Function: CSUBR performs the same function as CFUBR (see section 2.3.5) except that the elements of intt are truncated CAMAC data words.

A5. Define Crate Identifier

Name: CDCRT

Parameters:
- c (crate number, see section 3.3)
- intb (integer array, see section A6.5)

Function: CDCRT defines the crate number c in terms of the system-dependent information contained in the array intb. It permits the meaning of the crate number c to be changed within a procedure execution.

A6. Definitions of Parameters

Although many of the parameters used in the subroutines of Appendix A are identical with those described in section 3 of the main body of this document, a few new ones have been introduced. The parameters referred to only in this appendix are described below.

A6.1 k (status code)

The symbol k represents an integer used to provide status information with respect to a CAMAC action. The rightmost two bits contain the complements of the X and Q responses; the remaining bits in a word contain system-dependent error codes. The lowest-order bit contains the complement of the Q response; the next bit to the left (the 2's bit) contains the complement of X.

A6.2 chan (channel identifier)

The symbol chan represents an integer which contains a channel identifier.

A6.3 ints (truncated CAMAC data word)

The symbol ints represents a storage cell in computer memory used to hold bits 1 to n of a CAMAC data word, where n is defined for the implementation to be less than twenty-four. Higher bits in the CAMAC word are not used when the function is a READ; they are set to zeros if the function is a WRITE.

A6.4 intt (truncated CAMAC data array)

The symbol intt represents an array in computer memory each element of which contains a truncated CAMAC data word having the same form as ints (see section A6.3).

A6.5 intb (integer array)

The symbol intb represents an integer array, the length and contents of which are not defined in this document. It is intended to contain system dependent or implementation-dependent information associated with the definition of a crate number identifier. The information may include physical crate number, highway or computer interface identification, or operating system parameters necessary for access. The documentation for an implementation must describe the requirements for any parameters contained in this array.

APPENDIX B

FORTRAN IMPLEMENTATION

B1. General

This appendix recommends an implementation of the CAMAC standard subroutines for use with FORTRAN for the purpose of communicating with CAMAC-interfaced devices and synchronizing program execution with events associated with such devices.

B2. Description of Subroutines

B2.1 Primary Subroutines

B2.1.1 Declare CAMAC Register

Form. CALL CDREG (ext,b,c,n,a)

Function: CDREG combines the components b, c, n, and a into a single CAMAC register reference.

Implementation Notes:

ext: The subroutine must return a value which uniquely defines the specified CAMAC register or crate for other subroutines in the implementation. The means by which this objective is accomplished are not specified. The most common procedure is expected to be to return a value in which the parameters b, c, n, and a have been packed into a single integer with the fields defined for the most efficient execution of CAMAC actions on the interface to which the implementation applies. Other possibilities are allowed, however. For example the values of the parameters could be stored in a data structure and a pointer to the entry returned. If n and a are defined to be zero, then ext is defined to be a crate. If n and a are not zero, then ext is defined to be a register or other addressable entity.

Tables of Register References: It is advantageous in many applications to have tables of CAMAC register references, identical in form to the value returned by CDREG, available on a permanent basis to be loaded by programs or as arrays of constants within programs. Such tables not only save the time required to define the register references each time a program is executed but also can form parts of larger tables of logical device definitions. If such facilities can be supported by an implementation, the documentation should describe how to create these references in assembler code or in DATA statements in the compiler.

B2.1.2 Perform Single CAMAC Action

Form: CALL CFSA (f, ext, int, q)

Function: This subroutine performs a single CAMAC function at a single CAMAC address. If the function f is a read or a write function, a 24-bit data word is transferred.

Implementation Notes:

int: The internal reference int must be able to contain 24 bits. If the FORTRAN implementation supports integer variables of 24 bits or more, int can be an integer expression (for write) or an integer variable or array element (for read). Otherwise int must be an integer array (or a portion of one) with enough elements to contain 24 bits. When a CAMAC data word is stored in an integer variable, or when an integer constant is used as int, CAMAC bit 1 occupies the low-order bit position, and bit 24 occupies the twenty-fourth bit position to the left. (In a binary machine. In other radices an appropriate transformation to equivalent integer values must be made). If the FORTRAN implementation does not support integers with a length as great as 24 bits, then the bits of the CAMAC data word are stored in the array elements such that bit 1 is the lowest order bit in the highest numbered element. All the bits in the highest element are filled and the remaining bits are stored beginning in the lowest order bit of the next lowest numbered element, etc. For what is probably the most common example, 16-bit words, one could define an array of length 2, say CAMDAT. Then bits 1 to 16 are stored in CAMDAT(2) with bit 1 the lowest and bit 16 the highest. Bits 17 to 24 are stored in CAMDAT(1) with bit 17 in the lowest order position in the word. The upper eight bits of CAMDAT(1) are cleared to zero.

Restrictions on Parameter Values: In multiple-user systems where CAMAC facilities are shared among several independent users, it may be necessary to restrict access to crate controllers and certain modules by not allowing crate controller actions to be issued via CFSA and by forbidding access via CFSA to certain other modules used for system control. In simpler systems, however, such restrictions are unnecessary and prevent the implementation of test and diagnostic codes. Each implementor must judge his own situation. Such restrictions can, of course, be controlled by setting mode switches either during operation or at system generation. Where such restrictions appear to be necessary, it is recommended that the implementation give the system manager the option of invoking them or not.

B2.2 Single Action Subroutines

This set consists of A plus the following:

B2.2.1 Generate Dataway Initialize

Form: CALL CCCZ (ext)

Function: This subroutine generates the crate Initialize signal, Z, on the Dataway of the addressed crate.

B2.2.2 Generate Crate Clear

Form: CALL CCCC (ext)

Function: This subroutine generates the crate Clear signal on the Dataway of the addressed crate.

B2.2.3 Set or Clear Dataway Inhibit

Form: CALL CCCI (ext,l)

Function: This subroutine sets the crate Inhibit signal in the addressed crate if l is "true" and clears it if l is "false".

B2.2.4 Test Dataway Inhibit

Form: CALL CTCI (ext,l)

Function: This subroutine returns the value "true" in l if the inhibit signal in the addressed crate is set, "false" if not.

B2.2.5 Enable or Disable Crate Demands

Form: CALL CCCD (ext,l)

Function: This subroutine enables or disables demands from the specified crate.

B2.2.6 Test Crate Demand Enabled

Form: CALL CTCD (ext,l)

Function: This subroutine returns in l the value "true" if crate demands are enabled, "false" if not.

B2.2.7 Test Demand Present

Form: CALL CTGL (ext,l)

Function: This subroutine returns in l the value "true" if any GL bit is present, "false" if not.

B2.2.8 Declare LAM

Form: CALL CDLAM (lam,b,c,n,m,inta)

Function: CDLAM combines the components b, c, n, m, and inta into a single reference to the LAM.

Implementation Notes:

lam: The subroutine must return a value which uniquely defines the specified LAM for other subroutines in the implementation. The means by which this objective is accomplished are not specified. If the parameters necessary to specify the address of a LAM and any necessary auxiliary information can be packed into a single integer variable, then the value returned may contain this information. Otherwise it may be necessary to supply an arbitrary index, or other code, which identifies the LAM to the other subroutines which access it.

Permanent Assignment of Logical LAM's: In many systems LAM's are not dynamic and can be known to the operating system or the CAMAC I/O programs more or less permanently. In such cases the function CDLAM is not required. The LAM's can be defined via tables which are part of the CAMAC I/O software or which may be loaded by a particular program to identify the LAM's with which it deals. If a particular implementation supports definition by such tables, the documentation should describe how to create LAM references using assembler code or compiler DATA statements.

B2.2.9 Enable or Disable LAM

Form: CALL CCLM (lam,l)

Function: This subroutine enables or disables the module LAM identified by lam.

B2.2.10 Clear LAM

Form: CALL CCLC (lam)

Function: This subroutine clears the module LAM identified by the argument.

B2.2.11 Test LAM

Form: CALL CTLM (lam,l)

Function: This subroutine returns in l the value "true" if the addressed LAM is asserted, "false" if not.

B2.2.12 Link LAM to Service Procedure

Form: CALL CCLNK (lam,label)

Function: Perform a run time association between a logical LAM and an interrupt service routine.

B2.3 Block Transfers, Multiple Actions, and Inverse Declarations

This set consists of sets A and B plus the following:

B2.3.1 General Multiple Action

Form: CALL CFGA (fa,exta,intc,qa,cb)

Function: This subroutine performs a sequence of CAMAC functions at a corresponding sequence of CAMAC addresses and returns a sequence of Q responses. The number of functions executed, addresses accessed, and Q responses returned is given by the value of the parameter cb(1). CAMAC data words are transferred between the CAMAC system and the array intc whenever the specified function is read or write. Since the array fa can contain both read and write functions, data words may be written to and read from the same array by a single call to this subroutine. CAMAC-data-word positions in the array intc which correspond to functions in the array fa which are neither read nor write are not accessed by this subroutine.

Implementation Notes:

fa: Note that in this subroutine a succession of possibly unrelated functions are executed at a succession of CAMAC addresses.

intc: The internal reference intc, unless all functions in the array fa are control or test functions, which use no data, must be able to contain a CAMAC data word for each function executed. In order to maintain the positional correspondence between values in fa, exta, intc, and qa, a position in intc is skipped whenever a control or test function is executed. If the FORTRAN implementation supports integer array elements of 24 bits or more, then each execution of the function accesses a successive element of the array. The CAMAC data word occupies the low-order 24 bits of the integer array element; higher order bits are cleared by a read function and ignored by a write. If the FORTRAN implementation supports only integer arrays whose elements contain fewer than 24 bits, then each CAMAC data word is contained in an integral number of elements (a "block") sufficiently large to contain 24 bits. The length of the array must be great enough to include a block for each CAMAC data word transferred. Each CAMAC data word is positioned within its block so that the element within the block with the highest index is filled with the lowest order bits of the data word, and any bits not required to contain data are at the highest order positions of the element with the lowest index. Thus in the typical 16-bit case, where a block consists of elements n and n + 1, element n contains CAMAC bits 24 through 17 in its low-order eight bits and element n + 1 contains bits 16 through 1.

qa: This array contains the Q responses for each function executed. The format of each element of the array qa is identical to the format of the variable q in subroutine CFSA, i.e., a logical value is returned. No action based on these values is taken by the subroutine.

Restrictions on Parameter Values: In multiple-user systems where CAMAC facilities are shared among several independent users, it may be necessary to restrict access to crate controllers and certain modules by not allowing crate controller actions to be issued via CFGA and by forbidding access via CFGA to certain other modules used for system control. In simpler systems, however, such restrictions are unneccessary and prevent the implementation of test and diagnostic codes. Each implementor must judge his own situation. Such restrictions can, of course, be controlled by setting mode switches either during operation or at system generation. Where such restrictions appear to be necessary, it is recommended that the implementation give the system manager the option of invoking them or not.

Applicability: This subroutine is useful primarily in systems in which the overhead associated with initiating or specifying a CAMAC action is great compared with the time required to perform an individual CAMAC command. Systems which suffer from a high I/O overhead in the system software or where there is a remote intelligent processor with a relatively high communications overhead are among those in which this subroutine may be needed.

B2.3.2 Address-Scan

Form: CALL CFMAD (f,extb,intc,cb)

Function: This subroutine executes a single CAMAC operation, f, at a succession of addresses computed using the Address Scan algorithm.

Implementation Notes:

intc: See implementation note on intc in section B2.3.1.

Q: The Q responses to the CAMAC functions are not saved. The Q response is used to control the addressing algorithm and can provide no information to the calling program.

X: Special attention must be paid to the X response in the Address Scan mode. When a module responds with Q=0, it may at the same time give either X=1 or X=0; consequently in this mode of access the X response is ignored when the Q response is 0.

Restrictions on Parameter Values: In multiple-user systems where CAMAC facilities are shared among several independent users, it may be necessary to restrict access to crate controllers and certain modules by not allowing crate controller actions to be issued via CFMAD and by forbidding access via CFMAD to certain other modules used for system control.

B2.3.3 Controller-Synchronized Block Transfer

Form: CALL CFUBC (f,ext,intc,cb)

Function: This subroutine executes a single CAMAC function, f, at a single address, with the usage of the Q response defined as for the Stop Mode.

Implementation Notes:

intc: See implementation note on intc in section B2.3.1.

Q: The Q responses to the CAMAC functions are not saved. Since the Q response is used by the module to indicate end of block, it can carry no other information to the calling program.

Restrictions on Parameter Values: In multiple-user systems where CAMAC facilities are shared among several independent users, it may be necessary to restrict access to crate controllers and certain modules by not allowing crate controller actions to be issued via CFUBC and by forbidding access via CFUBC to certain other modules used for system control.

B2.3.4 LAM-Synchronized Block Transfer

Form: CALL CFUBL (f,ext,intc,cb)

Function: This subroutine executes a single CAMAC function, f, at a single address, with the usage of the Q response and LAM signal for the ULS mode.

Implementation Notes:

intc: See implementation note on intc in section B2.3.1.

A: The Q responses to the CAMAC functions are not saved. Since the Q response is used by the module to indicate end of block, it can carry no other information to the calling program.

LAM Control: Modules designed to be used in this mode must clear the synchronizing LAM during the execution of a read or write command.

Restrictions on Parameter Values: In multiple-user systems where CAMAC facilities are shared among several independent users, it may be necessary to restrict access to crate controllers and certain modules by not allowing crate controller actions to be issued via CFUBL and by forbidding access via CFUBL to certain other modules used for system control.

B2.3.5 Repeat-Mode Block Transfer

Form: CALL CFUBR (f,ext,intc,cb)

Function: This subroutine executes a single CAMAC function, f, in the Repeat mode.

Implementation Notes:

intc: See implementation note on intc in section B2.3.1.

Q: The Q response to the CAMAC functions are not saved. Since the Q response is used by the module to indicate data synchronization, it can carry no other information to the calling program.

System Integrity: A malfunction or erroneous operation may produce a situation in which a module will never respond with Q=1, causing an unending loop. To guard against this case a time out or maximum repetition count should be implemented in the controller hardware or software.

Restrictions on Parameter Values: In multiple-user systems where CAMAC facilities are shared among several independent users, it may be necessary to restrict access to crate controllers and certain modules by not allowing crate controller actions to be issued via CFUBR and by forbidding access via CFUBR to certain other modules used for system control.

B2.3.6 Analyze LAM Identifier

Form: CALL CGLAM (lam,b,c,n,m,inta)

Function: CGLAM accepts a LAM identifier as input and returns the values of the hardware or operating system parameters which define it. It performs the inverse transformation of CDLAM.

B2.3.7 Analyze Register Identifier

Form: CALL CGREG (ext,b,c,n,a)

Function: CGREG accepts a register identifier, ext, as input, analyzes it into its components, and stores them into b, c, n, and a.

B3. Parameter Types

The types of the subroutine parameters are given in this section. See section 3 for the meaning of each parameter.

B3.1 Single Integers

If a subroutine returns a value in a parameter of this type, it must be either an integer variable of an integer array element. If no value is returned, it may be any integer expression. The parameters of this class are:

a, b, c, ext, f, lam, m, n.

B3.2 Single Logical Values

If a subroutine returns a value in a parameter of this type, it must be either a logical variable or a logical array element. If no value is returned, it may be any logical expression. The parameters of this class are:

l, q.

B3.3 Integer Arrays

These parameters must be integer arrays. The parameters of this class are:

cb, exta, extb, fa, inta.

B3.4 Logical Array

This parameter must be a logical array. The only parameter of this class is:

qa.

B3.5 CAMAC Data Word

This parameter must be capable of storing a CAMAC data word twenty-four bits in length. If the FORTRAN integer variable can contain twenty-four bits or more, then the type for CAMAC data word is the same as for a single integer (see section B3.1). If the integer variable length is greater than the equivalent of twenty-four bits, the CAMAC data word is represented as an unsigned integer with bit 24 representing the highest order bit and bit 1 representing the lowest. If the FORTRAN integer variable cannot contain twenty-four bits, then the parameter must be an integer array sufficiently long to contain twenty-four bits. The CAMAC data word must be represented in the array as a multiple-precision integer with the lowest-order portion in the last array element and the highest-order portion in the first element. Bit 24 of the CAMAC word is taken to be the highest-order bit and bit 1 the lowest. The only parameter of this class is:

int

B3.6 CAMAC Data Array

This parameter must be an array of elements each of the same type as the CAMAC data word (see section B3.5). The only parameter of this class is:

intc

B3.7 Label

This parameter cannot be defined or assigned a type within the bounds of strictly standard FORTRAN, since it labels a procedure which is intended to be executed out of sequence with respect to the calling program. It must be defined appropriately for each implementation. The only parameter of this class is:

label.

APPENDIX C

FUNCTION-CODE MNEMONIC SYMBOLS

Symbol	Value	Definition
RD1	0	Read Group 1 Register
RD2	1	Read Group 2 Register
RC1	2	Read and Clear Group 1 Register
RCM	3	Read Complement of Group 1 Register
TLM	8	Test LAM
CL1	9	Clear Group 1 Register
CLM	10	Clear LAM
CL2	11	Clear Group 2 Register
WT1	16	Write Group 1 Register
WT2	17	Write Group 2 Register
SS1	18	Selective Set Group 1 Register
SS2	19	Selective Set Group 2 Register
SC1	21	Selective Clear Group 1 Register
SC2	23	Selective Clear Group 2 Register
DIS	24	Disable
XEQ	25	Execute
ENB	26	Enable
TST	27	Test

☆ U.S. GOVERNMENT PRINTING OFFICE: 1978— 261-324/74