Conf-811247--1

GA-A16590

A REAL-TIME CONTROL AND DATA-ACQUISITION SYSTEM FOR HIGH-ENERGY NEUTRAL-BEAM INJECTORS

by
A. S. GLAD and V. JACOBSON

MASTER

DECEMBER 1981

GENERAL ATOMIC COMPANY

DISCLAIMER

DISCLAIMER

GA-A16590

A REAL-TIME CONTROL AND DATA-ACQUISITION SYSTEM FOR HIGH-ENERGY NEUTRAL-BEAM INJECTORS

by
A. S. GLAD and V. JACOBSON

This is a preprint of a paper to be presented at the Seventh (1981) North American Annual MUSE Meeting, December 7-11, 1981, Ft. Lauderdale, Florida

Work supported by
Department of Energy
Contract DE-AT03-76ET51011

GENERAL ATOMIC PROJECT 3344
DECEMBER 1981

MSW

GENERAL ATOMIC COMPANY

A REAL-TIME CONTROL AND DATA ACQUISITION SYSTEM FOR HIGH ENERGY NEUTRAL BEAM INJECTORS*

Anthony S. Glad
General Atomic Company
P.O. Box 81608
San Diego, CA 92138
and
Van Jacobson
Lawrence Berkeley Laboratory
University of California
Berkeley, CA 94720

ABSTRACT

The need for a real-time control system and a data acquisition, processing and archiving system operating in parallel on the same computer became a requirement on General Atomic's Doublet III fusion energy project with the addition of high energy neutral beam injectors.

The data acquisition processing and archiving system is driven from external events and is sequenced through each experimental shot utilizing ModComp's intertask message service. This system processes, archives and displays on operator console CRTs all physics diagnostic data related to the neutral beam injectores such as temperature, beam alignment, etc.

The real-time control system is data base driven and provides periodic monitoring and control of the numerous dynamic subsystems of the neutral beam injectors such as power supplies, timing, water cooling, etc.

*Work supported by U.S. Department of Energy Contract #DE-AT03-76ET51011

The hardware requirements defined the need for both systems to operate on the same computer system. Two ModComp Classic computers were provided with each computer controlling two neutral beam injectors. Each neutral beam injector generates approximately 160K words of physics diagnostic data and has approximately 600-700 signals for real-time control and monitoring.

The purpose of this paper is threefold.

First, to provide a brief description of each system including the methods for implementing and making them operational under MAX IV.

Second, to describe the method for using ModComp's intertask message service to sequence the data acquisition system.

Third, to describe the method used to coordinate the control and data acquisition function under MAX IV.

1. INTRODUCTION

The Doublet III fusion experiment is a large Tokamak of non-circular cross-section constructed and operated by the General Atomic Company for the U.S. Department of Energy. It has been in operation since early 1978, and is designed to study the effects of plasma shape upon confinement properties of temperature and density.

The Doublet III machine operates by performing experimental shots at five minute intervals. The five minute intervals are divided into a two minute power supply and hardware setup, a 5 to 10 second actual shot and a two to three minute data acquisition analysis and archiving period.

To provide increases in the fusion plasma temperature, neutral beam injectors were built and interfaced to the Doublet III. These neutral beams inject high energy neutral particles into the plasma. The neutral beams require a high degree of computer control with additions of data acquisition and reduction of physics diagnostic data.

The neutral beam system is based on a ModComp Classic computer system, for which two separate software subsystems were constructed. The first, the data acquisition and processing system which was built by the University of California Lawrence Berkeley Laboratories (LBL) (Ref. 1), performs the physics data acquisition and analysis. The second subsystem, the periodic scan and control system, was initially the control system for Doublet III (Ref. 2), and was adapted to work in parallel with the LBL subsystem. It provides periodic monitoring and control of all external hardware supporting the neutral beams, such as vacuum, power supplies, cooling water, etc.

2. HARDWARE CONFIGURATION

The hardware configuration for the neutral beam computer system is illustrated in Fig. 1. This figure illustrates one neutral beam computer system controlling two neutral beam lines. In actuality, there are two identical computer systems, as illustrated, providing control for four beam lines. The system operates on a Modcomp Classic computer with 512K words of main memory. An 80 megabyte Ampex disk containing all programs, data base files, data and user files is the only mass storage device on the system. An 800 bpi tape drive is used for both locally archiving tests data obtained from the neutral beams and for backing-up the 800 megabyte disk.

Two Grinnel display interfaces are connected to the system. Each Grinnel provides control for three graphic display monitors (2 black and white and one color). Each set of three monitors are implemented in a console (Fig. 2) used to monitor and control one beam line. A communications interface provides programming terminal interface including one graphic terminal (Tektronix 4014) plus five Ann Arbor alphanumeric terminals used for program development and viewing the current configuration of the data base files. Also connected through the communications interface is a microprocessor controlling both touch panel and knob inputs. Touch panel is used as the input medium for displays on one of the Grinnel monitors. The knobs are used to change timing and power supply values and cursor position when displayed on a screen. The microprocessor converts the knob and touch panel inputs to an ASCII stream for input through an asyncronous port. The Versatec printer plotter is used to copy the Grinnel monitor and the Tektronix 4014 display in addition to providing normal program listing output. The console teletypewriter is the focal point for all alarm and error messages generated from the system in

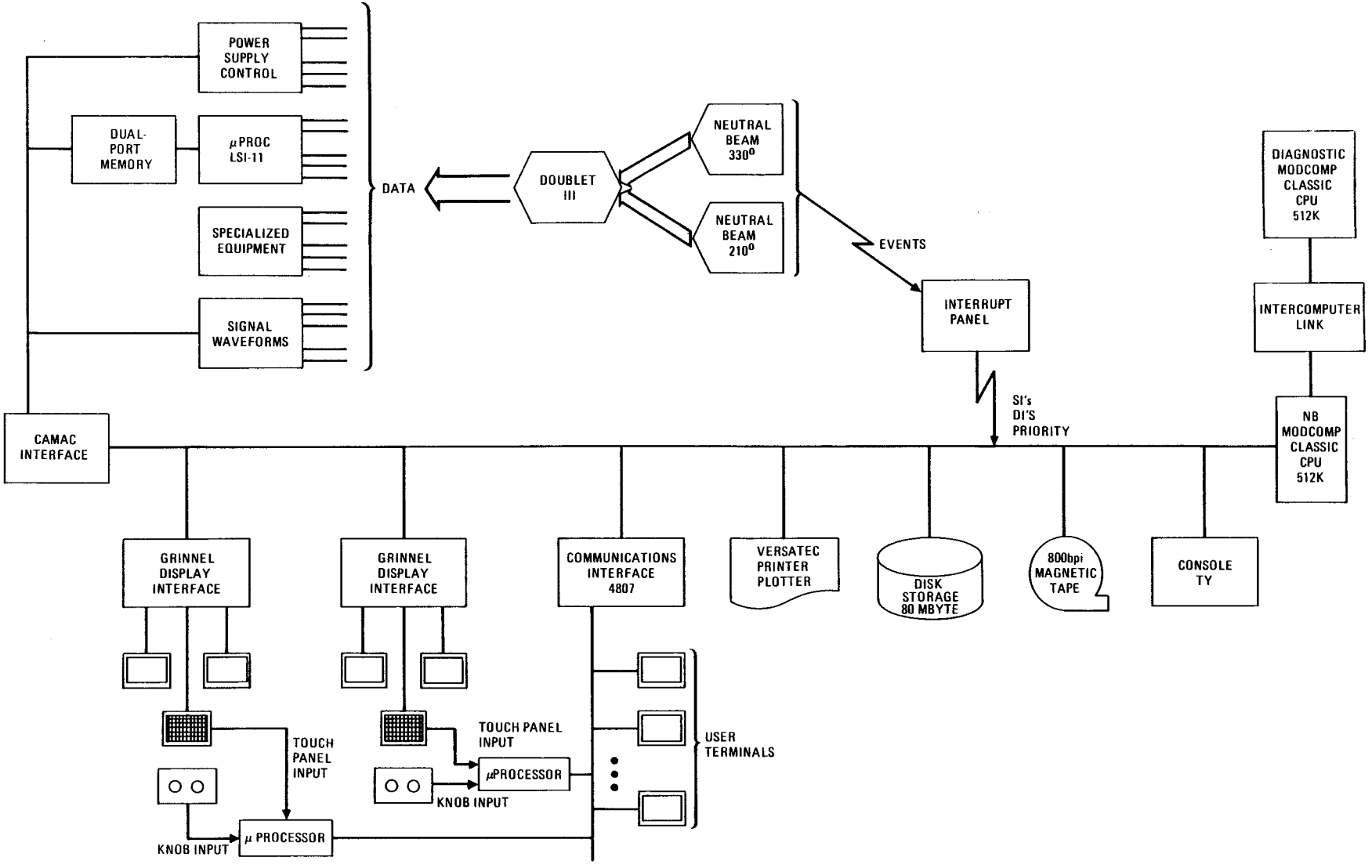

Fig. 1. Neutral beam computer hardware configuration.

Fig. 2. Neutral beam console.

addition to providing the normal ModComp console functions. The interrupt panel is a specialized interface converting external events into computer interrupts. It makes available the five priority interrupts (level 7 through B) plus 8 service interrupts (SI) and 8 data interrupts (DI) related to the panels device address (i.e., if device address is #20, then the 8 SI's and DI's are addressed #20 through #27). All data interface is performed through CAMAC, an IEEE standard multiplexed data acquisition system. All data for power supply control and monitoring, for acquiring signal waveforms and microprocessor processed data and for specialized equipment interfaces is connected through the CAMAC system. Finally, an intercomputer link interfacing the Neutral Beam Classic to a Physics Diagnostic Data Acquisition system (Ref. 3) provides archiving and analyzing all neutral beam data on the diagnostic computer system with the Doublet III physics data.

3. SOFTWARE CONFIGURATION

The software configuration for the neutral beam computer system is illustrated in Fig. 3. The software contains two independent subsystems operating in parallel with limited communications. The data acquisition subsystem was largely designed, built, and supplied to General Atomic from the University of California, Lawwrence Berkeley Laboratory. This subsystem functions to sequence an experimental shot of the neutral beams, acquire, process, and display all of the physics diagnostics data for the beams, and provide the interaction and displays for users at the control console.

The periodic scan and update subsystem is a refined version of the existing Doublet III control system. This subsystem functions to provide monitoring and control for power supplies, timing systems, water cooling, and all other systems necessary for operation of the beamlines, and dynamic updating of all signals currently being displayed.

3.1. DATA ACQUISITION AND PROCESSING SUBSYSTEM

The data acquisition subsystem consists of five elements: dispatcher, screen manager, data base, interrupt events, and neutral beam applications software.

3.1.1. Dispatcher

The dispatcher is the central point of the acquisiton subsystem. The dispatcher performs the function of system sequencing and is entirely driven by the ModComp intertask message service. The dispatcher receives descriptions of events and conditions that are required to create desired

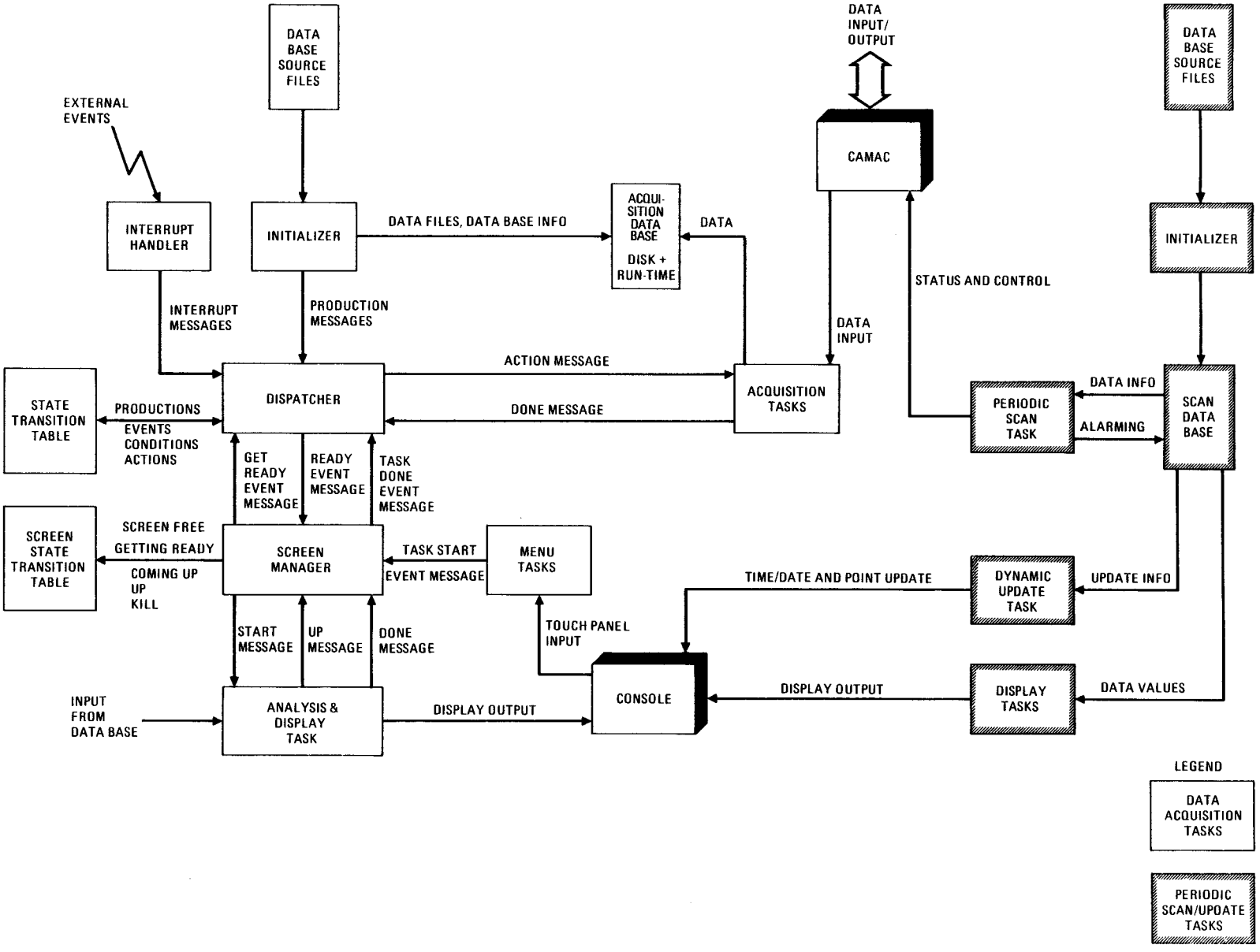

Fig. 3. Neutral beam software configuration block diagram.

actions, keeps track of related events and conditions and performs the actions when all conditions are met.

The dispatcher receives data termed productions, upon initialization of the system. The productions are a set of specific conditions that are required to be met before each action can be performed by a specific server (task).

An example of a production is illustrated in Fig. 4. The productions are written in a structured language format which is internally converted to a stream of intertask messages by a preprocessing task.

The dispatcher keeps track of all the events, conditions, actions and servers in a state transition table. This table is built for the dispatcher from a stream of intertask messages from an initializing task reading the productions from a source file.

Once the state transition table is built, the dispatcher will receive events (external or internally generated), decide whether an action is to be performed and if so, send an action message to the server task. When the server completes its action, it will send a done event message to the dispatcher which will be recorded in the state transition table. The done event could be the condition necessary to trigger another event.

3.1.2. Screen Manager

The screen manager handles all requests for displays and keeps track of the state of each screen so that the display requests can be properly sequenced. Each screen can be in one of five states: FREE, where the screen is available of a display; GETTINGREADY, where the screen is free and a request for a display is made but the productions defining the display has not met its conditions; COMINGUP, where a display has been activated and is coming up on the screen; UP, where a display is up on the

```
*     beam 210 source LO

when( SHOTSTART21LO )
      request display(1,BLOTDS,"------")

      request task(GETRDY,"21LO--")
      request task(TIMECO,"21LOPS")
      request task(SLOWIN, "21LOWD")
      request task(FASWIN, "21LOFD")
      request task(MISCAQ, "21LOBL")
      request task(THMCAQ, "21LOTC")
      request task(WFLOAQ, "21LOAL")

when( PSREADY21LO )
      request task(SPSINT, "21LO--")

when( BEAMOFF21LO )
      request display( 1,WAVEDS, "21LOSS")
      request task(FIRINT, "21LO--")

when( BEAMOFF21LO )
   and( task(THMCAQ, "21LOTC") )
      request task(FASWAQ, "21LOFD")

when( ABORT21LO )
      request task(ABOINT, "21LO--")
```

Fig. 4a. Shot sequence productions.

```
definetask (FASWAQ)
      formsgs( "21UPFD" )
            resetevents( SHOTSTART21UP )
            preconditions( BEAMOFF21UP, task(FASWIN,msg) )
            initializedone
            endfor
      formsgs( "21LOFD" )
            resetevents( SHOTSTART21LO )
            preconditions( BEAMOFF21LO, task(FASWIN,msg) )
            initializedone
            endfor
```

Fig. 4b. Typical task production.

screen; and KILLING, where a display has been sent a kill event to free the screen.

The screen manager receives a request in the form of a start task event message either from a previous display task, a menu task or an event created from the dispatcher. The screen manager generates a GETREADY event to the dispatcher to ensure all conditions for the task as defined in the productions have been met. When all conditions are met, the dispatcher will send a READY event message to the screen manager who will send a START event message to the display. The display will either send an UP or DONE event message depending whether the display remains in control of the screen. If a DONE message is received, the screen manager will send a TASK DONE to the dispatcher and also set the screen state to FREE. The screen manager keeps a circular request buffer for held requests received while a previous request is being processed.

3.1.3. Acquisition Data Base

The acquisition data base contains all necessary information for the acquisition, analyses, and display tasks. This data base is divided into a disk resident part and a runtime or memory resident part. The run time data base contains all information and data accessed frequently where delays in disk accessing cannot be tolerated. The disk data base contains all information accessed infrequently such as pictogram files and waveform data files.

The data base is built by an initializer task reading source files and creating a directory and data structure within the run time and disk data bases.

Access to the data bases is performed by messages sent to the routine data base manager. The request messages sent and reply messages are received through the intertask message service. Each request and reply consists of a single logical entry.

3.1.4. Interrupt Event Handling

The interrupt event handling is a specialized set of hardware and software to convert external timing events to message events for the dispatcher to trigger sequences of functions for the various operating modes. The interrupt system consists of an interrupt handler connecting to the five ModComp priority interrupts (level #7 - #13) and the interrupt interface's eight SI and DI levels. When an external event occurs, the interrupt handler determines the level, generates a corresponding message, and sends the message to the dispatcher.

The interrupt message service is not part of the ModComp intertask message service as of MAX IV Revision G. Installation of the interrupt message service required modifying the existing ModComp interrupt REX service and sysgen macro's in addition to sysgening in the interrupt message REX service.

3.1.5. Applications Software

All of the applications software, whether it be a acquisition task, analysis task, or display task has the same structure (Fig. 5). Each task is activated by a message from the dispatcher. The message identifies the data base data on which the task is to perform its function. The task will acquire the appropriate data base parameters it needs to perform its function, performs whatever functions are necessary, and send back to the data base any output that could be required by another task. Finally, the task will send back a done message to the dispatcher and will go to obtain another message. When no messages are present, the task will exit.

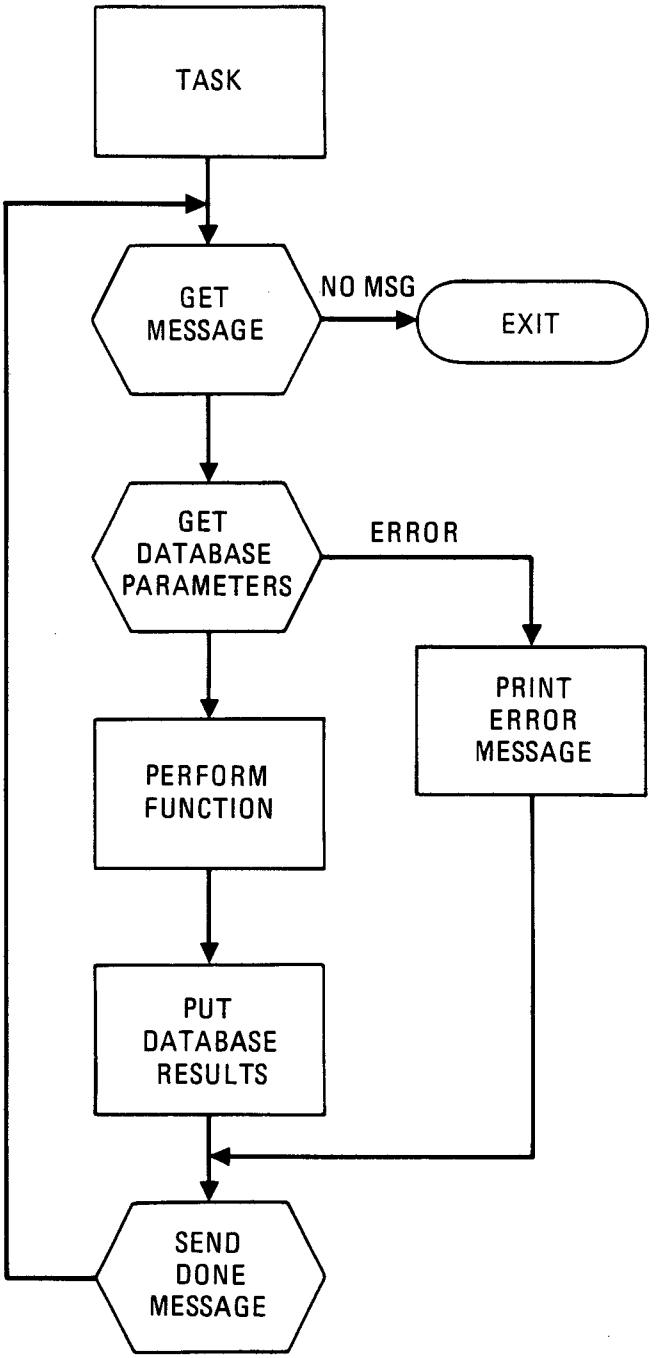

Fig. 5. Application task general structure.

3.1.6. Normal Operation

The normal sequencing of the neutral beam system can be illustrated by the shot sequence productions in Fig. 4a. The sequence illustrated is defined for one beamline source. Since there are two ion sources per beamline and two beamlines per computer, there are four similar sequences defined in each computer. In addition, since each of the ion sources can be operated simultaneously and asynchronously in a different mode, there can be four different sequences being processed at different states. For discussion purposes, only one sequence will be described. The system for each ion source is driven from five external interrupts. Each interrupt is defined as a dispatcher event. When a decision is made to start a neutral beam shot, the SHOTSTART interrupt is generated. This interrupt event causes all tasks that normally initialize the acquisition hardware to be activated. When all the hardware power supplies are ready, the PSREADY (power supply ready) interrupt event which activates a task acknowledging that the computer is ready. Nothing occurs until the beam hardware firing sequence is complete, which is indicated when the BEAMOFF interrupt event triggers the computer to acquire and process all data related to the shot. A fourth event not illustrated in Fig. 4a is READWFLO event which occurs asynchronous to normal operation. It uses a microprocessor which processes water flow and temperature data to determine beam energy losses in various points of the beamline. This event indicates when all subsystems have completed their processing. The final event ABORT, occurs anytime during the normal shot cycle and causes a task to initialize both the hardware and software to terminate current shot and setup for the next shot.

Currently, the sequences discussed are used for all modes of operation for an ion source. Currently being implemented are events defined for each mode and a separate production sequence for each mode, since all functions need not be performed in all modes of operation.

3.2. PERIODIC SCAN/UDPATE SYSTEM

The control of the neutral beam system has been segregated into a separate system, in part because of unique requirements, and in part to leave the data acquisition system undisturbed.

The unique requirements include a large number of signals which require frequent, if cursory, attention. Approximately 1200 inputs and outputs will be defined and checked for status, value changes, or updates every 1/2 second. This in turn has required a data base optimized to support the periodic scanning volume, together with tasks which support demand requests.

The following functions are performed by the Periodic Scan System:

1. Routine Monitoring. Automatic periodic surveillance of beam line status at all times.

2. Engineering Unit Conversion. Translation of all values between binary representation and correct Engineering Units.

3. Alarming. Warnings to the operator of status changes as they occur.

4. Logging. A printed log of all beam line status changes and all operator-issued or program-issued commands, in chronological order.

5. Operator Status Displays. Generation and display upon operator demand of summary or detailed machine status.

6. Operator Commands. Transmission of individual commands from an operator to perform all necessary system and subsystem functions.

7. Automatic Procedures. Execution of lists of detailed commands and status checks for the firing of beams in various operating modes.

8. Data Base Generation. Generation and maintenance of the tabular data necessary to drive all system tasks as pure procedures.

9. Status Storage and Retrieval. Archiving and restoration of system status for record keeping and shot reproducibility.

The major elements of the periodic scanning system include the scan executive, dynamic update, and console displays. These systems function in a manner similar to other computer systems used for process control.

3.2.1. Periodic Scan

The periodic scan program functions to process all dynamic points in the system on a periodic basis. The scan runs on a half second cycle and operates from a data base divided between a disk and memory-resident part, and a group of global tables.

The data base is built by an initializer program from either a set of source files or interactively from a terminal. The run-time, memory resident, part of the data base contains all the information necessary for the periodic processing. The disk part of the data base contains information which supplements the run-time part, but is not necessary for processing (i.e., point descriptions). The global tables contain all current values and status of signals, in addition to queues for dynamic alarm messages and task activations.

The periodic scan task is activated when the system is initialized, reads into local memory the entire run-time data base, connects to a timer and operates at its half second cycle.

3.2.2. Dynamic Update

The purpose of the dynamic update task is to periodically update all values currently being displayed, in addition to outputting the current date and time on each console screen at a one second interval.

The dynamic update task operates from two global tables. The first, the Console Descriptor Table defines all screens contained in the system and indicates whether there are points to be updated on the screen. The second, the Dynamic Update Table, contains a series of update commands for each screen defining the points to be updated, type of update and position on the screen for the update. The Dynamic Update Table is built by display tasks after a display with dynamic values has been put up.

The dynamic update task is activated upon system initialization and at a one second interval updates the time and all dynamic parts on each screen. The time is updated regardless of what type of display is up.

3.2.3. Console Display

The console display system was duplicated from the Doublet III system (Ref. 4) to keep consistency between it and the neutral beam system. The duplication included the display formats between the two systems and the software interface between display tasks and the system software. This effort provided two benefits. First, a complex display system was implemented in the short period of time necessary for operation. Secondly, it enabled the development of displays on one system to be transferred to the other with minimal effort.

Since the display screens were interfaced to the computer through a Grinnel controller, it was necessary to develop a symbiont to convert display requests to Grinnel controller commands. A set of high level language subroutines were written to allow the display task to build buffers of display requests to the symbiont. The control of the display tasks were performed by a keyboard menu task brought up on the touch panel screen. This keyboard task functions to select the dynamic displays on the other screens and provide a means of input to the displays.

The display tasks perform the action requested and exit keeping track of all necessary display information through a global common region. In addition to the console descriptor, the display task builds update buffers in another global common region used by the dynamic update task for periodic updating of values being displayed.

Since the dynamic update display tasks and data acquisition displays use the same hardware resources, it was necessary to interface the two systems to prevent conflicts in display updating. It was decided that the screen manager on the data acquisition system would clear the global tables used by the dynamic displays when a display is put up on the screen. This clearing will disable all the updating on the screen and require a dynamic display to be reselected form the typewriter menu before updating will resume for that particular screen.

4. SUMMARY

At General Atomic, we have successfully combined two distinct software subsystems on one computer. The subsystems operate independently but plans are being developed to combine the two subsystems since information will be passed between each.

A standard data base interface will be made so that a task requesting information will need not know in which subsystem the information is contained.

A command language will be developed to provide control for the four neutral beam sources connected to the system. This language will be similar to the command language on the Doublet III control system but will possibly be structured to handle from different source input streams. Once done, this language will provide programs used to sequence both the periodic scan system and the data acquisition system.

REFERENCES

1. Kohli, J. C., et al., "Instrumentation and Control of the Doublet III Neutral Beam Injector System," General Atomic Company Report GA-A15634, November 1979.
2. Drobnis, D. D., et al., "Computer Control of Tokamak Operation at Doublet III," General Atomic Report GA-A15661, November 1979.
3. Glad, A., et al., "Acquisition, Archiving, and Analysis of Doublet III Diagnostic Data on a Distributed Computer System," General Atomic Company GA-A16048, September 1980.
4. Glad, A. S., "Control Console Generation From Independent Devices," General Atomic Report GA-A14184, October 1976.