PROCESSOR/PROCESS MANAGEMENT
INTERRUPT HANDLING AND TIMER MANAGEMENT
SOFTWARE SUBSYSTEM DESCRIPTION
EXTENDED OPERATING SYSTEM, 3A PROCESSOR

CONTENTS PAGE

1. GENERAL

1.01 This section describes the interrupt assignments and processing as implemented by the 3A Processor Extended Operating System (EOS). A functional description of the EOS timer facilities and their management is also provided.

1.02 This section is being reissued to update reference documents in paragraph 1.03 an to modify Fig. 2, 3, 4, and 6. Revision arrows have not been used to denote minor changes. Equipment Test Lists are not affected.

1.03 The following sections contain descriptions of the various EOS functions related to this section.

SECTION	TITLE
254-300-110	3A Central Control, Description, Common Systems
254-300-120	3A Central Control, Theory of Operation, Common Systems
254-340-014	Memory Protection and Organization, Software Subsystem Description, Extended Operating System, 3A Processor
254-340-030	Processor/Process Management Creation, Event and Communication Control, Software Subsystem Description, Extended Operating System, 3A Processor
254-340-052	Device Handlers, Software Subsystem Description, Extended Operating System, 3A Processor
254-340-054	Terminal Administrator, Software Subsystem Description, Extended Operating System, 3A Processor
254-340-062	File System, Software Subsystem Description, Extended Operating System, 3A Processor
254-340-082	System Utilities, Software Subsystem Description, 3A Processor
254-340-084	Resident Maintenance, Software Subsystem Description, Extended Operating System, 3A Processor
254-340-106	Macros and Glossary, Software Subsystem Description, Extended Operating System, Common Systems

1.04 The following assembly units contain the code related to interrupt handling and timer facilities management in EOS.

- The Common Utility program (CUTIL), PR-4C622, performs the interrupt processing for a panel matcher interrupt.
- The Common Initialization program (CINIT), PR-4C618, performs the interrupt processing caused by an error interrupt or a maintenance channel interrupt.
- Control System Internal Timing program (TIMEAU), PR-4C144, performs all control system interval timing in the system.
- The Interrupt Service routine (INTSRV), PR-4C113, processes peripheral device interrupts.
- The Operating System Table (OSTABS), PR-XXXX (specified by an application), contains all interrupt transfer vectors and system generation parameters.

1.05 The glossary contained in part 5 of this section defines the terms, abbreviations, and acronyms used in this section. Refer to Section 254-340-106, Extended Operating System Macros and Glossary for a list of associated terms, abbreviations, and acronyms.

A. Overview of Interrupt Handling

1.06 Interrupts provide a mechanism whereby the EOS or application processes/tasks can be notified when some specific action (normally hardware) has occurred. There are a number of ways the EOS or an application can respond to an interrupt. A new process or task may be readied for execution, a message may be sent to a process or task, or an event flag may be set in a task as the result of an interrupt. For example, depressing a key on a teletypewriter causes a demand interrupt which must be reported to the proper task(s) to enable the input of a TTY message.

1.07 In order for application tasks to interface with the EOS, certain items are required:

- The interrupts must be equipped and the appropriate interrupt masks must be specified to the system.
- Identification of appropriate interrupt routines which process specific interrupts must be established.
- Interrupt priorities must be established and implemented.

B. Overview of Timer Management

1.08 The timing facilities provided by the EOS include a 24-hour system clock, a system calendar, and various active timers. There is a timer facility which provides the capability for a routine to be called periodically and to be placed in the EOS system tables. This facility executes a subroutine call to a specified subroutine at the end of a specified time interval. These calls take precedence over the timer kept in the EOS timer list. The system calendar is kept as a nonnegative displacement from March 1, 1972. The system clock is maintained in increments of 10ms. The EOS keeps a list of active timers, each of which is a block which specifies to the operating system that a process is to be informed when a particular time has occurred or when an interval of time has expired. A process is informed of these "active timer occurrences" through messages or events.

1.09 All functions related to timing are initiated by macro calls. The system macros pass parameters to assembly unit TIMEAU through service (or supervisor) calls (SVC 9). The six basic functions related to timing are listed below, with the name of the macro which invokes each function:

- Set the system time (SET_TIME)
- Read the system time (READ_TIME)
- Set the system date (SET_DATE)
- Activate a timer (ACT_TIMER)
- Deactivate a timer (DEACT_TIMER)
- Identify subroutines called by TIMEAU (TIMER_SUBR)

1.10 The timer facilities provided by the EOS can be used in many different ways. Some of these ways are:

- To determine the execution time of a routine. Note that a routine that activates higher-priority routines will be placed in a suspended state while the higher-priority routine executes.
- To activate a routine at a fixed time.
- To activate a routine upon the expiration of a specified time interval.
- To determine the time of day.

1.11 The timing functions are designed to be very flexible. They provide the user with all possible timing facilities without the necessity of managing the clock and are general enough to meet present and future timing requirements.

2. FUNCTIONAL DESCRIPTION OF INTERRUPT HANDLING

A. Interrupt Structure

2.01 The interrupt structure of the 3A Central Control (CC) consists of 16 separate interrupt levels specified by an interrupt set register (IS) and a corresponding 16-bit interrupt mask register (IM) which is used to inhibit or enable any of the interrupts. When an interrupt occurs, the corresponding bit in the IS is set. The IM prevents an external interrupt from interrupting the process unless it is specifically desired. A "1" set in the IM inhibits the corresponding interrupt; a "0" enables the interrupt. This mechanism provides the capability for the relative priority of the interrupts to be effectively controlled during the execution of an interrupt routine.

2.02 During certain processing, it is desirable to block all interrupts, eg, system initialization. This is accomplished by setting the block interrupts bit (BIN) in the system status (SS) register. When the BIN is set, all interrupts are blocked, regardless of the IM. When BIN is reset, interrupts are enabled according to the IM.

2.03 The current 3A CC interrupt assignment is:

Level 0—Available for application usage*
Level 1—Available for application usage*
Level 2—Available for application usage*
Level 3—Panel Matcher
Level 4—Available for application usage*
Level 5—Error Interrupt
Level 6—Available for application usage*
Level 7—Other 3A CC Interrupt
Level 8—Available for application usage*
Level 9—Timer Interrupt (10 ms)
Level 10—TTY and TDC Controllers (even)
Level 11—TTY and TDC Controllers (odd)
Level 12—Available for application usage*
Level 13—Manual panel execute
Level 14—Assigned to Direct Memory Address (DMA) (if DMA is present)
Level 15—Available for application usage.*

*This is true only for generic releases G2A and later; on releases prior to G2A they cannot be used.

Interrupts 10 and 11 are usually assigned to the TTYCs and TDCs, with interrupt 10 assigned to the even-numbered units and interrupt 11 to odd-numbered units. The EOS does not specifically require this assignment; however, there are many stand-alone programs that do require it. When devices are assigned to levels which are used by file system devices, the following conventions are followed:

(a) The device assigned to that level must be supported by the file system.

(b) All devices assigned to a given level are on the same hardware interface to the processor, ie, all on a parallel channel (not necessarily the same one) or all on a serial channel.

B. Interrupt Hierarchy

2.04 Interrupts are serviced in order of priority, which is established by the structure of the IS register where bit "0" represents the highest priority and bit "15" represents the lowest. However, relative priority of the interrupts may be controlled by setting an interrupt mask during the execution of interrupt routines. In this case a routine may set an interrupt mask in which a "0" will designate a priority for a specific interrupt which will give it higher priority than the interrupt being serviced by the routine. Interrupts of equal or lower priority are identified by bits set in the IM.

2.05 The EOS requires specific priorities for the interrupts which it controls and sets the appropriate masks to control these priorities. The interrupt hierarchy is defined in OSTABS by use of the HIERARCHY macro. An application establishes the hierarchy for interrupts associated with it in exactly the same manner. The hierarchy as defined by EOS is:

(a) Highest—Interrupts 3 and 13. When these are executing, no other interrupts can be processed.

(b) Medium—Interrupts 1,2,4,5,6,7 and 9. In this group, priority is in normal numerical order, ie, 5 is higher than 7, etc. Also, 3 and 13 are enabled. For example, if interrupt 7 is being serviced 3, 13, and 5 could interrupt.

(c) Lowest—Interrupts 10 and 11. In this case 10 handles even numbered controllers and 11 the odd. During execution of these interrupts the IM would provide the capacity for all higher priority interrupts.

Interrupt 14 is assigned to the direct memory access (DMA) on the systems equipped with DMA. The priority assignment in this case is between 9 and 10. The application is constrained to priorities below the medium level assigned to the 1,2,4...etc, group. Using releases G2A and after, applications may use interrupts 1,2,4, and 6 and may be of any priority. Interrupts 8,12, and 15 are available for application usage.

2.06 In addition to the basic hierarchy, the EOS also defines three miscellaneous interrupt masks:

(a) TASK—This label defines a mask which is used for all task level programs. In this mask all interrupts are enabled, except those which are undefined to the HIERARCHY macro.

(b) SVC—This label defines a mask which is used for all SVC routines. This mask blocks all interrupts except 3 and 13.

(c) ALL—This label defines a mask which is used when all interrupts are to be blocked except 3 and 13.

Interrupts 3 and 13 are never blocked since they are connected with manual operations of the status panel. They will not occur during the normal processing routines since specific manual requirements must be fulfilled before the function, eg, manual requests for system utilities or diagnostic functions.

C. Interrupt System Function

2.07 When an interrupt occurs, all unmasked bits of the IS are ORed together and create a signal which is used to jam-set the microaddress register (MAR) to a fixed location in microstore (Fig. 1), depending on the condition of the BIN. When the BIN is set, all interrupts are blocked. The microinstruction sequence tests the IS for the highest level of interrupt and translates the bit position into a data constant which points to a main memory location containing a pointer to the appropriate interrupt routine. In the EOS this location is defined as the interrupt transfer vector.

2.08 The interrupt transfer vector is based on location zero in the first 4K of write-protected store. It is defined in OSTABS and consists of a series of two word entries. Each entry consists of a branch long instruction (BASI) to the subroutine which services the interrupt.

3. FUNCTIONAL DESCRIPTION OF INDIVIDUAL INTERRUPTS

A. Panel Matcher Interrupt (3) (Fig. 2)

3.01 The conditions for this interrupt are established by a craftsperson entering a TTY message enabling the panel matchers to cause an interrupt when the exact match of a specific address or data word is detected. When a match is detected, interrupt 3 will occur. Entry from the transfer vector is to the MATCHINT subroutine in the Common Utilities program (CUTIL). MATCHINT increments an interrupt counter and tests to insure that the number of interrupts which have occurred since the counter was cleared is within a prescribed limit. When the limit is exceeded the interrupt is disabled. When the number is within limits, MATCHINT will cause the registers of the program being interrupted to be saved and call the UTILPROC subroutine in CUTIL to process the request. The function may be to monitor or load certain store locations or registers as requested by the original messages. Details of the monitor and load message functions are described in Section 254-340-082.

B. Error Interrupt (5) (Fig. 3)

3.02 An error interrupt is initiated when bit 14 or higher is set in the Error Register (ER). This group of bits is ORed together and the resulting signal sets bit 5 in the IS. The type of errors that can cause interrupt 5 are:

(a) A My store write protect violation causes an initialization in EOS-based systems and causes an interrupt in other 3A Processor applications (bit 11).

(b) The Other store write protect error (bit 14)

(c) The Other store error (bit 15)

(d) The Other store fast time-out (bit 16)

(e) The I/O multiple channel select (bit 17)

(f) A program timer reset received by the on-line 3A CC (bit 18)

(g) A switch received by the on-line 3A CC (bit 19)

(h) An I/O channel error (Parity Low bit)

(i) An I/O bad parity received (Parity High bit)

3.03 The ERR_INT subroutine in the common initialization program (CINIT) is the error interrupt routine specified by the transfer vector. ERR_INT causes the registers of the interrupted program to be saved and clears the IS. All information is collected from ER, and if there is only one error, the error causing the interrupt is identified. Once this is complete, the ER is cleared. Based on the error, a branch is made to the appropriate error correction routine. For all memory errors an attempt is made to initialize the off-line store. When this is successful, a memory update is performed and, on successful completion, an operational test is performed on the off-line store to determine if it can be written into and read from without error. Any failure is reported by an error message on the TTY. An attempt is made to clear I/O errors by clearing and reinitializing the channel. A sanity test is run on the other 3A CC to determine its status when the error causing the interrupt is the program timer time-out or an erroneous switch. Failures are reported by appropriate error messages.

C. Other 3A CC Interrupt (7) (Fig. 4)

3.04 The other 3A CC (off-line) can initiate one of three different requests which will cause interrupt 7. The three are: a stop request (SWINIT) resulting from a stop and switch activity, a switch-completed notification (SWCOMPL) given when the other processor has started execution, or a zero program timer request (ZEROPT).

3.05 The interrupt routine which processes interrupt 7 is MCHINT in CINIT. Initial processing is the same in each case and consists of saving the registers of the interrupted program and determining which of the three functions was requested. When the function has been identified, the proper routine is given control. The SWINIT function is performed by a microsequence and consists entirely of stopping the processor. SWCOMPL is entered after the other processor has taken over execution and is operating properly. The routine performs general restoral activity on registers and prepares the processor to respond if another stop and switch is requested.

D. The 10 ms Timer Interrupt (9) (Fig. 5)

3.06 The timer interrupt (9) is a feature of the EOS which provides the capability for calling of up to 8 subroutines from the 10 millisecond timer interrupt service routine. Three of the subroutines are required for EOS; the other five can be application-designated subroutines. The user specifies an entry point and the time interval between calls, in increments of the number of 10ms intervals between calls. The various subroutines and methods of establishing the timer subroutine definition table are described in Part 4.

3.07 Timer interrupts are initiated from the 10ms lead of the timing counter to bit 9 of the IS. The transfer vector specifies the TIMEINT subroutine of the timing control program (TIMEAU) as the interrupt routine. TIMEINT sets up the return by saving the current state and registers, detects stuck interrupts, identifies whether an interrupt or system call is being serviced, and sets the appropriate IM. Overlap conditions are checked (overlap is the condition which results when another interrupt occurs before processing is completed on a previous interrupt). The MISTIME counter is tested and incremented (MISTIME is the count of the number of times the subroutine loop has been started since the last return to the dispatcher). The maximum number of times that this can occur is defined by the constant MAX_MIST specified in OSTABS. The current value of this constant is 5. The count (SYS_CNT) of the interrupt level is decremented when timer interrupt overlap exists. When the processing is complete, 10ms is added to the time-of-day (TOD) count and the TOD timer is updated as required.

3.08 The interrupt subroutine loop is composed of three required EOS subroutines and as many application subroutines as have been defined (up to five). The loop is restarted at every interrupt up to the value specified by MAX_MIST. When MISTIME reaches or exceeds MAX_MIST, control is passed to an application routine which is defined in OSTABS as MAX_RTN. MAX_RTN is defaulted to a routine in TIMINT which resets MISTIME and the bit representing interrupt 9 in the word PROCS_ACT_INT, marking interrupt 9 as no longer active. Control is then passed to the dispatcher. The required EOS subroutines are:

(a) The activate timed activities subroutine (TIMEAU)

(b) The maintenance time subroutine (MAI_TIME)

(c) The file system time subroutine (FIL_TIME).

3.09 These subroutines and any defined application subroutines are specified in the operating system tables and maintained in a time subroutine table (TIM_SBR) in the system constant area (INTCALL). Each table entry is four words long and contains the time between calls in 10ms intervals, the time remaining until the next call, and the subroutine address.

3.10 The TIMEACT subroutine checks the entries in TIM_SBR and when times are reached, the appropriate replies are sent to the requesting processes. The MAI_TIME subroutine controls calling and adjusting priority of the periodic maintenance task. The interval between calls to the maintenance loop is specified by the application by defining the number of 10ms intervals. The FIL_TIME subroutine maintains a count of 10ms intervals for file system timing. These subroutines are described in part 4.

E. The TTYC or TDC Interrupt (10 or 11) (Fig. 6)

3.11 Interrupts 10 and 11 are used for the teletypewriter and tape deck controller interrupts. Interrupt 10 is used for even numbered units while 11 is used for odd numbered units. Each character entered will initiate an interrupt. The interrupt transfer vector specifies the INTSRV routine, and depending on the unit initiating the interrupt, either program unit INTSRV10 or INTSRV11 will perform the processing. The processing consists of setting up a return to the interrupted routine and identifying the interrupt in the FIXED area reserved for EOS tables and system constants. The only difference in processing is the interrupt number which is written into the FIXED area.

3.12 The program unit INTSRV_BEGIN is common to all interrupts. The processing consists of identifying the appropriate channel, either parallel or serial, polling the identified channel to locate the device which initiated the interrupt, and restoring interrupt capability once the device is located. The designated device driver is identified and control is passed to the driver. When the device is a teletypewriter, the driver is TTYDRV and entry is at TTYDRVI. When the device is the TDC, the driver is TDCDRV and entry is made at TCINTC. In each case the driver monitors and controls the device states, selects the proper operations whether read, write, etc. The driver then establishes an interface with the file system device task FILDEV.

F. Panel Manual Execute Interrupt (13)

3.13 The panel functions are executed by a panel-halt loop of microcode. The loop is initiated when the 3A CC is off-line and the MANUAL switch on the panel is operated. Under these conditions operating the EXECUTE switch on the 3A CC control panel sets interrupt bit 13.

3.14 When interrupt bit 13 is set, the states of the panel switches are interrogated by the panel-halt loop sequence and the functions designated by the switches are performed. After the requested function is executed the panel HALTED lamp is lit. The panel-halt loop sequence is then restarted by operating the HALT switch. Interrupt 13 is never blocked since there is no 3A CC code that the interrupt causes to be executed, and under normal processing conditions, the panel is disabled by the MANUAL switch.

G. Direct Memory Access Interrupt (14) (Fig. 7)

3.15 When a system is equipped with DMA, interrupt 14 is used. The interrupt is initiated by any device utilizing the DMA. The basic configuration of PROMATS utilizing DMA is described in this part. Application users may establish other configurations as required.

3.16 When an interrupt is invoked, the interrupt system will access the interrupt transfer vector for interrupt 14. The vector identifies the assembly unit INTSRV as the processing routine. Entry is at location INTSRV14. Processing at this location consists of setting up for the return to the interrupted program and identifying the interrupt in the EOS data area FIXED. Control is then transferred to the common interrupt processing program unit INTSRV_BEGIN.

3.17 INTSRV_BEGIN processing is identical to that described in paragraph 3.11. The control is passed to the proper device driver, which in this case, is the PROMATS DMA driver JHPROD, with entry at program unit ITPHDL. Processing consists of saving registers, accessing the proper device control block (DVCB), checking for diagnostic field interrupts, and identifying the proper channel and subchannel addresses. On each sequence a check is made to determine completion status. When the operation is complete, a return is made to the interrupted program via the EOS dispatcher. Until completion, a recurring interface is maintained with the EOS file system device task FILDEV to attach the DVCB and to perform requested functions such as read, write, etc. Details pertaining to device drivers are in Section 254-340-052.

4. FUNCTIONAL DESCRIPTION OF TIMER MANAGEMENT

A. Timer Subroutine Definition Table

4.01 EOS has the feature of calling application subroutines from the 10ms timer interrupt service routine (TIMEINT) in TIMEAU. The application subroutines are established in the EOS table by the user, who specifies entry point (including program name) and the time interval between subroutine calls. The time interval is specified in the number of 10 ms intervals between calls. When the timer interrupt processing does not finish before another timer interrupt occurs, the timer interrupt service routine restarts with the first subroutine in the table.

4.02 Each subroutine to be placed in the table is specified by the user with the TIMER_SUBR macro which is utilized at system generation time. The parameters for this macro are a list of three items with each item consisting of:

(a) Interval between calls in 10ms units.

(b) Assembly unit name.

(c) Program unit address.

4.03 Any number of TIMER_SUBR macros may be used and any number of items specified on one macro call. The present limit of application subroutines is five.

4.04 The table is constructed such that an entry of zeros following the last entry is necessary. Currently there is a maximum of eight subroutines which can be specified; three are required by EOS (TIMEACT, MAI_TIME, and FIL_TIME), which leaves five subroutines that can be specified by an application.

4.05 Each entry in the table (labeled TIM_SUBR) is comprised of four words. The first word is the interval between calls (TIM_BETWEEN_CALLS) in increments of 10ms; the second word is the time remaining between calls (REMAINING_TIME) in increments of 10ms; and the third and fourth words contain the twenty-bit subroutines address (SUBROUTINE_ADDRESS) of the required subroutine which is called when the time interval expires.

B. System Timer Functions

4.06 EOS timing functions are initiated at the interrupt level by interrupt No. 9 as covered in paragraphs 3.06 through 3.09 or by system calls via the system macros.

4.07 The timing system macros pass parameters to TIMEAU through a SVC 9 system call. Program Unit TIMEMSG in TIMEAU handles all requests for timer functions other than interrupt level.

4.08 The system timing functions may be related directly to the macro which invokes the function:

- Set the system timer (SET_TIME)
- Set the system date (SET_DATE)
- Activate a timer (ACT_TIMER)
- Deactivate a timer (DEACT_TIMER)

These functions may also be called by appropriate TTY messages. In these cases the initial entry is via interrupt 10 or 11 and processing is performed as described for those interrupts in Part 3.

4.09 To set the system time, a program executes a SET_TIME macro with the desired time as a parameter of the macro call. The desired time must be specified in hours, minutes, and seconds.

4.10 To set the system date, a program executes a SET_DATE macro with the desired date as a parameter to the macro call. The date is stated in month, day, and year.

4.11 When a program desires to "activate a timer", it executes the ACT_TIMER macro. A timer can be set to expire after an interval of time less than 24 hours from the current time, at the first occurrence of a particular time on the 24-hour clock, or at a given interval past the hour. These timers can be repetitive in that once they expire, they can be reset and placed back in the timer list or they can expire and not be reset. The interval or time at which to expire is a parameter of the ACT_TIMER macro. The event or message which will be sent when the timer expires is also a parameter of ACT_TIMER. This macro is used to awaken specified processes/tasks when a time interval expires, and can be accomplished by any process/task. This demand scheduling is invoked by using the repetitive and interval parameters of the macro.

4.12 The capability to deactivate a timer is provided by the DEACT_TIMER macro. Either an event number is specified which causes all timers of the specified process with that event number to be removed from the timer list, or a tag is specified, in which case all timers of that process with that tag are removed from the timer list. Table A contains the timer management functions program unit identifications.

5. GLOSSARY

5.01 The following terms and acronyms are contained in this section.

Alphanumeric—The letters of the alphabet and the digits 0 through 9.

Application—A set of functional system programs which use the services of the EOS.

Assembly Unit—A collection of code that is assembled or compiled as one entity. The assembly unit is the highest level of a modular program structure and may or may not contain functionally related subunits.

BIN—Block Interrupt Bit (disables all interrupts).

BTC—Block Timer Check.

Control Block—A group of words in memory which contain information relating to a piece of equipment.

CSECT—Pseudo-operation used to specify the beginning of a sequence of relocatable instructions.

DATASECT—A block of relocatable data, ie, a data CSECT.

Device Driver—A software unit which interfaces directly with a peripheral device hardware controller and interacts to control device functions.

DMA—Direct Memory Access.

Entry—A labeled location at which a CSECT or any logical block of code may be entered.

EOS—Extended Operating System.

EPL—ESS Progrmming Language.

Equipped—A predefined interrupt assigned a specific bit in the IS.

ER—Error Register.

ESS—Electronic Switching System.

Execution Module—Programs, subroutines, and all data and peripheral resources required to execute a task.

Exit—A location at which a program unit terminates and passes control to another location.

FILDEV—The general software interface between the file system and the device handlers.

Fixed—The primary data structure from which other data structures are linked.

IM—Interrupt Mask Register.

Interrupt—A special condition in which the current task of the processor may be suspended and another task (the interrupt service routine) initiated. The suspended task is resumed after the interrupt service routine is performed and if no further interrupts occur.

Interrupt Hierarchy—A structure which defines the priority order in which interrupts will be recognized and serviced.

I/O—Input/Output.

IS—Interrupt Set Register.

Location—The field in an address statement which specifies symbolic address.

Macro—A short precoded routine which is assigned a name by which it is labeled. On each call, the name is then replaced (expanded) into the sequentially coded lines of the routine.

MAINT—The maintenance task which supervises duplex operation.

MAR—Memory Address Register.

Microinstruction—A low level instruction which is used to form microinstruction sequences that are permanently stored in a read-only memory. The microinstruction sequences are used to implement the 3A Central Control instruction set and basic control functions.

Mnemonic—A combination of alphanumeric characters used in place of the binary code for an instruction which conveys the essential function of that instruction in a concise format, eg, ZCF: Zero the condition flip-flop.

Module—A small block of instructions within a CSECT which perform an identifiable subfunction.

Operation—A code which directs the action of the machine or other software, usually in a specific field of an instruction. It may be an opcode, pseudo-operation, macro, or EPL statement.

OSTABS—Operating system tables used by the application to define system resources, configuration, parameters, etc.

Parity High (PH)—A bit generated for odd parity for bits 15 through 8 in a word.

Parity Low (PL)—A bit generated for odd parity for bits 7 through 0 in a word.

Process—The execution of a series of programs whose order of execution is specified by a file of commands.

Process (EOS)—Dynamic execution of a command file, where each command implements a complete execution module.

PR-Number—A unique number assigned to an assembly unit listing which identifies the system to which the listing pertains.

Program Unit—A collection of code within an assembly unit which performs a well defined function.

PROMATS—Programmable Magnetic Tape System.

Pseudo-Operation—An operation which may be used in an assembly statement to control assembler activities but does not result in executable machine code.

PT—Program Timer.

SS—System Status Register.

SSP—System Status Panel.

Subroutines—A sequence of instructions called within another section of instructions to perform a specific function.

Symbol—A series of predefined characters or mnemonics which represent a binary value.

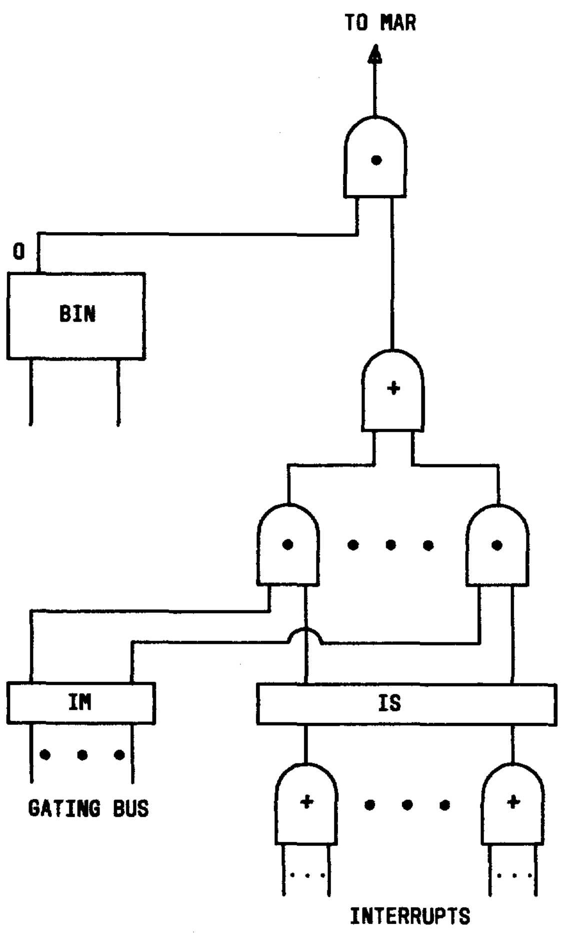

Fig. 1—Interrupt Logic

FUNCTION: INTERRUPT 3 (PANEL MATCHER) PROCESSING
THE CONDITIONS FOR THIS INTERRUPT ARE SET VIA A TTY MESSAGE ENABLING THE MATCHERS. AN EXACT MATCH OF SPECIFIED DATA OR ADDRESS WILL CAUSE INTERRUPT 3.

ASSEMBLY UNIT	ASSEMBLY UNIT MAJOR FUNCTION	PROGRAM UNIT	PROGRAM UNIT FUNCTIONS
CUTIL (PR 4C622)	COMMON UTILITY FUNCTIONS, LOAD OR MONITOR STORE LOCATIONS OR REGISTERS	MATCHINT	• CONTROL INTERRUPT COUNTER • SET UP RETURN • ALLOW OR DIS-ALLOW INTERRUPT • CALL PROCESSING ROUTINE
		UTILPROC	• INITIATE REQUESTED LOAD OR MONITOR FUNCTION

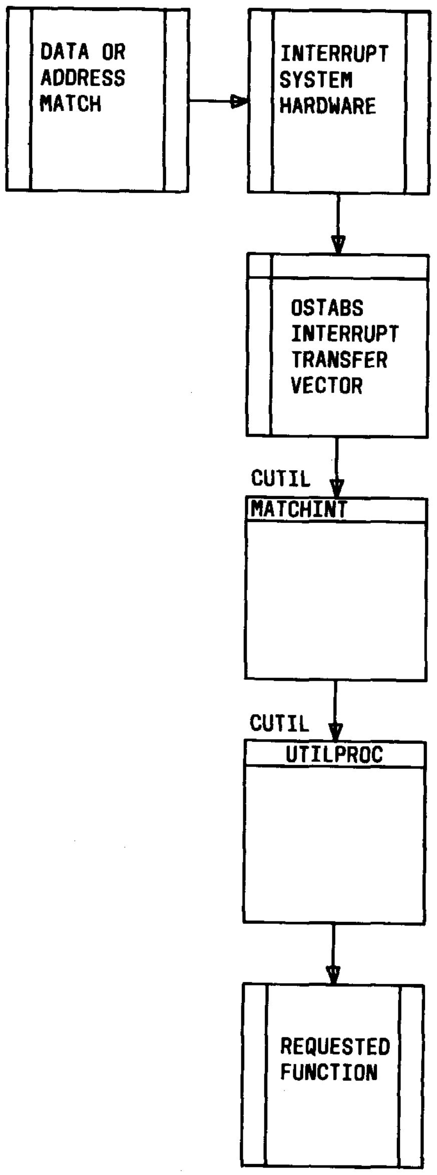

Fig. 2—Interrupt Processing, Interrupt No. 3, Panel Matcher

FUNCTION: INTERRUPT 5 (ERROR INTERRUPT) IS INITIATED WHEN ANY ONE OF THE BITS 14 THROUGH PH IS SET BY AN ERROR. THIS GROUP OF BITS IS ORED TOGETHER AND BIT 5 SET. ERRORS ARE: 14, STORE WRITE PROTECT ERROR; 15, STORE ERROR; 16, STORE TIME OUT; 17, I/O MULTIPLE SELECT ERROR; 18, PROGRAM TIMER TIME OUT; 19, ERRONEOUS SWITCH; PL, I/O CHANNEL ERROR; PH, I/O BAD PARITY.

ASSEMBLY UNIT	ASSEMBLY UNIT MAJOR FUNCTION	PROGRAM UNIT	PROGRAM UNIT FUNCTIONS
CINIT (PR 4C618)	COMMON INITIALIZATION FUNCTIONS AND ERROR RECOVERY ROUTINES	ERR_INT	• SET UP RETURN • IDENTIFY ERROR • CALL PROPER ROUTINE
		WRITE_ PROTECT_ OS OR STORE_ERROR_OS FAST_TIMEOUT_ OS	• PROCESSING COMMON TO THREE ENTRY POINTS. INITIALIZE STORE • PERFORM OPERATIONAL TEST
		IO_MULT OR IO_CHANNEL	• DISABLE THE CHANNEL IN EITHER CASE
		PT_RESET_ONL OR SW_ONL	• PERFORM A SANITY TEST ON THE OTHER CC
		IO_PARITY	• SAVE DATA FOR ANALYSIS. CORRECT PARITY

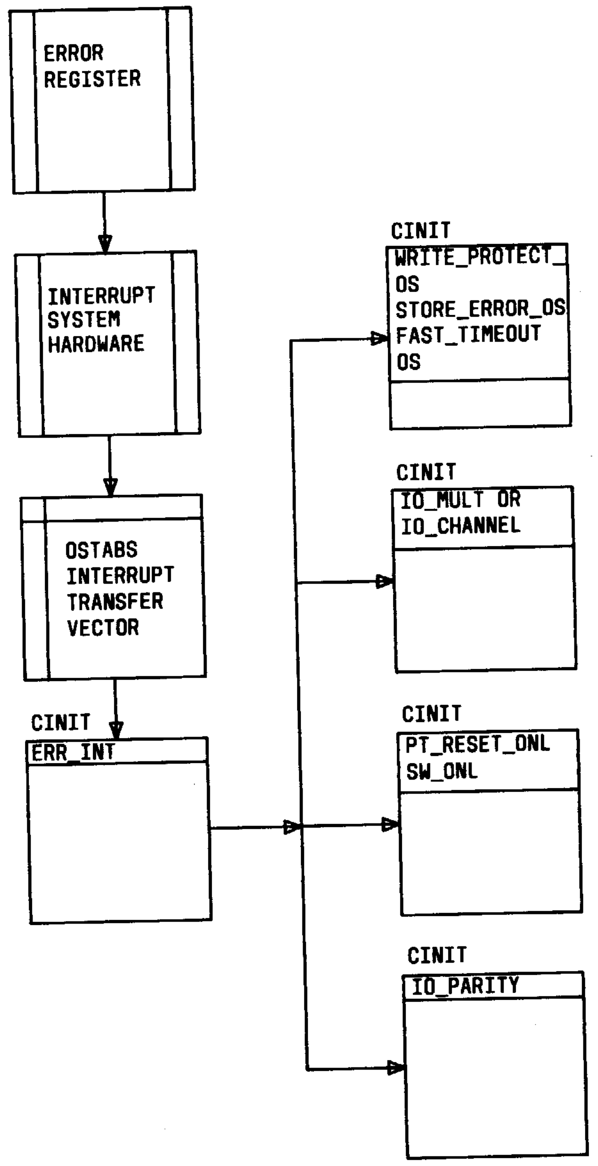
ERROR REGISTER
INTERRUPT SYSTEM HARDWARE
OSTABS INTERRUPT TRANSFER VECTOR
CINIT
ERR_INT
CINIT
WRITE_PROTECT_ OS STORE_ERROR_OS FAST_TIMEOUT OS
CINIT
IO_MULT OR IO_CHANNEL
CINIT
PT_RESET_ONL SW_ONL
CINIT
IO_PARITY

Fig. 3—Interrupt Processing, Interrupt No. 5, Error

FUNCTION: INTERRUPT 7 (OTHER 3A CC)

ASSEMBLY UNIT	PROGRAM UNIT	PROGRAM UNIT MAJOR FUNCTIONS
CINIT (PR 4C618)	MCH_INT	• SET UP RETURN • IDENTIFY FUNCTION
	SWINIT	• STOP THE PROCESSOR
	SWCOMPL	• RESTORE REGISTERS • SET UP PROCESSOR FOR STAND-BY MODE
	ZEROPT	• ZERO THE PROGRAM TIMER

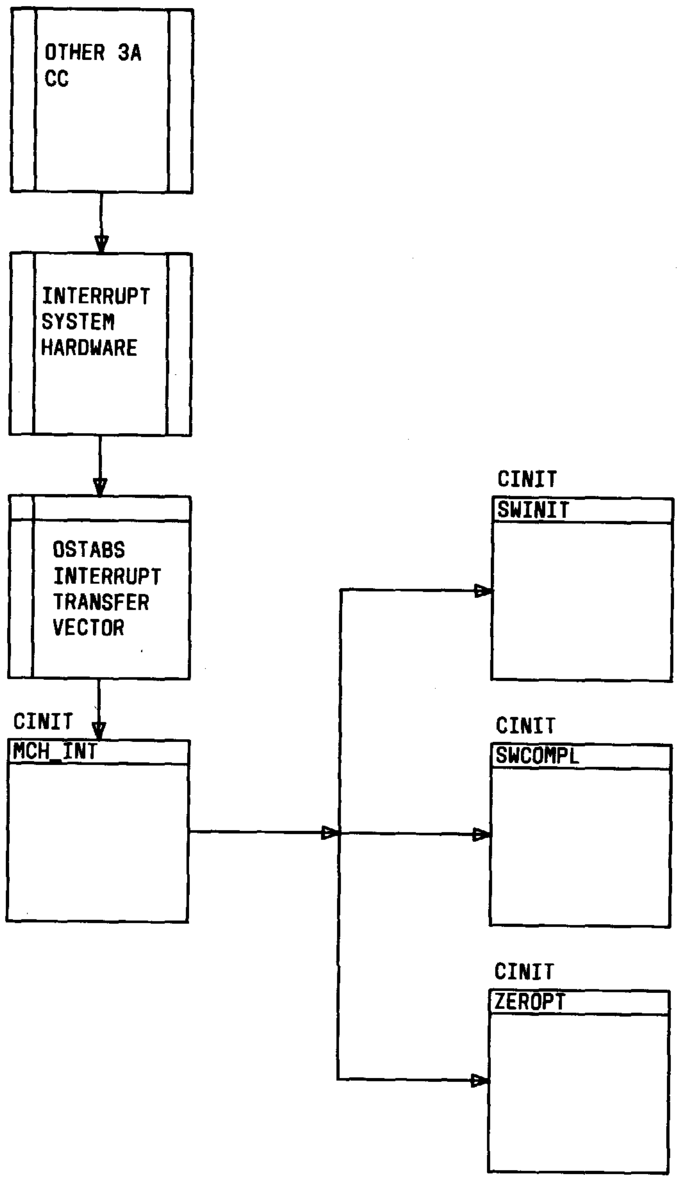

Fig. 4—Interrupt Processing, Interrupt No. 7, Other 3A CC

FUNCTION: INTERRUPT 9 (10 MS TIMER INTERRUPT)

ASSEMBLY UNIT	PROGRAM UNIT	PROGRAM UNIT MAJOR FUNCTIONS
TIMEAU (PR 4C144)	TIMEINT	• SET UP FOR RETURN • DETECT STUCK INTERRUPT • IDENTIFY IF INTERRUPT OR SYSTEM CALL • DETECT OVERLAY • INCREMENT MISTIME COUNTER INITIATE SUBROUTINE LOOP
	TIMEACT	• CHECK TIMER TABLE AND REPLY TO REQUESTING TASKS
	MAI_TIME	• CALL AND ADJUST PRIORITY OF MAINT TASK AT SPECIFIED TIME INTERVALS
	FIL_TIME	• MAINTAIN TIME COUNT FOR FILE SYSTEM ACTIVITIES

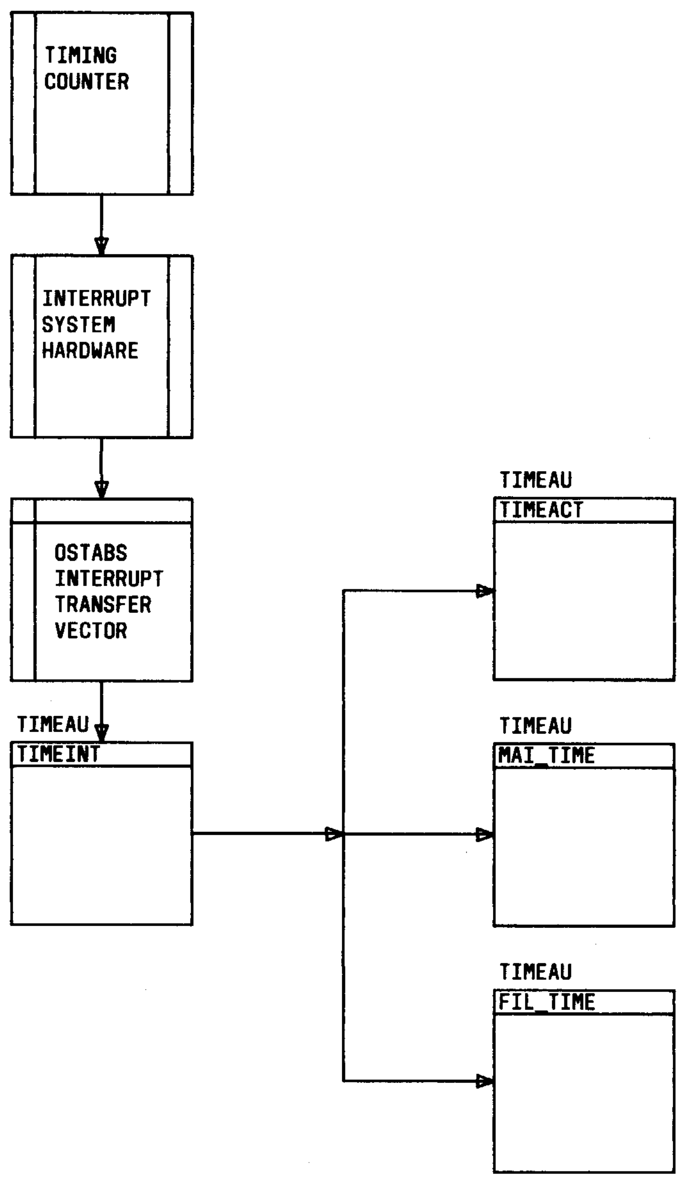

Fig. 5—Interrupt Processing, Interrupt No. 9, Timer

FUNCTION: INTERRUPT 14 DMA INTERFACE

ASSEMBLY UNIT	PROGRAM UNIT	PROGRAM UNIT MAJOR FUNCTIONS
INTSRV (PR 4C113)	INTSRV14	• SET UP RETURNS FROM THE INTERRUPT
	INTSRV_BEGIN	• COMMON PROCESSING FOR ALL INTERRUPTS
		• IDENTIFY CHANNEL (PCH OR SCH)
		• POLL DEVICES TO LOCATE INTERRUPTING DEVICE
		• REMOVE MASK TO RESTORE INTERRUPT CAPABILITY
		• IDENTIFY AND CALL PROPER DEVICE DRIVER
JHPROD (4R 4C209)	ITPHDL	• SAVE REGISTERS AND ACCESS DVCB
		• CHECK DIAGNOSTIC INTERRUPTS AND IDENTIFY CHANNEL AND SUBCHANNEL
		• CHECK FOR COMPLETION
FILDEV (PR 4C206)	OPRFNSH	• INTERFACES BETWEEN INTERRUPT AND PROCESS LEVEL

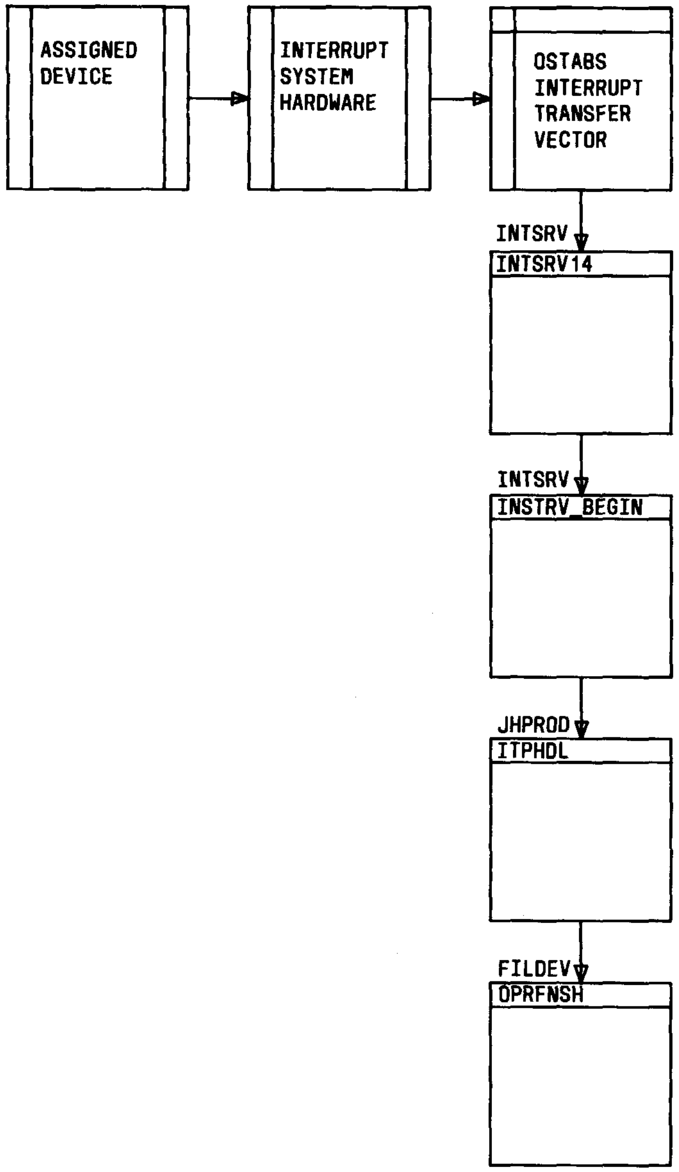

Fig. 6—Interrupt Processing, Interrupt No. 10 or 11, TTY or TDC

FUNCTION: INTERRUPT 10 OR 11 (TELETYPE OR TAPE DECK CONTROLLER)

ASSEMBLY UNIT	PROGRAM UNIT	PROGRAM UNIT MAJOR FUNCTIONS
INTSRV (PR 4C113)	INTSRV10 OR INSRV11	• SET UP RETURNS FROM THE INTERRUPT
	INTSRV_BEGIN	• COMMON PROCESSING FOR ALL INTERRUPTS • IDENTIFY CHANNEL (PCH OR SCH) • POOL DEVICES TO LOCATE INTERRUPTING DEVICE • REMOVE MASK TO RESTORE INTERRUPT CAPABILITY • IDENTIFY AND CALL PROPER DEVICE DRIVER
TTYDRV (PR 4C219)	TTYDRVI	• MONITOR AND CONTROL DEVICES, SELECT OPERATIONS, IE, READ, WRITE, ETC. • INTERFACE WITH FILE SYSTEM DEVICE TASK
TDCINT (PR 4C215)	TCINTC	• MONITOR AND CONTROL DEVICE, SELECT OPERATIONS, IE, READ, WRITE, ETC • INTERFACES WITH FILE SYSTEM DEVICE TASK
FILDEV (PR 4C206)	OPRFNSH	• EVENT ROUTINE WITH INTERFACES BETWEEN INTERRUPT AND PROCESS LEVEL
	COMPLET	• PROCESS COMPLETION OF OPERATIONS AND PASS CONTROL TO THE TERMINAL ADMINISTRATOR TASK

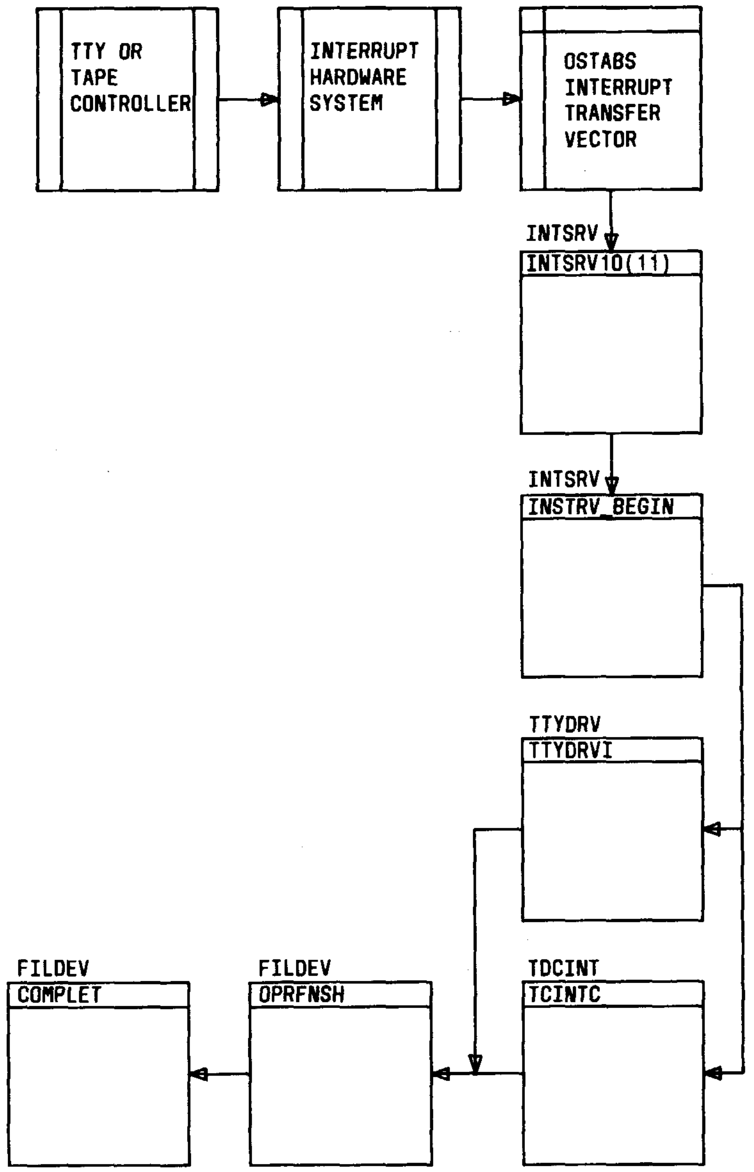

Fig. 7—Interrupt Processing, Interrupt No. 14, DMA

TABLE A

TIMER MANAGEMENT FUNCTIONS
AND
PROGRAM UNIT IDENTIFICATION

ASSEMBLY UNIT	PROGRAM UNIT	FUNCTIONS
OSTABS	TMSBRTBL	This table specifies the subroutines which are called by TIMEAU. The table consists of three EOS subroutines and up to five application subroutines.
TIMEAU	TIMEINT	Contains the code which assumes control on each timer interrupt. This routine sets up for the interrupt return and then passes control to the TIMEACT subroutine or to any application specified subroutine which executes at interrupt level. Once the routines are finished processing, TIMEINT gives control to the dispatcher.
	TIMEACT	This routine examines the first entry in the timer subroutine table to find the status of timer requests and what action is specified.
	TIMEMSG	This unit processes all timer messages except those initiated by the ACT_TIMER macro. This unit also contains the entry point for kernel routines which call service programs as subroutines. It also contains the entry point for the routine called from a process via a SVC 9 call.
	DEACT	This unit deactivates all timer entries of a given process which utilizes a specified event flag or tag.
	SETDATE	This unit sets a date into the system clock.
	MAI_TIME	This unit is called at each 10ms interrupt to determine when it is time to change the priority of the periodic maintenance task MAINT.
	SETTIME	This unit inserts a time value into the system clock and the timing list entries to maintain proper intervals.

TABLE A (Contd)

TIMER MANAGEMENT FUNCTIONS
AND
PROGRAM UNIT IDENTIFICATION

ASSEMBLY UNIT	PROGRAM UNIT	FUNCTIONS
	READTIMEC	This unit reads a system date and time into a message parameter list.
	ADDINRTNC	This unit adds an entry to the interrupt routine execute table or changes the timing of a current entry.
	DEUNRTNC	This unit removes an entry from the interrupt routine execute table.
	EXTODISP	This unit executes an exit to the dispatcher with the system state stored in the system area.
	FIL_TIME	This unit keeps a count of the 10ms interrupts for use in timing the file system. At each interrupt the top entry in the timeout queue is checked and when the count is less than equal to the current count, an event flag is set in the file system task FILDEV.