# DEVICE HANDLERS
SOFTWARE SUBSYSTEM DESCRIPTION
3A PROCESSOR, EXTENDED OPERATING SYSTEM

CONTENTS PAGE

1. GENERAL

1.01 This section contains a functional description of the device handlers available with the Extended Operating System (EOS). The device handlers together with the EOS file system provide EOS tasks, application tasks, and the craft person with a means of accessing peripheral devices. The file system provides the software for associating a local file with a physical device and for servicing requests for input/output (I/O) operations on that file. It also handles requests from the craft person concerning peripheral devices. The device handlers provide the software interface between the file system and the peripheral devices.

1.02 Whenever this section is reissued, the reason(s) for reissue will be listed in this paragraph.

A. References

1.03 The following sections contain background information that may be helpful in understanding the operations of the device handlers:

SECTION	TITLE
254-340-031	Processor/Process Management, Interrupt Handling and Timer Management, Software Subsystem

SECTION	TITLE
	Description, 3A Processor, Extended Operating System
254-340-062	File System, Software Subsystem Description, 3A Processor, Extended Operating System.

1.04 The following programs and tables provide EOS facilities used in implementing the device handlers.

(a) The operating system tables (LOSTAB), provide the various application parameters which define the system hardware and software configuration.

(b) The device handler interface program (FILDEV), PR-4C206, is the portion of the file system that provides the interface to the device handlers.

(c) The interrupt service routine (INTSRV), PR-4C113, services interrupts from file system devices and calls the device handlers.

B. Functional Applications of the Device Handlers

1.05 The device handlers receive requests for operations to be performed on peripheral devices. These requests can originate as a result of the following:

- File system I/O operations
- Manual requests from a craft person
- DEV_CONTROL macro.

File system I/O operations can originate from either EOS or application tasks. Manual requests from the craft person are channeled through the file system to the device handlers. The DEV_CONTROL macro enables the user to communicate directly with a device handler without going through the file system.

1.06 Currently, EOS supplies device handlers for the following types of devices:

- Cartridge tape unit referred to as the tape data controller (TDC)
- Teletypewriter controller (TTYC)
- Devices connected to a remote serial interface (RSI) interface
- Programmable Magnetic Tape System (PROMATS) 9-track tape drive with a direct memory access (DMA)
- PROMATS 9-track tape drive without a DMA.

There is only one device handler per device type; therefore, all devices corresponding to a given type are accessed by the same device handler.

1.07 A device handler translates a request into one or more "device commands." It then executes the command(s) and notifies the task or craft person when the request is complete. A device handler also monitors the status of the associated device(s). When major changes in this status occur, the device handler notifies the tasks that have requested to be notified of such changes.

2. FUNCTIONAL OVERVIEW

A. Functional Structure of the Device Handlers

2.01 The program elements of a device handler are invoked as subroutines. A device handler can be invoked at one of two levels: the process (or task) level and the interrupt level. The process level is entered on calls from the file system or from the DEV_CONTROL macro. The interrupt level is entered when the associated device interrupts to signal the completion of a device command.

2.02 Each device handler consists of a scheduler and a driver. The scheduler translates a request into the appropriate device commands and passes these commands to the driver for execution. The scheduler is responsible for storing the current status of a device and for notifying tasks of major changes in this status. The scheduler must also determine when a requested operation is complete or when the operation cannot be completed because of an error condition. The task that issues a request is notified via its specified completion event when the operation is successfully complete. If the operation is terminated because of an error, the task is notified via event 9.

2.03 A third element that may be considered part of a device handler is referred to as the sequencer. The sequencer provides the linkage between the scheduler and driver. The sequencer is contained in the EOS task FILDEV (see paragraph 2.14) and is common to all device handlers.

2.04 The driver executes the device commands specified by the scheduler. The driver can either wait for a command to complete before returning to the sequencer or, if the device is interrupt-driven, it can return immediately and rely on the device to generate an interrupt when the command is complete. If the driver has to wait for a command to complete, no other processing can take place while the command is executing.

2.05 The manner in which the device handlers are organized is designed to place the device-dependent functions and device-independent functions in separate routines. This separation of functions maximizes flexibility and minimizes duplication of code. It also provides a common point to intercept calls to the driver so that the load on the DMA and the system can be regulated.

2.06 The driver for an interrupt-driven device has two entry points: one for process-level calls and one for interrupt-level calls. The interrupt-level entry point is called by the interrupt service routine when the device interrupts. The process-level entry point is called by the sequencer. To prevent a device handler from concurrently receiving requests at both the process level and interrupt level, the interrupts for a device are blocked when the device handler is entered and unblocked after it is exited.

2.07 At the process level, the device control block is referenced by the sequencer to determine which device handler to call to process a request. At the interrupt level, the device handler is called by the interrupt service routine. If the interrupt level has an acknowledge interrupt (ACKI) capability, the interrupt service routine can determine which device interrupted. The device handler of the interrupting device is then called. If there is no ACKI function available, all device handlers of devices on that interrupt level are called.

2.08 The EOS uses the following tables to define the device handlers and associated interrupts:

(a) Device driver (DEV_DRV) table—This table is created in LOSTAB during system generation and defines each device handler. Each device handler in the table is assigned a name. This name is also placed in the device equipment (DEV_EQP) table to cross-reference the device to the associated device handler. The DEV_DRV table also defines the interrupt level (if applicable) and process level entry points of the device driver and the entry point of the scheduler of each device handler.

(b) Interrupt equipment (INT_EQP) table—This table is built during system initialization and specifies which devices are connected to each interrupt level.

B. Device Handler Communication

2.09 The following control blocks are used in providing communication between the device handlers and FILDEV and between the schedulers and drivers within the device handlers:

(a) Device control block (DVCB)—Describes a device to the associated device handler and contains the information required to communicate with the device. A DVCB is built for each device during system initialization. The contents of the DVCB is shown in Table A.

(b) File control block (FCB)—An FCB is associated with each opened file and is used to pass information about the file between the file system and a device handler. The contents of the FCB is shown in Table B.

(c) Input/output control block (IOCB)—Provides communication between the scheduler and driver of a device handler. The contents of the IOCB is shown in Table C.

2.10 The DVCB_ACTN field in the DVCB is one of the inputs to a device handler. Following is a list of the possible inputs for this field. The number in parentheses is the actual value of the associated symbol.

- DEVICE_OPR (1)—Perform the operation specified in the DVCB_OPR field

- FILE_OPR (2)—Perform the operation specified in the FCB
- RESPONSE_NORMAL (3)—The previous device command was completed without error
- RESPONSE_FAIL (4)—An error was encountered with the previous device command
- TIMEOUT (5)—The timer activated by the device handler has expired
- SYSTEM_INIT (6)—A processor initialization has occurred
- MANUAL_REQ (7)—The craft person submitted a manual request.

2.11 When the input for the DVCB_ACTN field is DEVICE_OPR, there is no file associated with the requested operation. In this case, the DVCB_OPR field specifies the operation to be performed by the device handler. Currently, the only possible inputs for this field are as follows:

- INITS—OPR (0)—Initialize the device
- MAN_MODE (13)—Put device in the manual mode.

2.12 When the input for the DVCB_ACTN field is FILE_OPR, the requested operation is associated with a file. In this case, the DVCB_OPR field contains the address of the FCB associated with the file and the FCB_FOPR field in the FCB specifies the operation. Following is a list of possible inputs for this field:

- READ_OPR (1)—Read
- WRIT_OPR (2)—Write
- WEOF_OPR (4)—Write end of file
- REWIND_OPR (5)—Rewind
- SKIP_REC_OPR (7)—Skip record
- SKIP_FM_OPR (8)—Skip to file mark
- BACKSP_REC_OPR (9)—Backspace a record
- BACKSP_FM_OPR (10)—Backspace to file mark
- RECORD_OPR (11)—Skip to record
- RD_STATUS_OPR (12)—Read device status
- CLOSE_OPR (14)—Close operation on file.

2.13 The DEV_CONTROL macro enables a user to bypass the file system and communicate directly with a device handler. Using this macro, the user can initialize any device, retrieve the error count (see paragraph 2.19) for a device and reset the error count to zero. When a request has been received by a device handler to place a device in the manual state, any user that has requested to be notified of changes in device status can deny or allow the request using this macro.

C. Device Handler Interface

2.14 The EOS task FILDEV provides the interface between the device handlers and the file system. All calls to the device handlers from the file system are made by FILDEV. File system calls can result from EOS or user-requested file operations or manual requests from the craft person. In any case, FILDEV checks the request for errors and gathers data needed in processing the request and then passes the request on to the appropriate device handler.

2.15 The FILDEV task also provides the interface between the interrupt level and process level of a device handler. On an interrupt-level call, a device handler receives control at the interrupt-level entry point of the driver. When the interrupt-level processing is complete, the driver transfers control to FILDEV to invoke the process level functions of the device handler.

2.16 At the process level, FILDEV provides the linkage between the scheduler and driver of a device handler. It does this through the sequencer which coordinates the sequencing of calls to the scheduler and driver as directed by the DVCB_ACTN field. Normally, this field is not zero on entry to FILDEV (see paragraph 2.10). However, in the case where the interrupt-level routine of the driver has requested a return to the process level, it may have deferred some work to the process level to be done before the scheduler is called. Therefore, the DVCB_ACTN field is checked before calling the scheduler. If the field is zero, there is no request to send to the scheduler; therefore, the process-level routine of the driver is called to fill in the field. If the field is still zero on return from the driver, the device handler is exited. If the DVCB_ACTN field is not zero on entry to FILDEV or on return from the driver, the scheduler is called. The scheduler returns to FILDEV with an indication of whether the driver should be called to execute a device command. If a command is to be executed, the process-level routine of the driver is called. Otherwise, the device handler is exited.

D. Device Status

2.17 Devices controlled by EOS can be in one of five states:

- Available
- Unavailable
- Faulty
- Manual
- Unknown.

The available state signifies that the device is ready for use. All other states indicate that the device is not ready for normal use. A device handler places a device in the unavailable state when it detects a condition that prevents the device from transferring data. If a device is experiencing problems in transferring data but there seems to be no problem in communicating with the device, the device is placed in the faulty state. A device is placed in the unknown state during a system initialization with a level of LMSG or greater. The craft person can place a device in the manual state by entering a TTY message. This state prevents the device from being used by the file system and permits diagnostics to be run on the device or permits the craft person to use the device.

2.18 The craft person can request the current state of a device using a TTY message. The DEV_STATUS macro enables a task to request the current status of a device. It also enables a task to request to be notified whenever the status of a device changes to available, unavailable, faulty, or manual. An option is available with the DEVICE macro in LOSTAB which specifies whether the controlling task (ie, a task that has issued the DEV_STATUS macro) wants to be notified when the craft person places a device in the manual state. If a task is to be notified, then the device is first placed in an intermediate state known as the remove state. If no action is taken by the task monitoring the device within 7 seconds, the device is then placed in the manual state. If the task finds that it needs the device before the 7 seconds are up, it can deny the request using the DEV—CONTROL macro which will cause the device to be restored to its original state.

E. Error Handling

2.19 Error conditions encountered by the device handlers can result from erroneous requests from the file system or device failures. Errors are reported to the craft person via TTY messages and the user via designated event routines. All device handlers keep a count of the number of recoverable and nonrecoverable errors they have encountered. This data can be obtained either by entering the REPT:STAT TTY message or by issuing a DEV_CONTROL macro call.

2.20 If a device handler receives an invalid operation with a FILE_OPR request, the FCB_IVIO flag in the FCB is set and the requested operation is terminated. If an invalid operation is received with a DEVICE_OPR request, the operation is ignored.

2.21 If the possibility exists that a device can enter a condition in which it will never respond to a device command, those commands must be timed. The timing mechanism that performs this function is controlled through FILDEV. A subroutine in FILDEV can be called to activate the timer. The duration of the timer can be specified in 1 millisecond intervals. If a command exceeds the time limit specified, a time-out occurs and an event routine in FILDEV receives control.

3. DEVICE HANDLER FLOW OF CONTROL

3.01 Table D lists the device handlers currently provided by the EOS. The table defines each device handler in terms of the associated device type, the assembly unit names of the scheduler and driver and their associated entry points. All of the device handlers have the same basic structure. Also, the flow of control through the device handlers and the functions performed by their corresponding program elements are basically the same. Because of these similarities, this section will not describe any particular device handler. Instead, a description is provided of the basic interaction sequence that occurs between the different program elements of a device handler and FILDEV when processing a file system request. A flow diagram of this control sequence is shown in Fig. 1 for the process level and in Fig. 2 for the interrupt level.

A. Process-Level Flow of Control

3.02 The task FILDEV receives control at event routine 3 for manual requests from the craft person and at event routine 6 for I/O requests from processes and tasks. In either case, FILDEV checks the request for errors, blocks interrupts to the device, and calls the sequencer. The sequencer (DEVITF) is a program element within FILDEV which is invoked as a subroutine.

3.03 The first function performed by the sequencer when called by either event routine 3 or 6 is to call the scheduler. The scheduler determines what device operations, if any, need to be performed by the driver to satisfy the request. The scheduler returns to the sequencer with an indication of whether the driver needs to be called. This indication is provided using the condition flip-flop (CF). If the CF is set to "0", then the scheduler has no work for the driver and the sequencer returns to the event routine from which it was invoked. If the CF is set to "1", then the scheduler has a device command to be executed.

3.04 When a command is to be executed, the scheduler builds an IOCB (Table C) and stores the address of the IOCB in the DBCB_IOCB field in the DVCB. The IOCB contains the command to be executed and a flag indicating whether the DMA is needed in processing the command. Normally, the commands will correspond to the device operations defined in paragraph 2.12.

3.05 If on return from the driver the DVCB_ACTN field is zero, the device is waiting for a response and the sequencer returns to the controlling event routine. Receipt of the response will be signaled by an interrupt. The interrupt-level flow of control is described in Part 3B. If the DVCB_ACTN field is not zero, the device has generated a response for the scheduler. This means either the command was successful or an error was encountered. The sequencer then deactivates any timers (see paragraph 2.21) and calls the scheduler.

3.06 When a requested file operation is complete or an error is encountered, before returning to the sequencer, the scheduler calls a subroutine in FILDEV to set an appropriate event flag in the task that originated the request (see paragraph 2.02). Upon completion of a manual request that caused the status of a device to change, the subroutine IOSTATCG in FILDEV is also called to notify the craft person and any task that has requested to be notified, of the change in status. The scheduler then returns to the sequencer with the CF set to "0". This indicates there is no more work to be done and the sequencer returns to the event routine in FILDEV from which it was invoked.

B. Interrupt-Level Flow of Control

3.07 The driver for an interrupt-driven device has the option of waiting for a device command to complete before returning to the sequencer or returning immediately and relying on the device to generate an interrupt when the command is complete. If the interrupt option is not used, the processing sequence conforms entirely with that described in Part 3A. Otherwise, processing follows the sequence described in paragraphs 3.08 and 3.09 when a device interrupts.

3.08 When a device generates an interrupt, the interrupt service routine (INTSRV) passes control to the interrupt-level entry point of the driver that issued the command. When the driver completes the interrupt processing, it sets event flag 4 in FILDEV to effect a return to the process level of the device handler. When event routine 4 in FILDEV receives control, it deactivates any timers in progress for the device, frees the DMA if it is being used by the device, blocks interrupts, and calls the sequencer.

3.09 Normally, the interrupt-level routine of the driver sets the DVCB_ACTN field in the DVCB to either "RESPONSE_NORMAL" or "RESPONSE_FAIL." However, the field is set to zero when some additional work is being deferred to the process level that must be completed before the scheduler is called. Therefore, on entry to the sequencer from event routine 4, the DVCB_ACTN field is checked for zero. If the field is not zero, the scheduler is called. If the field is zero, the process-level routine of the driver is called. If the field is still zero on return from the driver, the driver is again waiting for a response from the device and the device handler is exited. Otherwise, the scheduler is called.

C. Time-Out Processing

3.10 Event routine 14 of FILDEV receives control when a time-out occurs for a file system device (see Fig. 3). Two options are available to process a time-out. Either the interrupt-level routine of the driver is called or the scheduler is called with the DVCB_ACTN field in the DVCB equal to "TIMEOUT." The option is selected when the timer is activated. The ACTTIMER subroutine in FILDEV is called by the driver to activate a timer for a device command. The option is selected through a flag passed to ACTTIMER.

3.11 The interrupt-level routine of the driver is called when the time-out is to be processed as a pseudo interrupt. Pseudo interrupts are used for such purposes as controlling the timing involved in positioning an I/O device. The scheduler is called when the time-out is being used to detect a possible error condition (see paragraph 2.21). When the scheduler is called, either the device command is retried or the operation is terminated with an error return to the task or process that initiated the operation.

4. GLOSSARY

4.01 The following basic terms are defined in the context of their use in this section:

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

Interrupt—A hardware generated signal that causes the program unit currently executing to break and another function to be performed that is required immediately.

LMSG—The initialization level at which system dynamic memory is zeroed. It is the level at which EOS takes the most drastic action without causing a bootstrap.

Macro—A precoded sequence of instructions to which a label has been assigned. This label along with any parameters can be used as an instruction in a source language. On each call, the macro instruction is replaced by its equivalent instruction sequence.

Subroutine—A sequence of instructions to perform a particular function which is common to several programs.

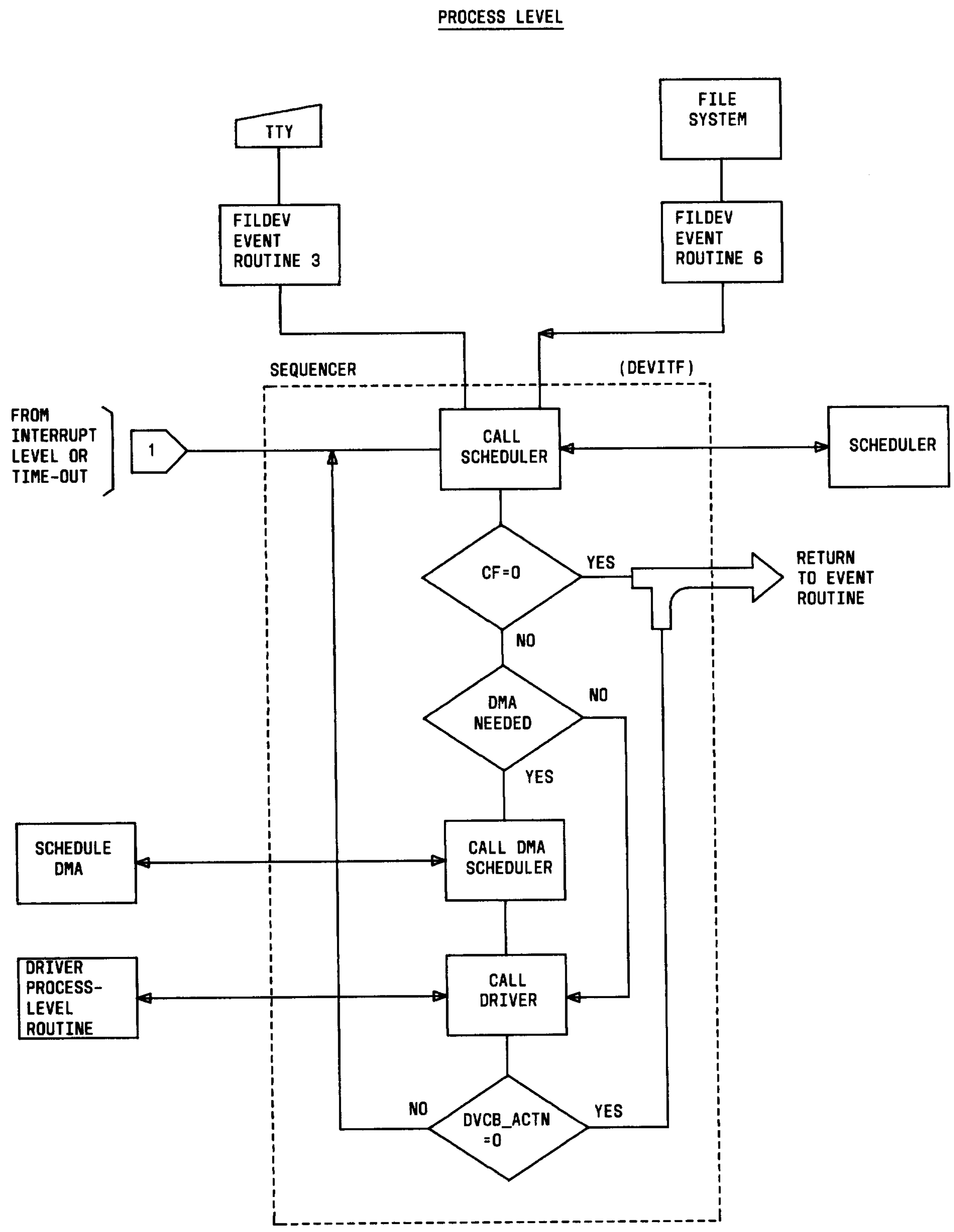

Fig. 1—Device Handler Process Level Control Flow

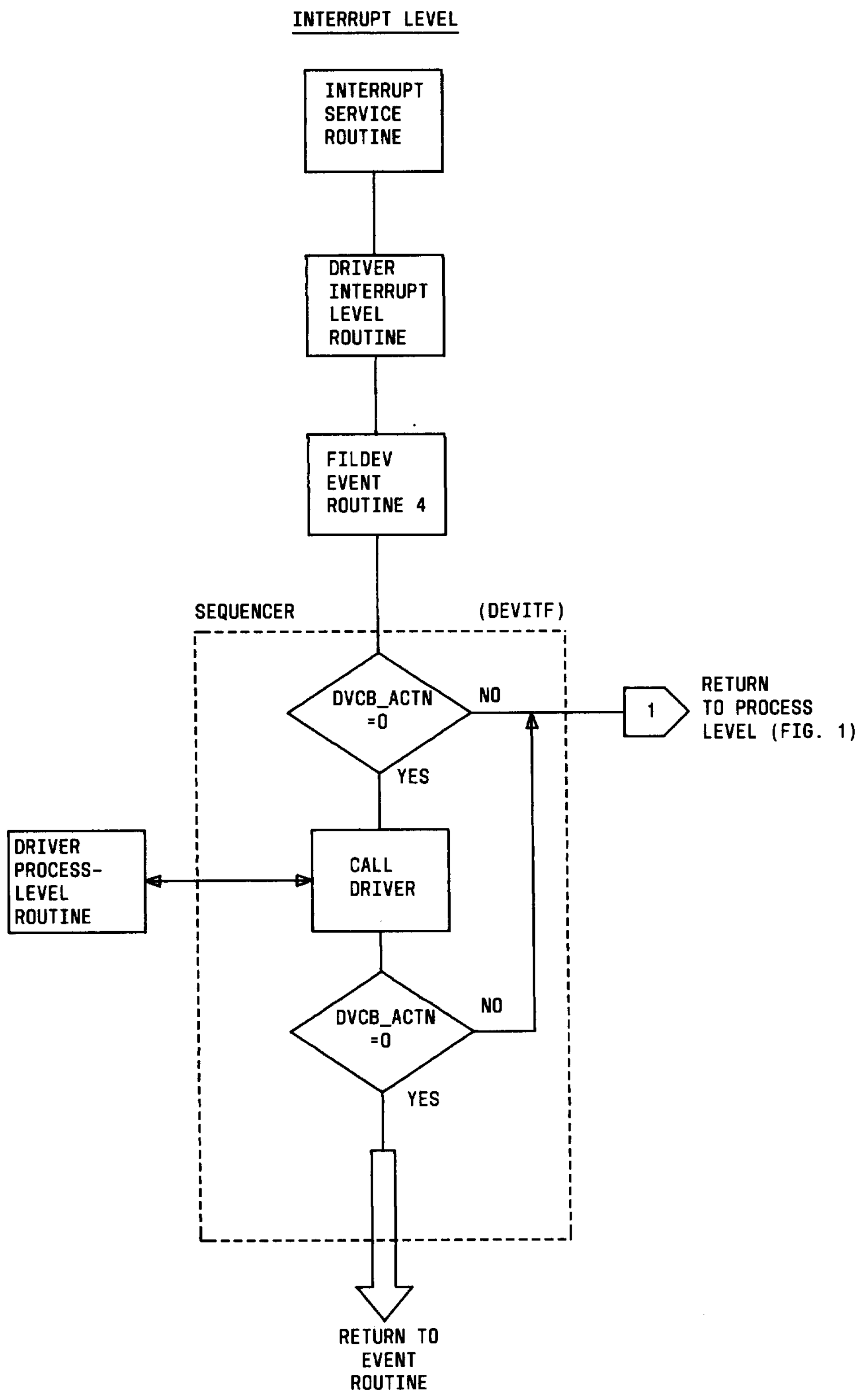

Fig. 2—Device Handler Interrupt Level Control Flow

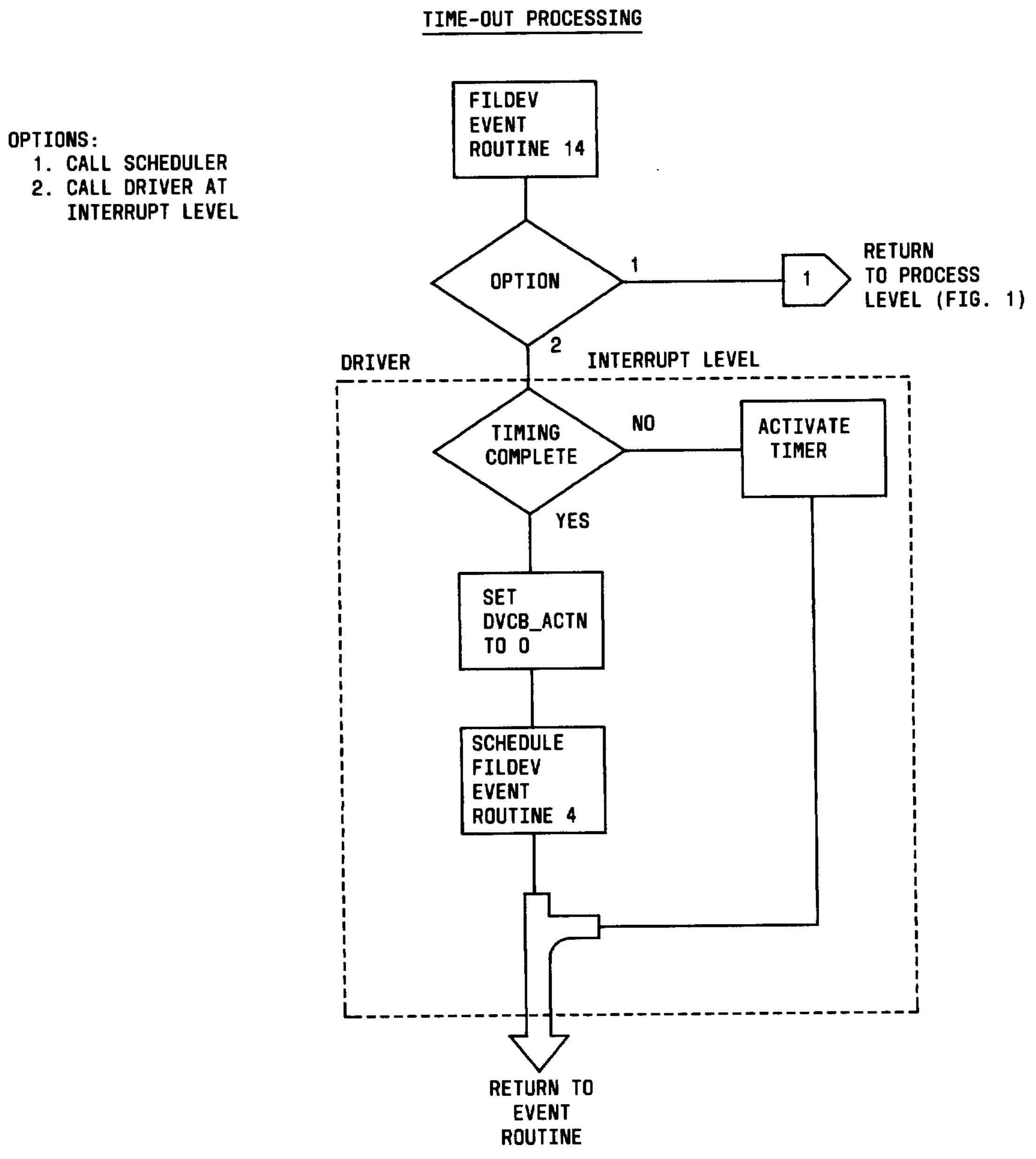

Fig. 3—Device Time-Out Processing

TABLE A

CONTENTS OF DEVICE CONTROL BLOCK

FIELD (NOTE)		FIELD DESCRIPTION
DVCB_HEADER	(FS)	System Table Header
DVCB_TYP	(FS)	Device Type
DVCB_DRV	(FS)	Device Handler Index Into DEV_DRV
DVCB_CTYP	(FS)	Type of Channel Device is Connected to
DVCB_STAT	(FS)	Status Flags
DVCB_OPTN	(FS)	Device Option Flags
DVCB_RTCD	(DH)	Return Code Flags
DVCB_CSBA	(FS)	Physical Address of Device
DVCB_DEVA	(FS)	Device Field of Physical Address
DVCB_FLAGS	(DH)	Flags Available to Device Handlers
DVCB_DMAA	(FS)	DMA Address of Device
DVCB_DVEQ	(FS)	DVEQ Table Index
DVCB_HEAD	(FS-DH)	Chain Start of FCBS Opened on This Device (Device Handlers May Change Order of the Link List)
DVCB_BUFA	(FS)	Address of Buffer Assigned to Device
DVCB_BUFS	(FS)	Length of Buffer (in Words)
DVCB_IOCB	(DH)	Address of IO Control Block
DVCB_ACTN	(FS-DH)	'Action Request' Used as Input to State Routine (Scheduler)
DVCB_STATE	(DH)	Current State of Device
DVCB_OPR	(FS)	New Operation Code (or FCB Address)
FOLLOWING FIELDS USED FOR COMMUNICATING DEVICE STATUS TO THE FILE SYSTEM AND CRAFTSPERSON		
DVCB_MPTY	(DH)	Message Priority
DVCB_APTY	(DH)	Message Action Priority
DVCB_FSDS	(DH)	Message Keyword (Device Status)
DVCB_SWD1	(DH)	Message Status Word 1
DVCB_SWD2	(DH)	Message Status Word 2
DVCB_SWD3	(DH)	Message Status Word 3
DVCB_SWD4	(DH)	Message Status Word 4
DVCB_SPARE	(DH)	Meaning Assigned by DH

Note: FS or DH indicates whether the file system (FS) or the device handler (DH) can modify the field.

TABLE B

CONTENTS OF FILE CONTROL BLOCK

FIELD (NOTE)		FIELD DESCRIPTION
FCB_HEADER	(FS)	System Table Header
FCB_UFBA	(FS)	Address of Files UFB
FCB_ACMD	(FS)	Type of Access Method Employed
FCB_LFTA	(FS)	Local File Table Address
FCB_DVCB	(FS)	Address of DVCB
FCB_STAT	(FS)	Status and Operation Flags
FCB_RTCD	(DH)	Return Codes for File System Operations
FCB_SDNM	(FS)	Sending Processes (or Tasks) Name
FCB_BUFL	(FS)	Length of Users Buffer (in Bytes)
FCB_BUFA	(FS)	Address of Users Buffer
FCB_FOPR	(FS)	Current File System Operation on File
FCB_INDX	(FS)	Index for File Operation
FCB_ACTN	(DH)	State Routine Input
FCB_IOST	(DH)	State of File
FCB_PRTY	(DH)	Priority of Operation
FCB_BLKS	(FS)	Block Size of File
FCB_RCNO	(DH)	Current Record Number of File
FOLLOWING FIELDS ARE LAID OUT AS AN IOCB		
FCB_MEMA	(DH)	Memory Address
FCB_DCMD	(DH)	Device Command
FCB_DMAR	(DH)	DMA Required Flag
FCB_DEVA	(DH)	Device Address
FCB_TRNL	(DH)	Transfer Length
FOLLOWING ARE SUBFIELDS IN THE FCB_RTCD FIELD AND ARE USED AS RETURN CODES		
FCB_DER	(FS-DH)	Device Error (Device in UAV or Man State)
FCB_IOER	(FS-DH)	IO Error
FCB_BRKR	(FS-DH)	Break (or Attention) Received
FCB_RCTR	(FS-DH)	Record Truncated on Read
FCB_EOD	(FS-DH)	End of Data
FCB_IVIO	(FS-DH)	Invalid Operation for Device

Note: FS or DH indicates whether the file system (FS) or the device handler (DH) can modify the field.

TABLE C

CONTENTS OF I/O CONTROL BLOCK

FIELD (NOTE)	FIELD DESCRIPTION
MEMA (DH)	Memory Address
DCMD (DH)	Device Command
DMAR (DH)	DMA Request Flag
DVAD (DH)	Device Address
TRNL (DH)	Transfer Length

Note: FS or DH indicates whether the file system (FS) or the device handler (DH) can modify the field.

TABLE D

EOS DEVICE HANDLERS

DEVICE TYPE	SCHEDULER		DRIVER		
	PIDENT	ENTRY POINT	PIDENT	PROCESS LEVEL ENTRY POINT	INTERRUPT LEVEL ENTRY POINT
TDC	TDCSTA	TCSTATC	TDCHND	TCDRVC	
			TDCINT		TCINTC
TTY	TTSTAT	TTSTAT	TTYDRV	TTYDRV	TTYDRVI
PROMATS (DMA)	TAPTSK	TAPTSKC	JHPROD	JPROD	ITPHDL
PROMATS	TAPTSK	TAPTSKC	PCHMAT	PROME	-NA-
RSI	RSIDRV	RSISTA	RSIDRV	RSIDRA	RSIILA