HEWLETT PACKARD

RTE Operating System

Driver Writing Manual

Software Technology Division
11000 Wolfe Road
Cupertino, CA 95014-9804

Manual Part No. 92200-93005
E0693

Printed in U.S.A. June 1993
Eighth Edition

NOTICE

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THE MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

RESTRICTED RIGHTS LEGEND

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARs 252.227.7013.

Printing History

The Printing History below identifies the edition of this manual and any updates that are included. Periodically, update packages are distributed that contain replacement pages to be merged into the manual, including an updated copy of this printing history page. Also, the update may contain write-in instructions.

Each reprinting of this manual will incorporate all past updates; however, no new information will be added. Thus, the reprinted copy will be identical in content to prior printings of the same edition with its user-inserted update information. New editions of this manual will contain new information, as well as all updates.

To determine which manual edition and update is compatible with your current software revision code, refer to the Manual Numbering File or the Computer User's Documentation Index. (The Manual Numbering File is included with your software. It consists of an "M" followed by a five digit product number.)

Fifth Edition	Feb 1980	
Update 1	Jul 1981	Manual Enhancement
Reprint	Jul 1981	Update 1 incorporated
Sixth Edition	Dec 1983	RTE-6/VM compatibility
Seventh Edition	Aug 1987	Rev. 5000 (Software Update 5.0)
Eighth Edition	Jun 1993	Rev. 6000 (Software Update 6.0)

Documentation Map

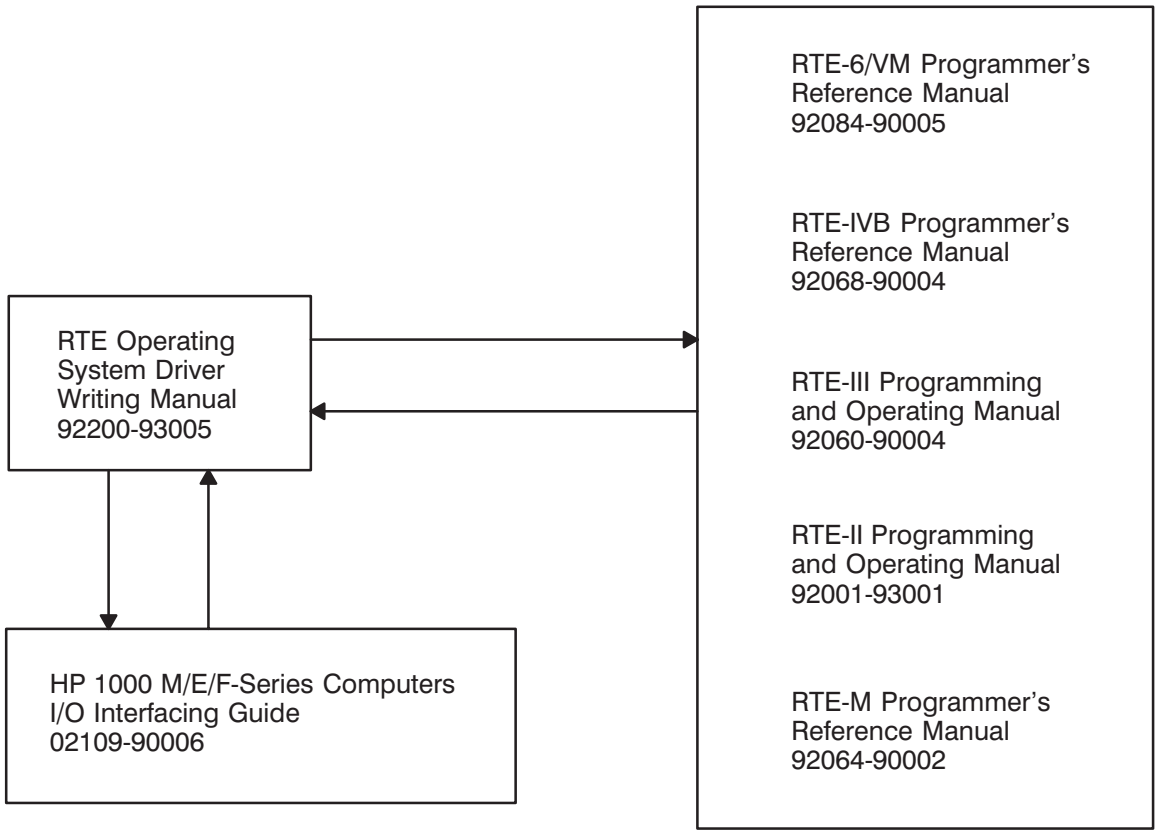

Table of Contents

Chapter 4
Writing Privileged RTE Drivers

List of Illustrations

Tables

1

Introduction

Purpose

The *RTE Operating System Driver Writing Manual* is a reference for those users who wish to develop their own device drivers. A device driver provides the software interface between a peripheral device and the RTE operating system. Many drivers for Hewlett-Packard peripherals have already been written and are available from HP. Users who wish to interface peripherals that are not supported by HP will require specialized drivers. The information in this manual will aid the user in the development of such routines.

Note that it is not the purpose of the manual to describe the various HP-supplied drivers in detail. Each of these is described in a separate manual specific to the driver.

Scope

The manual first provides a general description of the input/output (I/O) characteristics of the RTE family of operating systems on M/E/F-Series Computers. The techniques and requirements for developing device drivers are then presented in subsequent sections.

Because all of the RTE operating systems have the same general I/O structure, the manual can be used to develop general purpose drivers for use in any of these RTE systems. There are some areas where differences between operating systems may affect driver structure and operation, these areas are clearly pointed out in the text with notations such as “RTE-IV only” or “RTE-III only”. Phrases such as “RTE-III only” should be interpreted as referring to both RTE-III (disk-based system) and RTE-M/III (memory-based equivalent of RTE-III).

Supporting Documentation

To use this manual effectively, you should be thoroughly familiar with HP Assembly Language and with the Programming and Operating Manual for the RTE system in which the driver is to be used. Refer to the Documentation Map at the front of this manual for information on these and other available manuals. For specific information on an HP supplied driver, refer to the appropriate driver manual.

2

RTE Input/Output Structure

Introduction

In RTE, centralized control and logical referencing of input and output (I/O) operations effect simple device-independent programming. By means of several user-defined I/O tables, I/O drivers, and program EXEC calls, the programmer is relieved of most I/O problems. To understand the software I/O characteristics of RTE, you should be familiar with two hardware related terms used in this manual:

I/O Controller	A combination of I/O card, cable, and (for some devices) controller box used to control one or more I/O devices on a computer I/O select code.
I/O Device	A physical unit (or portion of a unit) identified in the RTE operating system by means of an Equipment Table entry and a subchannel assignment.

Each I/O device is interfaced to the computer through an I/O controller. This controller is associated with one or more of the computer I/O select codes. Interrupts from controllers on specific select codes are directed to specific memory locations in the computer for system processing.

It is also important to note the difference between a synchronous device and a non-synchronous device. An interrupt from a synchronous device controller must be processed within a specified time period, or the data will be lost. Examples of synchronous devices are moving-head disk drives and nine-track magnetic tape drives. Non-synchronous devices have no such requirement, and interrupts from these device controllers can be serviced whenever the computer is able to do so. Examples of non-synchronous devices include paper tape punches and readers.

Software I/O Structure

The RTE I/O structure is made up of two general types of software (the system I/O processor and the various device drivers) and a number of I/O tables and a communications area (the Equipment Table, the Device Reference Table, the Interrupt Table, the Driver Mapping Table (RTE-IV and RTE-6/VM only), and the Base Page Communications Area). These tables and areas are used for communication between the system and the drivers, and for control of the many I/O operations that can be in progress simultaneously. Each component of the I/O structure is discussed individually in this subsection. A summary of the overall I/O process is given in the next subsection.

Input/Output Device Drivers

Input/Output device drivers provide the software interface between peripheral I/O devices and the operating system. Drivers are responsible for the initiation, continuation, and completion of all data transfers between an I/O device and the computer. Drivers communicate with the system directly via parameter passing, and indirectly through the various tables and communications areas (particularly the Equipment Table and the Base Page Communications Area) that are discussed later in this subsection. There are two types of drivers, standard and privileged. Standard drivers are simpler and can be used for most asynchronous devices and some high-speed and synchronous devices (if DCPC transfers are used); these drivers are discussed in Chapter 3. Privileged drivers are more complex and are generally used for high-speed and synchronous devices that require driver interaction on each data word transferred (that is, DCPC transfers cannot be used); these drivers are discussed in Chapter 4.

System I/O Processor

The system I/O processor provides the software interface between user programs that perform I/O and the drivers that actually handle the I/O operations. The system I/O processor checks user I/O calls for validity, suspends programs while their I/O is in progress (if necessary), calls drivers to initiate the I/O data transfers, directs controller interrupts to the appropriate drivers, and restarts programs suspended for I/O. The mechanism for communication between the system I/O processor and user programs is the EXEC call and its associated parameter. Communication between the system I/O processor and drivers is handled directly via parameters and indirectly through the various I/O tables discussed in this section.

Two general areas within the system I/O processor are discussed in this manual, IOC and CIC. The Input/Output Control module (IOC) is entered when a user program makes an I/O request. IOC is responsible for initiating the I/O transfer by calling the appropriate driver. The Central Interrupt Control module (CIC) is entered when a device controller interrupt is detected. CIC is responsible for calling the correct driver to handle the interrupt.

Base Page Communications Area

A block of storage in base page contains the system's communications area and is used by RTE to define request parameters, I/O tables, scheduling lists, operating parameters, memory bounds, and so forth. The RTE Assembler and Macroassembler allow absolute references to addresses less than octal 2000 so that user programs can read information from the base page. Programs cannot alter the base page, however, because of the memory protect feature of RTE. Table 2-1 illustrates the portion of the Base Page Communications Area that pertains to I/O operations. The meaning and use of the various words illustrated in the table will become clear in subsequent sections of this manual. For a complete description of the Base Page Communications Area, refer to the appropriate RTE System Programming and Operating Manual.

Table 2-1. Base Page Communications Area – I/O Operations

Octal Location	Contents	Description
. . .		
01650	EQTA	Address of Equipment Table (EQT)
01651	EQT#	Number of EQT entries
01652	DRT	Address of Device Reference Word 1 Table
01653	LUMAX	Number of logical units (in Device Reference Table)
01654	INTBA	Address of Interrupt Table
01655	INTLG	Number of Interrupt Table entries
01656	TAT	Address of Track Assignment Table (disk-based systems only)
01657	KEYWD	Address of keyword block
01660 01661 01662 01663 01664 01665 01666 01667 01670 01671 01672	EQT1 EQT2 EQT3 EQT4 EQT5 EQT6 EQT7 EQT8 EQT9 EQT10 EQT11	Addresses of first 11 words of current EQT entry (see location of 01771 for last four words)
01673	CHAN	Current DCPC Select Code (6 or 7)
01717	XEQT	ID segment address of current program
. . .		
01737	DUMMY	I/O channel of privileged interrupt card (0 if none)
. . .		
01770	MPTFL	Memory Protect On/Off (0/1) flag
01771 01772 01773 01774	EQT12 EQT13 EQT14 EQT15	Addresses of last 4 words of current EQT

Equipment Table

The Equipment Table (EQT) is used to maintain a list of all the I/O equipment in the system. This table consists of a number of EQT entries, with one EQT entry for each I/O controller defined in the system at generation time. The EQT entry contains all of the information required by the system and the associated driver to operate the equipment, including: the I/O select code in which the controller is interfaced to the computer, the driver type, and the various requirements and specifications of the controller or driver (for example, DCPC, buffering, timeout, powerfail, and so forth). To distinguish between multiple I/O devices connected to a single controller, the system also inserts the subchannel number of the device being referenced into the EQT entry before calling the driver.

The format of each EQT entry is illustrated in Figure 2-1. Some information in the EQT entry is static, other parts are dynamic. Information marked <A> is fixed at generation time (or, for the I/O select code number in RTE-IV and RTE-6/VM, at reconfiguration time) and never changes during online operation of the system. Words marked <B> are also fixed at generation time (or, for RTE-IV and RTE-6/VM, at reconfiguration time), but can be changed online via operator commands. Information marked <C> is modified or set up for the driver prior to each I/O initialization; it informs the driver of the nature of the request. Words marked <D> are not used by the system and are therefore available to the driver for use as temporary storage for the duration of each I/O request.

EQT words 9 and 10 are available for use as temporary storage unless optional parameters were specified in the EXEC call. For the case of an EXEC call with the control word's Z-bit clear, EQT words 9 and 10 contain the values of the optional parameters. If the Z-bit was set, however, EQT word 9 contains the address of the optional buffer while word 10 contains the length of the buffer.

If the number of words marked <D> does not provide sufficient temporary storage for the driver, additional space can be allocated at generation time by specifying that an EQT entry extension is needed for a particular EQT entry. This space can only be used to extend the referenced EQT entry and, therefore, should only be allocated for drivers that need the additional space. When an EQT entry extension is specified, EQT entry words 12 and 13 are used to identify the location and length of the extension (because the extension does not immediately follow the EQT entry) and, therefore, should not be modified by the driver. Otherwise, these words are available as temporary storage. EQT word 12 should be checked to ensure that sufficient words were provided in the generation. Once this is done, the driver can modify word 12 because the system does not use it.

For programming convenience, the addresses of the words in the current EQT entry (except for words in the extension, if an extension exists) are placed in the Base Page Communications Area by the system before calling the driver to initiate or continue an I/O operation. In RTE-6/VM, if the first word is set up correctly, it is assumed that the others are correct and are not reset. A driver should use these addresses instead of computing them from the EQT entry number and the start of the Equipment Table. In this way, the driver can remain independent of the actual organization of the Equipment Table in memory.

All Equipment Table entries are located sequentially in memory beginning with EQT entry number 1. The address of the first entry and the total number of entries in the table can be found in the Base Page Communications Area.

Word	Contents															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	R	I/O Request List Pointer <C>														
2	R	Driver Initiation Section Address <A>														
3	<F> A	Driver Continuation/Completion Section Address <A>														
4	D <A>	B <B>	P <E>	S <E>	T <C>	Lower 5 bits of Subchannel # <C>				I/O Select Code # <A>						
5	AV <F>	Equipment Type Code <A>							Status <E>							
6	CONWD (Current I/O Request Word) <C>													MSB <C>		
7	Request Buffer Address <C>															
8	Request Buffer Length <C>															
9	Temporary Storage <D> or Optional Parameter <C>															
10	Temporary Storage <D> or Optional Parameter <C>															
11	Temporary Storage for Driver <D>															
12	Temporary Storage for Driver <D> or EQT Extension Size, if any <A>															
13	Temporary Storage for Driver <D> or EQT Extension Starting Address, if any <A>															
14	Device Timeout Reset Value <B>															
15	Device Timeout Clock <C>															

Where the letters in brackets (<>) indicate the nature of each data item as follows:

- <A> = fixed at generation time (or, for I/O select code # in RTE-IV, at reconfiguration time); never changes.
- <B> = fixed at generation time; can be changed online.
- <C> = set up or modified at each I/O initialization.
- <D> = available for use as temporary storage by driver.
- <E> = can be set by driver.
- <F> = maintained by system.

Figure 2-1. Equipment Table Entry Format (Sheet 1 of 2)

And where:

R	=	reserved for system use.
I/O Request List Pointer	=	pointer to list of requests queued up on this EQT entry. First entry in list is current request in progress; zero if no requests.
A	=	1 if DCPC was allocated dynamically at request of driver continuation (RTE-IVB only).
D	=	1 if DCPC required.
B	=	1 if automatic output buffering used.
P	=	1 if driver is to process powerfail.
S	=	1 if driver is to process timeout.
T	=	1 if device timed out (system sets to zero before each I/O request)
Subchannel #	=	last subchannel addressed (lower 5 bits).
MSB	=	most significant bit of the subchannel (bit 6).
I/O Select Code #	=	I/O select code for the I/O controller (lower number if a multi-board interface).
AV	=	I/O controller availability indicator: 0 = available for use. 1 = disable (down) 2 = busy (currently in operation). 3 = waiting for an available DCPC channel.
Equipment Type Code	=	in general, indicates type of device on this controller. When this octal number is linked with "DVy," it identifies the device's software driver routine. Some standard driver numbers are: 00 to 07 = paper tape devices or consoles 00 = teleprinter or keyboard control device 01 = photoreader 02 = paper tape punch 05 = HP 26xx-series terminals 07 = multi-point devices 10 to 17 = unit record devices 10 = plotter 11 = card reader 12 = line printer 15 = mark sense card reader 20 to 37 = magneic tape/mass storage devices 23 = 9-track magnetic tape (800/1600 BPI) 31 = HP 7900 moving head disk 32 = HP 7905/06/20/25 moving head disk drive, 33 = flexible disk drives, CS/80 moving head disk drive, or cartridge tape drive 36 = writable control store 37 = HP-IB 40 to 77 = instruments
Status	=	actual physical status or simulated status at the end of each operation.
CONWD	=	combination of user control word and user request code word in the I/O EXEC call (see EQT word 6).

Figure 2-1. Equipment Table Entry Format (Sheet 2 of 2)

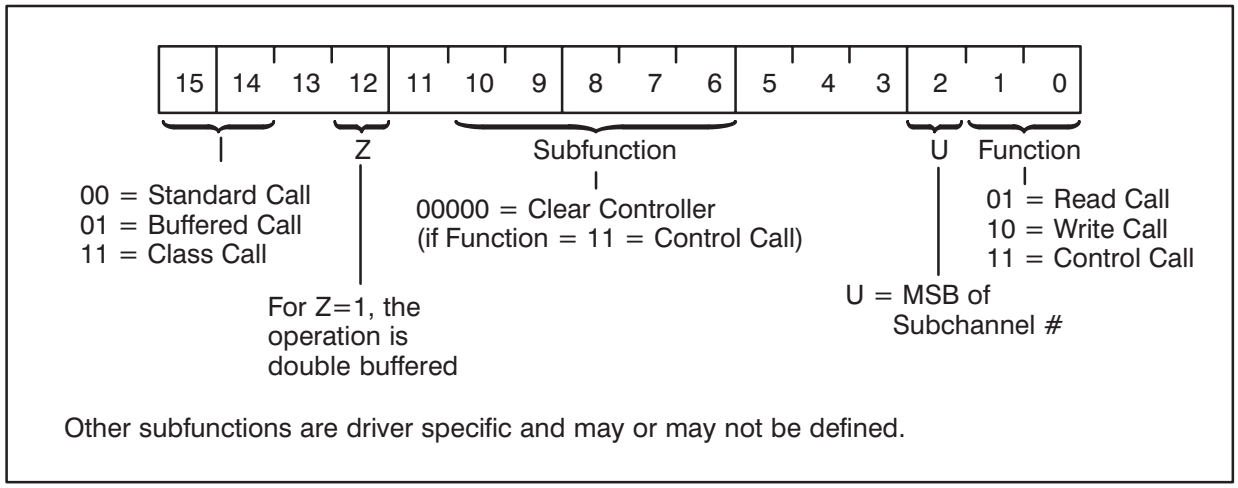

Figure 2-2. Expansion of CONWD Word (EQT Entry Word 6)

Logical Unit Numbers

Logical unit numbers (LUs) provide the RTE user with the capability of logically addressing the physical devices defined by the Equipment Table. LU numbers are maintained by the Device Reference Table (see below), and their definition can be changed online by the LU operator request. This scheme allows the programmer to reference changeable logical units instead of fixed physical units.

The functions of LUs 0 through 6 are predefined in the RTE system as follows:

- 0 – "bit bucket" (null device, no entry in Device Reference Table)
- 1 – system console
- 2 – reserved for system (system disk subchannel in disk-based systems)
- 3 – reserved for system (auxiliary disk subchannel in disk-based systems)
- 4 – standard output device
- 5 – standard input device
- 6 – standard list device

LU 8 is recommended for the magnetic tape device, if one is present in the system. Peripheral disks must be assigned LUs greater than 6. Additional LUs may be assigned for any functions desired.

Device Reference Table

The Device Reference Table (DRT) is part of the mechanism by which unit numbers for I/O are implemented. RTE users request I/O by specifying a logical unit number. The DRT is used to translate this logical unit number into a physical device, as specified by an EQT entry number and subchannel. The DRT is also used to queue requests for I/O on a device when it is unavailable (down). (The DRT is not used to queue requests when the device is up. The request list for available (that is, up) devices originates from word 1 of the EQT entry as illustrated in Figure 2-1.)

Each DRT entry is two words long (two and one-half words for RTE-6/VM). There is one entry for each logical unit number defined at generation time, beginning with logical unit 1. The format of each entry is illustrated in Figure 2-3. The word of the entry contains several items, including: 1) the EQT entry number of the controller assigned to the logical unit, and 2) the subchannel number of the specific device on that controller to be referenced. The second word of each entry contains the status of the logical unit: up (available) or down (unavailable). If the device is down, word two also contains a pointer to the list of requests waiting to access the LU. The extra half word in RTE-6/VM is used to indicate an LU lock (zero for no lock, non-zero for a resource number lock).

The DRT table is actually split into two separate parts (three parts for RTE-6/VM). The first part contains word 1 of each DRT entry, the second part contains word 2 of each DRT entry, and the third part (in RTE-6/VM) contains byte 5 of each DRT entry. This format is illustrated in Figure 2-4. The starting address and length of part one of the table can be found in the Base Page Communications Area. Part two is located in memory immediately following part one and has the same length as part one. Part three is located in memory immediately following part two.

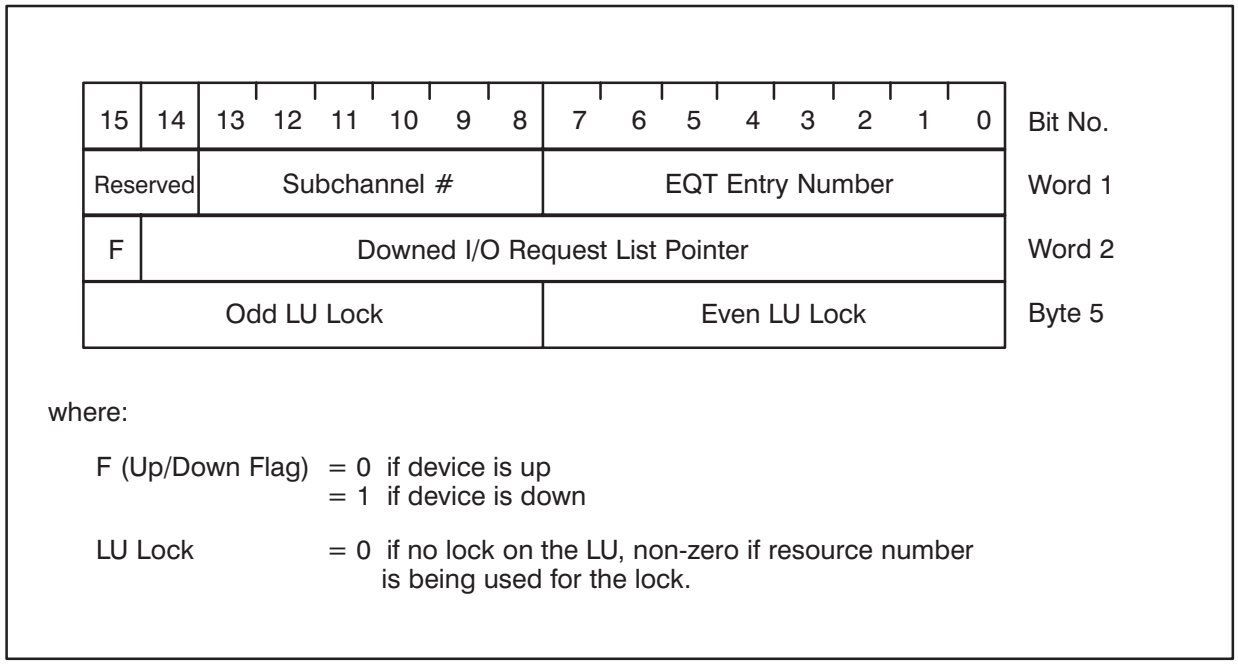

Figure 2-3A. Device Reference Table Entry Format (for RTE-6/VM)

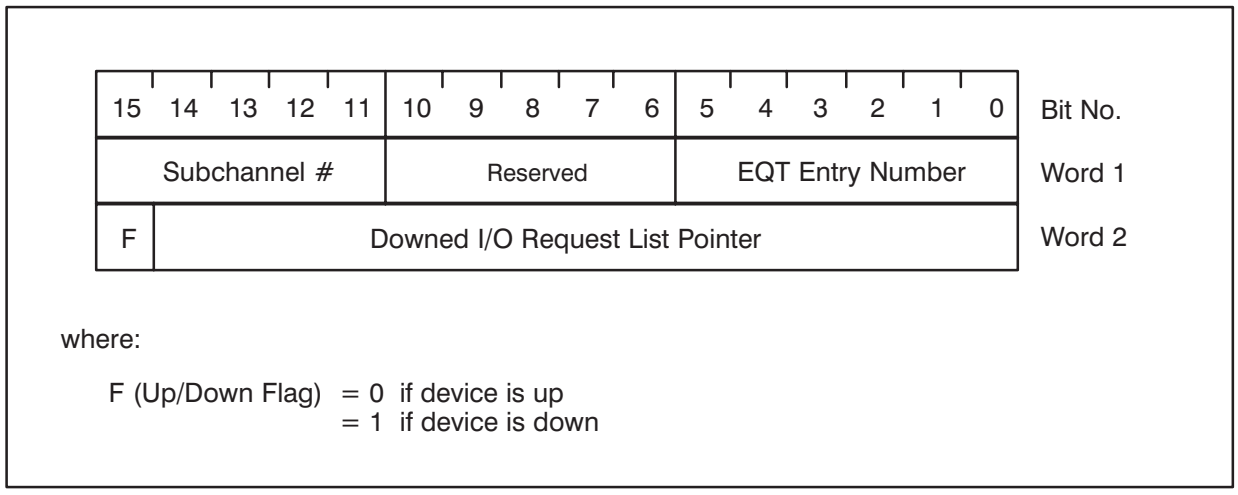

Figure 2-3B. Device Reference Table Entry Format (except for RTE-6/VM)

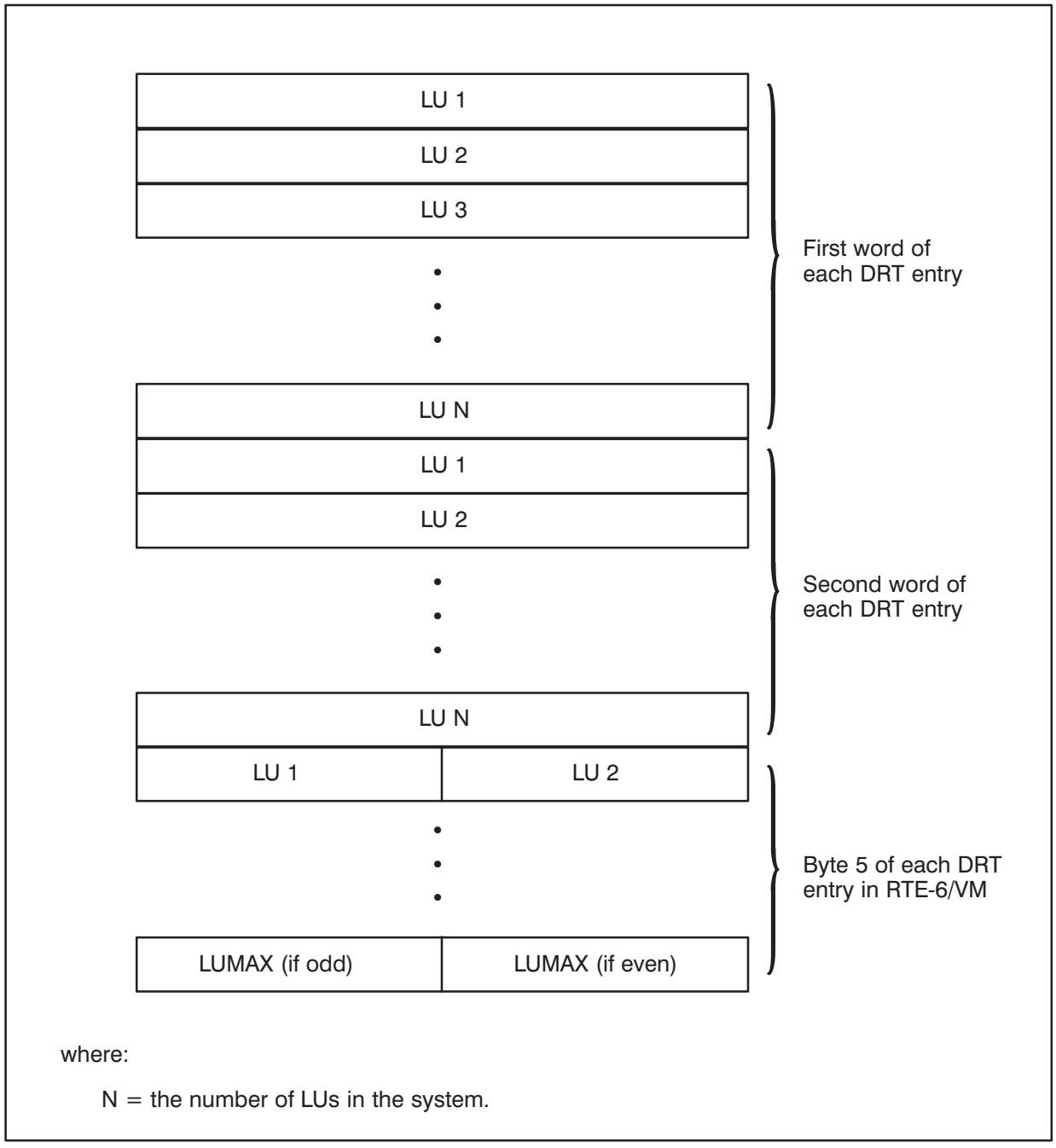

Figure 2-4. Device Reference Table

Computer Interrupt Mechanism

When a device controller interrupts RTE, the computer transfers control to one of a group of memory locations on base page known as the interrupt trap cell. The I/O select code of the interrupting controller determines the location of the transfer. For example, interrupts from select code 12 cause a transfer to memory location 12. Interrupts from select code 13 cause a transfer to memory location 13, and so on. Select code numbers range from 4 to 77 (octal). Thus, the group of memory locations from 4 to 77 (octal) comprises the entire set of interrupt trap cells.

Transferring control to an interrupt trap cell causes the instruction located there to be executed. For all devices operating under the control of CIC, this instruction is a JSB LINK,I, where LINK is a base page link containing the address of the entry point to CIC. This instruction is initially set up by the RTE generator and is reset into the trap cell each time the system is rebooted. In RTE-6/VM, the JSB LINK, I on an E-or F-Series CPU is replaced by four new instructions that handle these interrupts. The new instructions are described under Operating System Trap Cell instructions in the next chapter. The fact that the JSB instruction references an indirect address causes the computer to hold off any further interrupts for one instruction after the JSB. This gives CIC a chance to issue a CLF 0 instruction (which disables the interrupt system entirely) to prevent further interrupts from occurring while the current one is being processed.

Because CIC is entered at the same location for all device controller interrupts under its control, a method is needed by which the select code of the interrupting device controller can be determined. CIC obtains the interrupting select code number by accessing the contents of the computer's Central Interrupt Register via an LIA 4 instruction. CIC can then use this information to index into the Interrupt Table (see next subsection) to determine how to process the interrupt.

The interrupt trap cells are not limited to containing a JSB LINK,I instruction (where LINK contains the address of CIC). Other instructions can be placed in a trap cell by the generator or by a system routine. However, the trap cell should not contain any instruction other than a HALT instruction, an OS trap cell instruction (RTE-6/VM only) or a JSB indirect to an interrupt processing routine (such as CIC or a user-written routine) that saves the state of the machine on entry and restores it to its original state on exit. This includes saving and restoring the registers, the state of the memory protect fence, and so forth.

Specifically, I/O instructions and NOP instructions must not be put into trap cells because they do not provide a way to restore the system to its original state. Microcode macros (that is, jumps to microcoded routines) may be used if a microcoded driver is used to process the interrupts or if you are using the OS trap cell instructions (RTE-6/VM only).

Note that if a JSB instruction is placed in the interrupt trap cell, it must reference an indirect address. The indirect address keeps the interrupt system suppressed for one instruction after the JSB, as explained above. This allows the interrupt processing routine to issue a CLF 0 instruction to prevent further interrupts from occurring while the state of the machine is being saved. (Note that the generator automatically provides a base page link for all JSB instructions it places in the interrupt trap cells. A JSB indirect instruction is created whenever an "ENT," "PRG," or "EQT" entry is specified during generation.)

Systems without the powerfail/auto-restart feature have a HLT 4 instruction inserted by the generator into the powerfail interrupt trap cell (memory location 4). As a result, the computer will halt when a powerfail interrupt occurs. An example of a JSB to a user-written interrupt processing routine is discussed later in the "Writing Privileged Drivers" section of this manual.

Interrupt Table

The Interrupt Table directs CIC's actions when an interrupt occurs on any I/O select code that contains a JSB LINK,I instruction (where LINK contains the address of CIC). CIC can call a driver, schedule a specified program, or handle the interrupt itself.

There is one Interrupt Table entry for each I/O select code from 6 up to the highest select code defined in the system at generation. (Systems with I/O reconfiguration ability at bootup (for example, RTE-IV and RTE-6/VM) always include Interrupt Table entries for all select codes, even if some select codes were not defined in the initial generation.) Each Interrupt Table entry is one word long and can have three possible values: zero, positive, or negative.

1. If the entry is zero, the select code is undefined in the Interrupt Table. Any interrupts on this select code are illegal and cause the following message to be printed:

   ```
   ILL INT xx
   ```

   where *xx* is the octal I/O select code number. RTE then clears the interrupt flag on the select code and returns to the suspended process at the point of interruption. (Note that an Interrupt Table entry can also be zero if interrupts on the associated select code are handled by a special routine instead of by CIC and a driver. Refer to the "Writing Privileged Drivers" section later in this manual for more information on this subject.)

2. If the contents are positive, the entry contains the address of the EQT entry associated with the controller on the select code.

3. If the contents are negative, the entry contains the negative of the address of the ID segment of the program to be scheduled whenever an interrupt occurs on the select code. If such a program is not dormant when an interrupt occurs on the select code, the following message is output to the system console:

   `SC03 INT` *xxxxx*

   where *xxxxx* is the program name. RTE then clears the interrupt flag on the select code and control is returned to the suspended process at the point of interruption. This implies the program should be quick, if scheduled often, and preferably memory-resident.

All Interrupt Table entries are located sequentially in memory beginning with the entries for I/O select codes 6 and 7 (DCPC). This format is illustrated in Figure 2-5. There are no entries for I/O select codes 4 and 5 because the system is able to process interrupts from these select codes (powerfail interrupts, memory protect violations, and so forth) without the need for an Interrupt Table entry. The address of the first word of the table and the number of entries in the table can be found in the Base Page Communications Area.

Note Do not confuse the interrupt trap cell area of the computer, which is located on base page, with the Interrupt Table of RTE, which is located elsewhere. The interrupt trap cells are those memory locations (4 to 77 octal) to which control is transferred when an interrupt occurs. The Interrupt Table, on the other hand, is merely a convenient way for RTE to record what action CIC should take when an interrupt occurs on a select code under CIC's control.

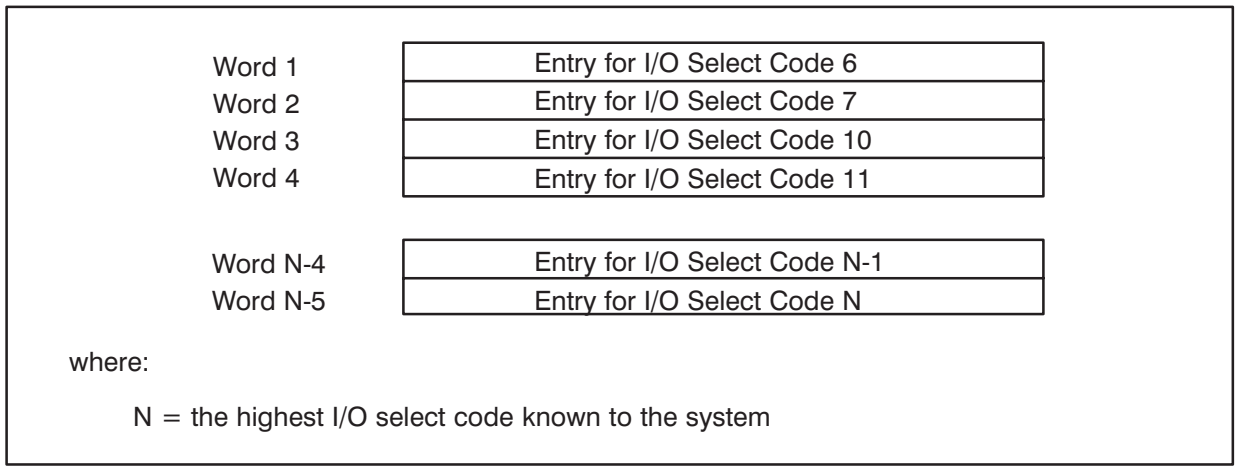

Figure 2-5. Interrupt Table

Driver Mapping Table (RTE-IV and RTE-6/VM Only)

In the RTE-IV and RTE-6/VM Operating Systems, drivers can be placed in one of two areas: in the System Driver Area (SDA) or in one of the driver partitions. Most standard drivers are placed in driver partitions. The SDA is primarily used for privileged drivers, drivers that do their own mapping, and very large drivers.

The Driver Mapping Table (DMT) is used to record where a driver resides in physical memory and other static and dynamic information about the driver and the location of the I/O request buffer.

There is one DMT entry associated with each EQT entry defined at generation time. Each entry is two words long, as illustrated in Figure 2-6. Word 1 is set up at generation time and its contents are never changed. It indicates whether the drive resides in the System Driver Area (SDA) or in a driver partition. If it is in the SDA, it also indicates whether or not the driver is doing its own memory mapping. (See the "Subroutines for Special Mapping Functions" subsection in Chapter 3.) If the driver is in a partition, word 1 also indicates the starting physical memory page number of the driver partition in which it is located.

Word 2 of the DMT entry is dynamic in nature and is set up at each I/O initialization of the associated EQT entry. This word indicates whether the I/O request buffer is located within a disk-resident program, memory-resident program, or system area. If a disk-resident program is making the request and the I/O request buffer is located within the program (that is, an unbuffered request), word 2 also indicates the physical memory page number of the disk-resident program's base page. This information is used to save time on setting up the proper map when processing interrupts handled by the driver.

General Operation of RTE I/O

Input/Output transfers in RTE can be conveniently broken into three parts for discussion: initiation, continuation, and completion. A user program is involved only in the initiation and completion phases, the system I/O processor and the device drivers are involved in all three phases. The following is a simplified discussion of each phase. As an aid to understanding this explanation, the general flow of events for an unbuffered I/O READ request is illustrated in Figure 2-7.

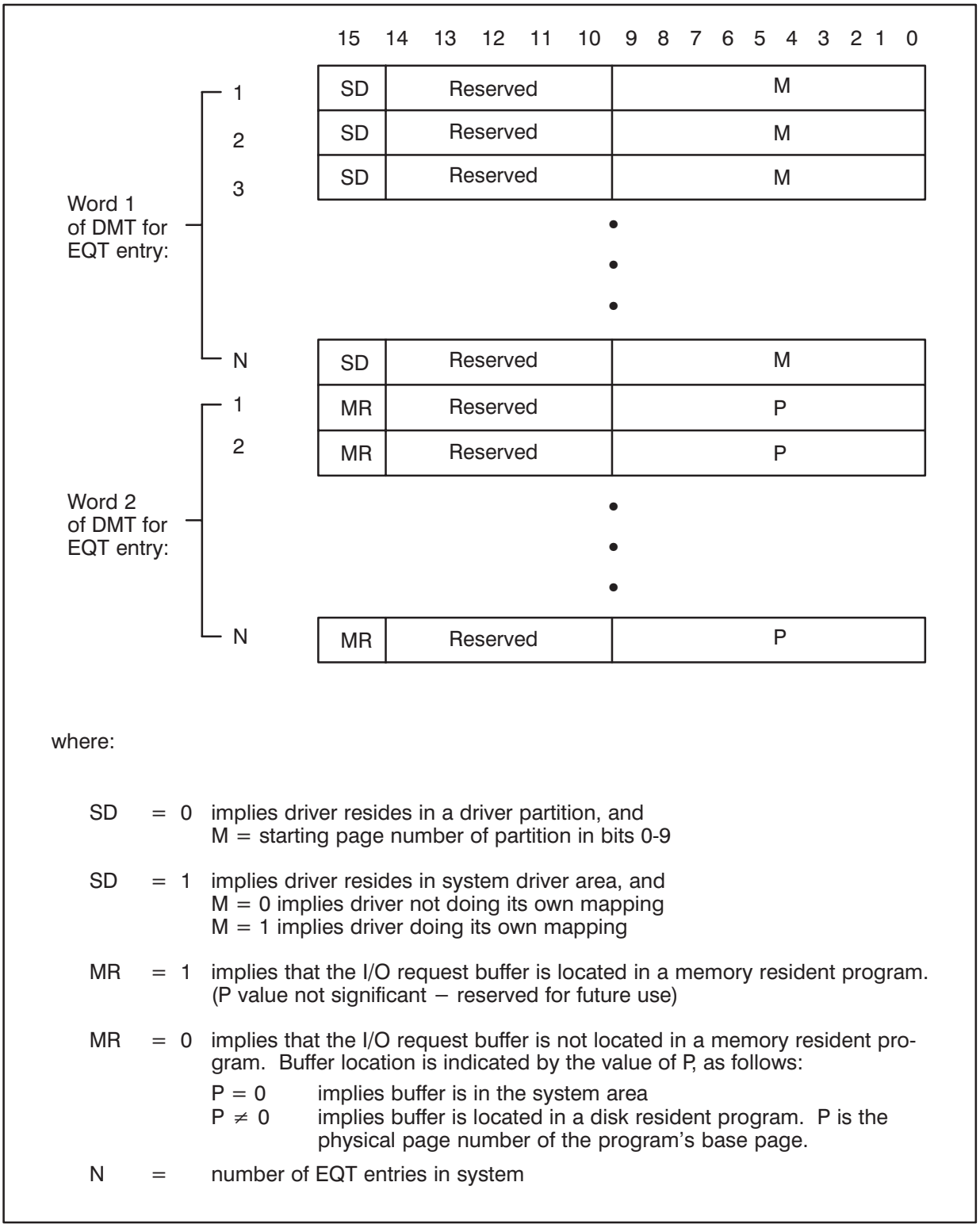

Figure 2-6. Driver Mapping Table

Figure 2-7. Unbuffered I/O READ Request

I/O Initiation

A user program makes an EXEC call to initiate I/O transfers. Parameters passed along with this call specify the logical unit, control information, buffer location, buffer length, and type of request (READ, WRITE, or CONTROL) to be made. The user request is channelled to the IOC (Input/Output Control) module of the system by the RTE request processor. The request is checked for legality and rejected if any errors are found. If there are no errors, the logical unit number supplied is used to index into the DRT (Device Reference Table) to determine which I/O controller (EQT entry number) and device (subchannel) are actually being referenced. The I/O request is then linked into the request list for the referenced controller.

When the device controller becomes available (that is, no prior requests are pending), the parameters of the request are put into the associated EQT entry, the addresses of the EQT entry words are set into the Base Page Communications Area, the proper map (System or User) is enabled (performed in systems with Dynamic Mapping only), and the "initiation" section of the driver is called. This section initializes the device controller, starts the data transfer or control function, and returns to IOC.

IOC then returns to the system's dispatching module to begin execution of the highest priority scheduled program. If the operation was successfully initiated by the driver, the data transfer is now under way.

I/O Continuation

When the device controller finishes transferring a data word or block of words, it interrupts the computer. This causes a transfer to one of the interrupt trap cell locations in the computer's memory, and the instruction located there is executed. For most I/O devices, this instruction is a JSB LINK,I (where LINK contains the address of the entry point to CIC). Execution of this instruction causes control to be transferred to CIC, the Central Interrupt Control module of the system. CIC obtains the number of the interrupting select code from the computer's Central Interrupt Register and uses it to index into the Interrupt Table. In RTE-6/VM, most of this is handled by a trap cell instruction.

For those I/O processes operating under the control of CIC and a driver, the Interrupt Table tells CIC which EQT entry is associated with the interrupting select code. CIC looks at the EQT entry, determines which driver is responsible for handling the interrupt, enables the correct map (System or User) in systems with Dynamic Mapping, and calls the driver's "continuation/completion" section to process the interrupt. The driver either accepts the data from the device (read operation) or sends more data to the device (write operation) and restarts the device. Return is then made to CIC with a code indicating that more interrupts are expected. This process (interrupt, CIC, driver, CIC) is repeated once for each word or block of words transferred until the entire transfer is complete.

I/O Completion

Eventually the driver will determine that the required amount of data has been transferred and that the I/O process is now complete. The driver then returns to CIC with a special code indicating that the I/O operation is complete and can be terminated; no more interrupts are expected.

CIC, in turn, transfers control back to IOC to terminate the IOC process. IOC causes the program that made the initial I/O request to be placed back into the scheduled list and checks to see if there are any other I/O requests pending for this controller. If at least one request is pending, the initiation section of the driver is again called to begin the next operation. IOC then returns control to the system's dispatching module to begin execution of the highest priority scheduled program.

3

Writing Standard RTE Drivers

Introduction

This section describes in detail the structure, operation, and design of standard RTE drivers. Standard drivers are fairly simple in structure and can generally be used to control most asynchronous devices. They can also be used to control synchronous and high-speed devices if these devices are driven under DCPC control. DCPC processing is also described in this section.

An alternate method for controlling synchronous and high-speed devices is to employ the more complex privileged driver. Chapter 4 of this manual describes the differences in the design of privileged drivers versus standard drivers. Thus, if you wish to design a privileged driver, the material in this section should be read and understood before continuing with the privileged driver discussion in Chapter 4.

Note that the operation of RTE requires that synchronous and high-speed devices be driven either by a standard driver utilizing DCPC transfers or by a privileged driver. This is necessary to ensure that interrupts from such devices are serviced within the required response time. You should keep this requirement in mind when deciding upon the type of driver to be written.

General Driver Structure and Operation

An I/O driver, operating under control of the Input/Output Control (IOC) and Central Interrupt Control (CIC) modules of RTE, is responsible for all data transfers between an I/O device controller and the computer. Each driver is written in two functional sections: an initiation section and a continuation/completion section. The initiation section is responsible for starting up the device and initiating the first data transfer. The continuation/completion section is responsible for processing each interrupt generated by the device under its control. This involves accepting data from the device (read operation), sending more data to the device (write operation), and then restarting the device to continue the transfer. Eventually, the continuation/completion section determines that a sufficient amount of data has been transferred and terminates the I/O operation.

A standard RTE driver operates with the interrupt system disabled, or effectively disabled, if the system contains a Privileged Interrupt card. Refer to the “Writing Privileged Drivers” section of this manual. This means that once a driver is entered to process an interrupt, no other interrupts (except privileged interrupts) can be serviced until the driver completes its operation and returns to CIC. (CIC turns the interrupt system back on to allow other interrupts to occur.) Drivers should therefore be coded as efficiently as possible to minimize the amount of time that the interrupt system is disabled and the processing of other interrupts is delayed.

Driver Naming Requirements

To facilitate the identification of driver programs and entry points, the following naming scheme has been devised. This scheme must be incorporated into the design of all RTE drivers so that the RTE system generator programs can identify the drivers and relocate them in the proper memory area of the operating system.

a. Driver names must be five characters in length, beginning with the characters "DV" and ending with a two-digit octal number (known as the equipment type code of the device).

b. The initiation and continuation/completion sections must have entry points whose names are four characters in length, beginning with the character "I" or "C", respectively, and ending with the same two-digit octal number used in the driver name.

Thus, if *nn* is the octal equipment type code, I*xnn* and C*xnn* are the entry point names of the initiation and continuation/completion sections, respectively. DV*ynn* is the driver name.

The user is allowed some flexibility in the choice of the *x* (in I*xnn* and C*xnn*) and *y* (in DV*ynn*) characters referred to above. This flexibility allows several drivers with the same octal equipment type code to have unique names and entry points. The rules for the choice of *x* and *y* are:

If *y* is "R" then *x* = "."

If *y* is not "R" then *x* = *y*

Using the above rules, a driver named DVR66 has entry points I.66 and C.66. A driver named DVA66 has entry points IA66 and CA66.

The octal equipment type code (*nn*) can be any octal number between 00 and 77. A table of "standard" type codes is given in Figure 2-1. Care should be taken to choose the type code and/or *x* and *y* characters so that new driver names and entry points do not conflict with those of any standard HP drivers or other user-written drivers present in the system.

Initiation Section

The IOC module of RTE calls the driver initiation section when an I/O transfer is initiated. (Note that the initiation section may also be entered when a DCPC channel is assigned by IOC in response to a dynamic DCPC request by the driver. Refer to the "DCPC Processing" subsection of this manual.) Prior to actually entering the driver initiation section, IOC sets up all information needed by the driver to process the call in the associated EQT entry and in the Base Page Communications Area, as follows:

a. Locations EQT1 through EQT15 in the Base Page Communications Area are set to contain the address of each word of the EQT entry associated with the call. Base page word EQT1 is set to contain the address of EQT entry word 1, base page word EQT2 is set to contain the address of EQT entry word 2, and so on. If the driver uses DCPC (that is, if bit 15 of EQT entry word 4 is set), IOC also assigns a DCPC channel to the driver and stores the DCPC channel number in base page word CHAN.

b. Words 6 through 10 of the EQT entry pointed to by the Base Page Communications Area are set to contain the request parameters from the user's EXEC call (request code, subfunction, buffer address, buffer length, and optional parameters, if present). Note that EQT entry word 6 (CONWD) contains the CONWD from the user's EXEC call, modified to contain the request code in bits 0 and 1 in place of the logical unit. The subchannel being referenced by the call is placed into bits 6 through 10 of EQT entry word 4. For RTE-6/VM only, the subchannel number occupies six bits, the sixth (most significant) bit being bit 2 of word 6. Refer to the EQT diagram in Figure 2-1.

c. CIC also sets up and enables the correct map (System or User) needed by the driver to process the call. This step is performed in systems with Dynamic Mapping only.

d. EQT word 14 is copied to EQT word 15 to set the default timeout.

After performing these tasks, IOC enters the driver directly via a jump subroutine to the initiation entry point Ixnn (JSB I*xnn*). Upon entry, the A-Register contains the I/O select code of the controller being referenced in the call. (This same information is present in bits 0 through 5 of EQT entry word 4.) Later, when the driver has completed (or rejected, if necessary) the initialization procedure, it must return to IOC via a jump indirect through the Ixnn entry point (JMP I*xnn*,I).

Once entered, the driver is free to use EQT entry words 6 through 13 in any way, but words 1 through 4 must not be altered, except for the D, P, and S bits of EQT word 4. If an EQT entry extension was specified at generation, the space in the extension is also available to the driver. In this case words 12 and 13, which define the extension, must not be modified. The driver can also update the status field in word 5, if appropriate, but this must be done without altering the rest of word 5. Finally, EQT entry word 15 may be modified, if desired, to override the default timeout value for the device. Refer to the "I/O Controller Timeout" subsection in this manual.

EQT word 14 can be altered to permanently change the default timeout value (for example, the serial drivers do this in response to a CN 22 call).

Figures 3-1 and 3-2 show the EQT fields set by RTE and the generator, while Figure 3-3 shows the EQT fields the driver can modify.

Word	Contents															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	R	I/O Request List Pointer <C>														
2	R	Driver “Initiation” Section Address <A>														
3	<F> A	Driver “Continuation/Completion” Section Address <A>														
4	<A> D	<B> B	<E> P	<E> S	<C> T	Lower 5 Bits of Subchannel # <C>					I/O Select Code # <A>					
5	AV <F>	Equipment Type Code<A>						Status <E>								
6	CONWD (Current I/O Request Word) <C>													<C> MSB		
7	Request Buffer Address <C>															
8	Request Buffer Length <C>															
9	Temporary Storage <D> or Optional Parameter <C>															
10	Temporary Storage <D> or Optional Parameter <C>															
11	Temporary Storage for Driver <D>															
12	Temporary Storage for Driver <D> or EQT Extension Size, if any <A>															
13	Temporary Storage for Driver <D> or EQT Extension Starting Address, if any <A>															
14	Device Timeout Default Value<B>															
15	Device Timeout Clock<C>															

Figure 3-1. EQT Fields Set by RTE (shaded portions)

Word	Contents															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	R	I/O Request List Pointer <C>														
2	R	Driver “Initiation” Section Address <A>														
3	<F> A	Driver “Continuation/Completion” Section Address <A>														
4	<A> D	<B> B	<E> P	<E> S	<C> T	Lower 5 Bits of Subchannel # <C>					I/O Select Code # <A>					
5	AV <F>	Equipment Type Code<A>						Status <E>								
6	CONWD (Current I/O Request Word) <C>													<C> MSB		
7	Request Buffer Address <C>															
8	Request Buffer Length <C>															
9	Temporary Storage <D> or Optional Parameter <C>															
10	Temporary Storage <D> or Optional Parameter <C>															
11	Temporary Storage for Driver <D>															
12	Temporary Storage for Driver <D> or	EQT Extension Size, if any <A>														
13	Temporary Storage for Driver <D> or	EQT Extension Starting Address, if any <A>														
14	Device Timeout Default Value<B>															
15	Device Timeout Clock<C>															

Figure 3-2. EQT Fields Set by the Generator (shaded portions)

Word	Contents															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	R	I/O Request List Pointer <C>														
2	R	Driver “Initiation” Section Address <A>														
3	<F> A	Driver “Continuation/Completion” Section Address <A>														
4	<A> D	<B> B	<E> P	<E> S	<C> T	Lower 5 Bits of Subchannel # <C>					I/O Select Code # <A>					
5	AV <F>				Equipment Type Code <A>						Status <E>					
6	CONWD (Current I/O Request Word) <C>													<C> MSB		
7	Request Buffer Address <C>															
8	Request Buffer Length <C>															
9	Temporary Storage <D> or Optional Parameter <C>															
10	Temporary Storage <D> or Optional Parameter <C>															
11	Temporary Storage for Driver <D>															
12	Temporary Storage for Driver <D> or EQT Extension Size, if any <A>															
13	Temporary Storage for Driver <D> or EQT Extension Starting Address, if any <A>															
14	Device Timeout Default Value <B>															
15	Device Timeout Clock <C>															

Figure 3-3. EQT Words the Driver Can Modify (shaded portions)

Functions of the Initiation Section

As part of the general I/O structure of RTE, the initiation section of a standard driver performs the functions illustrated in Figure 3-4. A more detailed description of the initiation section functions is given below.

1. Checks for powerfail/auto-restart entry by examining bit 15 of EQT entry word 5, which is set to 1 only on this type of entry. If bit 15 is set, the appropriate powerfail/auto-restart processing should be done. This check need only be made by drivers that are designed to process powerfail interrupts as described in the "Powerfail Processing" subsection of this manual.

2. Rejects the request by following the procedure described in step "7" if:

   a. A status check of the device or controller indicates that it is inoperable, or

   b. The request code or other parameters are illegal.

3. Configures all I/O instructions in the driver to reference the specific I/O select code (and DCPC channel, if used) of the device controller.

4. Initializes DCPC, if used. Refer to the "DCPC Processing" subsection of this manual.

5. Initializes software flags and activates the device controller. All variable information pertinent to the transmission must be saved in the EQT entry associated with the controller, because the driver may be called for another controller before the first operation is complete.

6. Optionally sets the device controller timeout clock (EQT entry word 15) to modify the default timeout value inserted there by the system. Refer to the "I/O Controller Timeout" subsection of this manual.

7. Returns to IOC (via JMP I*xnn*,I) with the A-Register set to indicate initiation or rejection (and the cause of the rejection) as follows:

A-Reg	Status
0	The operation was initiated successfully.
1	Operation rejected: read or write illegal for device (IO07, program aborted).
2	Operation rejected: control request illegal or undefined.
3	Operation rejected: equipment malfunction or not ready (IONR message issued).
4	The operation was immediately completed. This means that the driver was able to completely satisfy the request without the need of a subsequent interrupt and that the program making the I/O call can be rescheduled immediately. The B-Register should be set to the positive number of words or characters (depending upon which the user specified) transferred. This value is known as the transmission log.
5	A DCPC channel is required, but none was assigned by IOC. This can only occur when the "DCPC channel required" bit is not set in the EQT entry, and the driver decides that it needs a DCPC channel to process this specific call. Refer to the "DCPC Processing" subsection of this manual.

6 A DCPC channel was assigned by IOC, but the driver did not use it and wants to give it up. After IOC deallocates the channel, processing continues as for a successful initiation return (A = 00). (RTE-IV and RTE-6/VM only.)

7 to 99 IO*xx* message issued: the program making the I/O request is aborted (unless the no-abort bit was set in the call), and an I/O error message is printed on the system console. Note that this return can be used for unbuffered user I/O requests only. This, therefore, excludes the use of return codes 7 through 99 on any Class, buffered or system I/O requests. The error mesage has the following format:

```
IOxx yyyyy
NNNNN ABORTED
```

where:

xx = the return code from the driver (decimal 07 to decimal 99),

yyyyy = the address of the aborted I/O request in program *NNNNN*, and

NNNNN = the name of the program that made the I/O request.

This type of return can be used by drivers to generate their own I/O error messages at the system console. Note that certain codes are reserved for system use, as follows:

Return Code	Reserved for
7 – 59	HP system modules and system drivers.
60 – 99	User written drivers.

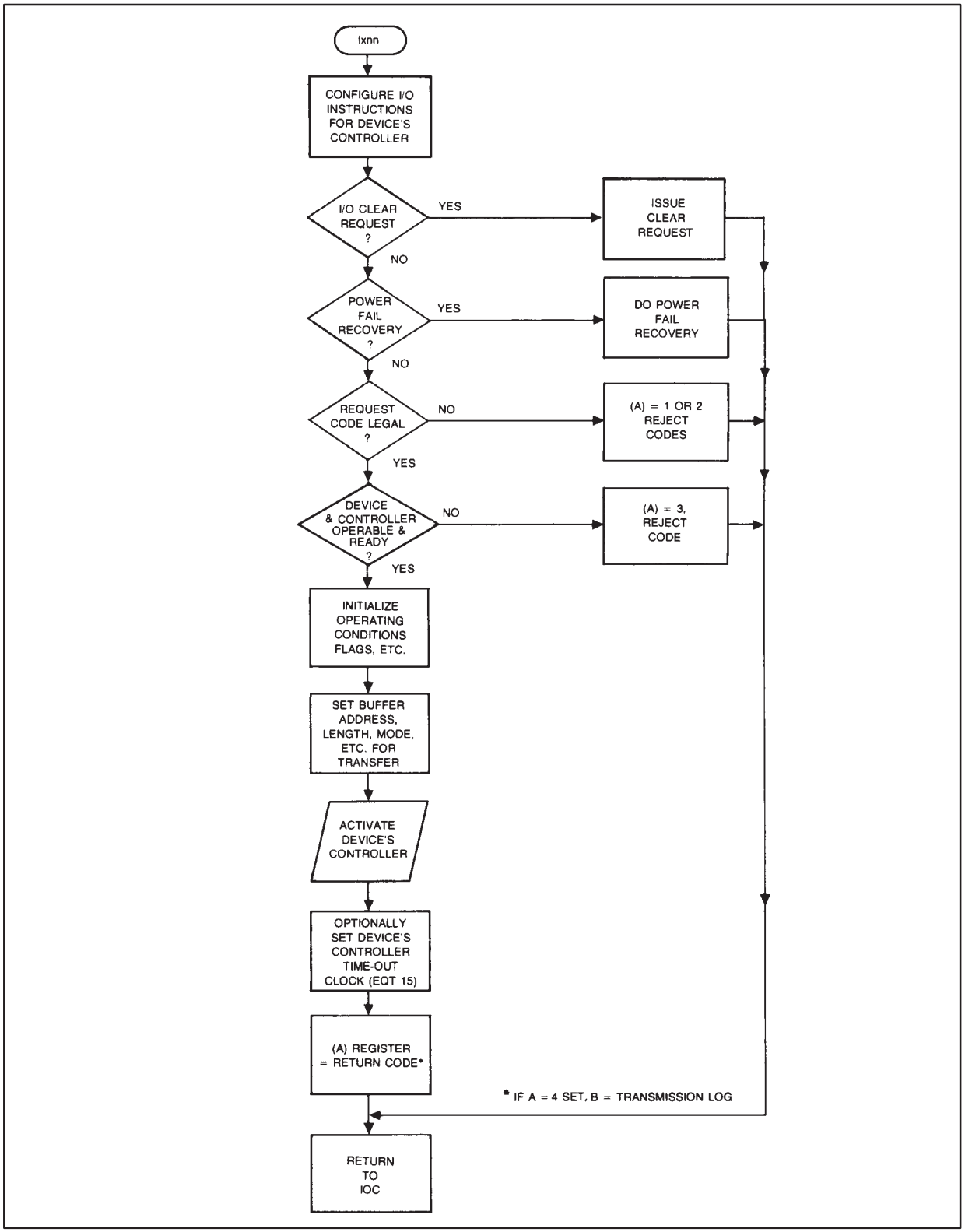

Figure 3-4. I/O Driver Initiation Section

Continuation/Completion Section

The CIC module of RTE calls the continuation/completion section of a driver whenever an interrupt is recognized on an I/O controller associated with the driver. Before calling the driver to process the interrupt, CIC issues a clear flag instruction (CLF) to the interrupting select code, sets the addresses of the associated EQT entry into the Base Page Communications Area, sets the interrupt source code (I/O select code of interrupting controller) into the A-Register, and copies EQT word 14 into EQT word 15. The interrupting I/O select code address is also available in EQT entry word 4.

CIC also sets up and enables the correct map (System or User) needed by the driver to process the call. (This step is performed in systems with Dynamic Mapping only.) The driver is then entered with the correct map enabled by executing a jump subroutine to the continuation/completion section of the driver at entry point C*xnn* (JSB C*xnn*). This call has the following format:

Location	**Action**
	(Set A-Register equal to interrupt source code)
P	JSB C*xnn*
P+1	Completion return from C*xnn*
P+2	Continuation return from C*xnn*
P+3	Get/give up DCPC continuation return from C*xnn* (RTE-IV and RTE-6/VM only)

The return points from C*xnn* to CIC indicate whether:

1. The transfer has been completed.

2. The transfer is continuing, that is, further interrupts are expected from the device controller.

3. The driver is requesting the allocation or deallocation of a DCPC channel.

The continuation/completion section of the driver is flowcharted in Figure 3-5 and performs the following functions. Note that steps “1” through “5” are always executed whenever the driver is entered at C*xnn*. Then, depending on whether the I/O operation is now complete or is still continuing, the driver executes either step “6” (completion return) or step “7” (continuation return), respectively.

1. Checks whether bits 14-0 of EQT entry word 1 (the controller I/O request list pointer) equal zero. If so, a spurious interrupt has occurred (that is, no I/O operation was in progress on the controller). The driver can handle the spurious interrupt in one of two ways:

   a. Ignore the interrupt by setting EQT entry word 15 (the timeout clock) to zero to prevent timeout and making a continuation return as described in step “7” below.

   b. Ignore the interrupt and have RTE display a message by making a completion return as described in step “6”. No special A-Register return value is needed. Upon return from the driver, RTE looks at the I/O request list pointer and, if the pointer is zero, displays the following message on the system console (where the *sc* indicates the interrupting select code):

      ```
      ILL INT sc
      ```

2. If the interrupt is valid, that is, bits 14-0 of word 1 are non-zero, the driver configures all I/O instructions in the continuation/completion section to reference the interrupting select code.

3. Checks to see if a timeout has occurred on the device by checking bit 11 of EQT entry word 4. If this bit is set, the device has timed out and any required timeout processing should be done. Note that this check need only be made by drivers that are designed to process timeout interrupts (as described in the "I/O Controller Device Timeout" subsection of this manual). Drivers not processing timeout interrupts are not entered on device timeout.

4. If both the DCPC and the device controller interrupts are expected, but only the device controller interrupt is significant, the DCPC interrupt can be ignored by making a continuation return to CIC as described in step "7" below. Refer to the "DCPC Processing" subsection of this manual for a method to suppress the DCPC interrupt entirely.

5. Performs the input or output of the next data item if the device is driven under program control. One of three possible actions is then taken:

   a. If the transfer is not complete, the driver follows the procedure described in step "7" below to make a continuation return.

   b. If the transfer is complete, the driver follows the procedure described in step "6" below to make a completion return.

   c. If the driver detects a transmission error, it can reinitiate the transfer and attempt a retransmission. A counter for the number of retry attempts can be kept in the EQT entry. After initiating each retry, the driver makes a continuation return to CIC as described in step "7" below.

6. (Completion Return.) At the end of a successful transfer or after completing the retry procedure, the driver performs the following steps before returning to CIC:

   a. Sets the actual or simulated device controller status into bits 7 through 0 of EQT entry word 5 without altering the rest of word 5.

   b. Sets the number of words or characters (depending on which the user requested) transmitted into the B-Register. This value is known as the transmission log.

   c. Clears the device controller (and DCPC if used).

   d. Sets the A-Register to indicate successful completion and the reason as follows:

If A = 0	The operation was successfully completed.
If A = 1,2,3,4	The operation was not completed, where: 1 = device or controller malfunction or not ready (IONR issued) 2 = end-of-tape or end-of-information (IOET issued) 3 = transmission parity error (IOPE issued) 4 = device timeout (IOTO issued)

   e. Return to CIC at P+1 (JMP C*xnn*,I).

7. (Continuation return.) When the driver wishes to continue the transfer, that is, additional interrupts are expected, the driver performs the following steps before returning to CIC at P+2:

   a. Sets the device controller (and DCPC if used) for the next transfer or retry.

   b. Optionally sets the device timeout clock (EQT entry word 15) to modify the value inserted there by the system. Refer to the "I/O Controller Timeout" subsection of this manual.

   c. Returns to CIC at P+2 or (for RTE-IV and RTE-6/VM) P+3 as follows:

```
ISZ Cxnn          or          ISZ Cxnn
JMP Cxnn,I                    ISZ Cxnn
                              JMP Cxnn,I
```

   for the P+3 return, the contents of the A-Register determines the action taken by the system. The A-Register should be set as follows:

   A = 5 Request the allocation of a DCPC channel. After assigning a DCPC channel, the system will re-enter the driver at I*xnn*.

   A = 6 Request the deallocation of a DCPC channel. After releasing the DCPC channel, the system continues processing as if this were a P+2 continuation return.

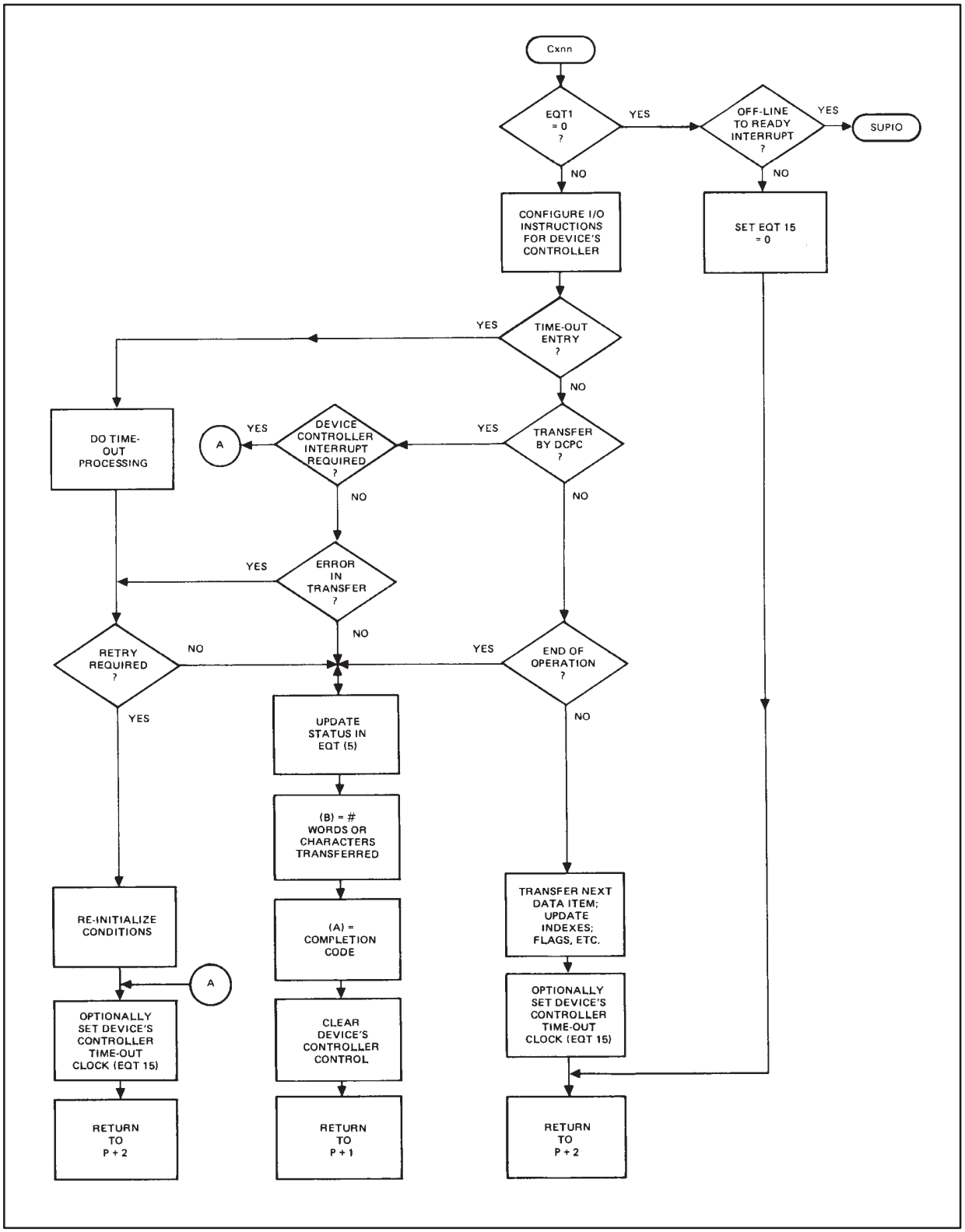

Figure 3-5. I/O Driver Continuation/Completion Section

Device Clear on Program Abort

If an I/O suspended program is aborted while waiting for a controller, the system attempts to clear the controller by issuing a clear control request to the driver, that is, request code 3, subfunction code 00, as indicated in EQT entry word 6. All drivers written for use in RTE must be prepared to handle this request even if no other control requests are supported for the controller.

If the controller can be cleared without interrupt (that is, immediately), the driver should perform the clear operation and return to IOC with the A-Register equal to 2 (control request illegal) or 4 (immediate completion). Either return is sufficient in this case, they are both treated the same by the system.

If an interrupt is required, the driver should return with the A-Register equal to 0. In this case, the system forces a 1-second timeout for the controller. When the device controller interrupts, the driver should complete the clear operation and make a successful completion return (A-Register = 0) to CIC at P+1. However, if the interrupt does not occur within the 1-second timeout, the system itself issues a clear control command (CLC) to the controller's select code(s). Note that in this case the driver is not entered to process the timeout even if it had previously set the "driver processes timeout" bit in EQT entry word 4. Refer to the "I/O Controller Timeout" subsection in this manual.

I/O Controller Timeout

Each I/O controller can have a timeout clock to prevent indefinite I/O suspension. Indefinite I/O suspension can occur when a program initiates I/O, and the device controller fails to return a flag indicating that the transfer is complete. This can occur as the result of either a hardware malfunction or improper program encoding. Without the controller timeout, the program making the I/O call would remain in I/O suspension indefinitely, awaiting the completion indication from the device controller.

EQT entry words 14 and 15 function as an I/O controller's timeout clock. EQT entry word 15 is the actual working clock. Prior to each call to the driver, word 15 is set to a value "m," where "m" is a negative number of ten-millisecond time intervals. Thereafter, this counter is incremented by one at each ten-millisecond tick of the system's real-time clock. If the controller does not interrupt within the required time interval (that is, before the counter in EQT entry word 15 goes to 0), it is considered as having "timed out".

The EQT entry word 15 clock for each controller can be individually set by either of the following two methods:

1. The system always inserts the contents of EQT entry word 14 (a negative number) into EQT entry word 15 before the initiation or continuation/completion section of a driver is entered, unless it is running (EQT word 15 is not zero). Word 14 can be preset to "m" by entering a timeout value during the EQT entry phase of generation, or it can be set or modified online with the TO operator request.

2. When the driver initiates I/O and expects to be entered due to a subsequent interrupt, the driver itself can set the value "m" into EQT entry word 15 just before it exits. The value "m"

can either be coded permanently into the driver or can be passed to the driver as an I/O request parameter.

Note The system always inserts the contents of EQT entry word 14 into EQT entry word 15 before entering a driver, with the following exceptions: 1) if an initiation call is being made and word 15 is already non-zero, it is not reset, and 2) if a continuation call is being made and word 14 is zero, word 15 is not reset. In any case, a timeout value inserted by the driver directly into word 15 overrides any value previously set by the system.

A timeout value of zero is equivalent to not using the timeout feature for a particular controller. If a timeout parameter is not entered, its value remains zero and timeout is disabled for the controller.

Timeout is enabled for a controller only while the controller is processing an I/O request. The working timeout clock (EQT entry word 15) is set as described above. If the controller does not time out, the clock is then cleared by the system after one of these driver returns:

1. An Initiation return indicating a rejected initiation request (A-Register not equal to zero).

2. A Completion return.

Driver Processing of Timeout

When a controller times out, a driver has the option of performing its own timeout processing or of letting the system handle it entirely. A driver that processes its own timeouts indicates this to the system by setting bit 12 of EQT entry word 4. Because the system never clears this bit, it needs to be set only once. When bit 12 is set, the following action takes places upon controller timeout:

1. Bit 1 in EQT entry word 4 is set by the system.

2. The driver is entered at C*xnn* with the A-Register set to the I/O select code of the controller that timed out. The same information is available in EQT entry word 4.

3. The driver recognizes that the entry is for timeout by examining bit 11 of EQT entry word 4. When bit 11 is set, a timeout has occurred, and the driver should perform whatever processing is necessary. This can involve completing the operation in progress or restarting the device and continuing the operation. If the latter option is taken, the driver should clear bit 11 prior to exiting in case it is entered again before completion of the operation. This enables the driver to distinguish between a normal continuation entry (bit 11 = 0) and another timeout entry (bit 11 = 1). Note that IOC only clears bit 11 prior to entering the driver at I*xnn* on an initiation call.

4. The driver may decide to continue (that is, restart the device) or complete (that is, terminate) the operation, as follows:

   a. If the driver decides to complete the operation, it sets the A-Register equal to 4 (to indicate that a timeout has occurred), sets the B-Register equal to the transmission log, and returns to CIC at P+1. This causes CIC to set the LU down and to print the following message:

      ```
      I/O TO L #x E #y S #z
      ```

      where:

      #x is the LU number being set down,

      #y is the number of the EQT entry associated with this LU, and

      #z is the channel associated with this LU.

      It is possible to complete the operation without having a message printed. To do this, the driver simply makes a normal completion return (A-Register = 0, B-Register = transmission log) to CIC at P+1.

      In either case (A = 4 or A = 0), CIC reschedules the calling program and passes it the transmission log returned by the driver.

   b. If the driver decides to continue the operation, it makes a normal continuation return to CIC at P+2.

System Processing of Timeout

When a timeout occurs and bit 12 of EQT entry word 4 is not set, the system handles the interrupt itself in the following way:

1. The program that made the initial I/O request is rescheduled, and a zero transmission log is returned to it.

2. The LU is set down, and the following message is printed:

   ```
   I/O TO L #x E #y S #z
   ```

   where:

   #x is the LU being set down,

   #y is the number of the EQT entry associated with this LU, and

   #z is the subchannel associated with this LU.

3. A clear control instruction is issued to the controller's select code(s) through the EQT entry number located in the Interrupt Table.

Note that the driver is never entered for timeout processing when bit 12 of EQT entry word 4 is zero. This means that only those drivers that set bit 12 to indicate that they are to process timeout need to check for a timeout entry.

Because the system issues a CLC instruction to the controller's select code(s), each controller interface card requires an entry in the Interrupt Table during generation. Otherwise, the system would not be able to issue the CLC instruction when a controller timed out.

Note Some drivers expect an "*sc*,PRG,*xxxx*" entry in the interrupt table phase of the generation to cause the ID segment address of program "*xxxx*" to be put in the Interrupt Table. On initialization, the driver will copy the ID segment address to the EQT or EQT extension, and then copy its own EQT address into the Interrupt Table.

DCPC Processing

The Dual Channel Port Controller (DCPC) feature of the HP 1000 M/E/F-Series of computers can be used to transfer blocks of data between I/O devices and the computer at high data transfer rates. The DCPC transfers are initiated in software, but the actual word-by-word transfer is handled under hardware control. Words are transferred to or from the computer via a "cycle stealing" technique that operates concurrently with the normal execution of programs. This design eliminates the overhead associated with driver processing of individual interrupts and allows synchronous and high-speed devices to be controlled by standard RTE drivers.

This subsection discusses some of the aspects of DCPC transfers in the RTE operating systems. It is assumed that you are already familiar with the general techniques of DCPC programming as described in the appropriate computer reference manual.

RTE Control of DCPC Assignment

RTE controls the allocation of the two DCPC channels available via the first two words of the Interrupt Table. Interrupt Table entry word 1 records the current assignment of DCPC channel 1, and word 2 records the current assignment of DCPC channel 2. This arrangement is illustrated in Figure 3-6. This figure also illustrates the format of each individual DCPC Assignment Word. Note that DCPC channels 1 and 2 generate interrupts on I/O select codes 6 and 7, respectively, and hence are often referred to as DCPC channels 6 and 7.)

Each DCPC Channel Assignment Word in the Interrupt Table can be in one of two states, assigned or unassigned. If the entire word is zero, the respective DCPC channel is unassigned and is, therefore, available for use. A non-zero word implies that the DCPC channel is currently assigned. Bits 0 through 14 contain the address of the EQT entry (which in turn points to the driver) to which the DCPC channel has been allocated. Once a DCPC channel is allocated, a driver can set bit 15 in the appropriate DCPC Channel Assignment Word as a flag to the operating system. Use of this flag is fully explained later in this subsection.

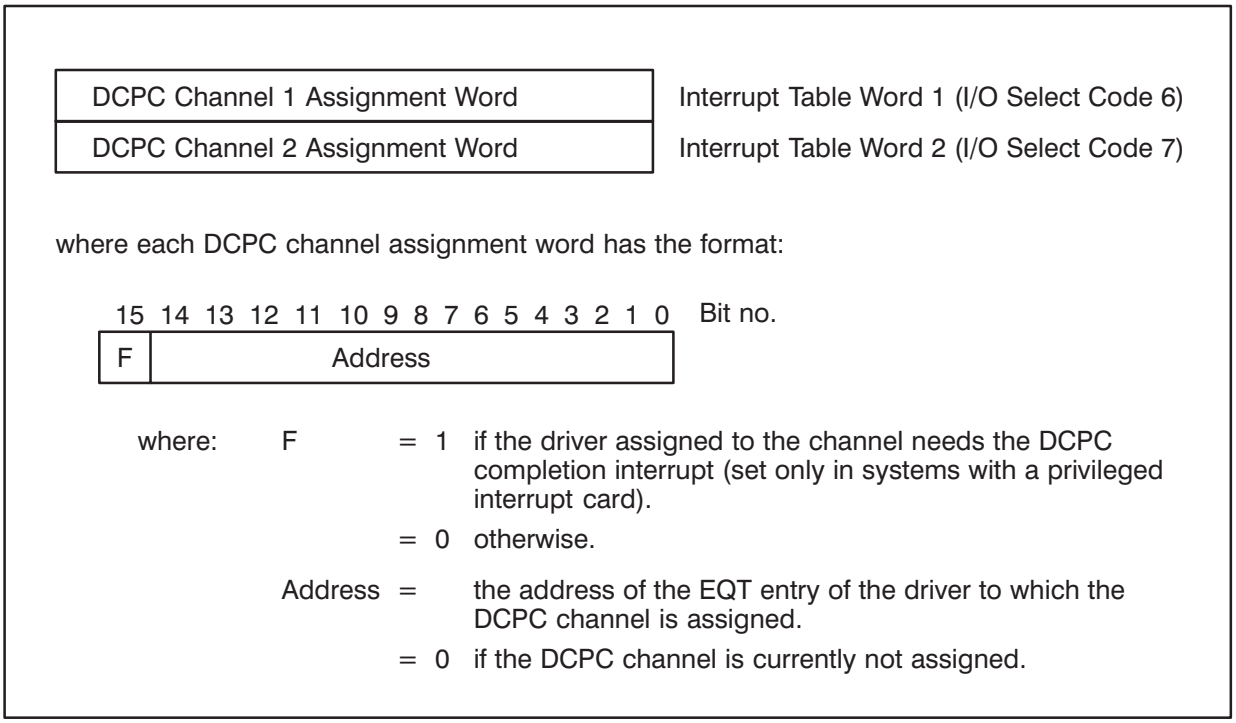

Figure 3-6. DCPC Channel Assignment Words

DCPC Assignment by RTE

Before a driver can initiate a DCPC transfer, it must be assigned by RTE the exclusive use of a DCPC channel. This prevents simultaneous access to the channel by several drivers. A driver can be assigned a DCPC channel in the following two ways:

Preferred Method

If the driver's EQT entry had a "D" specified at generation time, the "DCPC channel required" bit is permanently set in the EQT entry (EQT entry word 4, bit 15). In this case, the system always assigns a DCPC channel to the driver at each I/O initiation. The assigned DCPC channel number can be found (at initiation only) in the Base Page Communications Area word CHAN. It should then be saved in one of the temporary storage words of the EQT entry because it is not available in CHAN on later entries to the driver for the same I/O request.

Alternate Methods

If a driver needs a DCPC channel only for a certain subset of the functions that it performs, it can a) set the "D" bit as described above and return the DCPC when it is not needed, or b) dynamically ask the system to assign it a DCPC channel as required. In this case, the DCPC option is not selected for the driver's EQT entry at generation time, and the "DCPC channel required" bit is not set in the EQT entry.

A driver can dynamically request a DCPC channel from either its Initiation or (RTE-IV and RTE-6/VM) Continuation/Completion sections. Each method is described in the following subsections.

Alternate Method I: Initiation Request

The driver will be entered at I*xnn* without a DCPC channel assigned by IOC. The driver must analyze the request and determine if a DCPC channel is required. If so, the driver requests a DCPC channel from IOC by returning via a jump indirect through I*xnn* (JMP I*xnn*,I) with the A-Register equal to 5. IOC then assigns a DCPC channel and recalls the driver.

However, IOC does not differentiate between the initial call to the driver and the recall with the DCPC channel assigned. The EQT entry is set up identically in both cases, and the driver is entered at I*xnn*. Furthermore, it is possible that the driver may never be recalled with the DCPC channel assigned for a particular I/O request. For example, this can occur if the program making the request is aborted before IOC has a chance to assign the DCPC channel. If the program is aborted, the driver will not be entered again until another program requests I/O for a device under the driver's control.

Thus, a driver can never know from the calling parameters or from its past history whether it is being called for the first time for an I/O request (that is, no DCPC channel is assigned) or whether it is being recalled with a DCPC channel now assigned. The only way the driver can distinguish between these two cases is to access the two DCPC Channel Assignment Words (in the Interrupt Table) to determine whether a DCPC channel is currently assigned to the driver.

If the value in either DCPC Channel Assignment Word is equal to the address of the EQT entry currently being serviced by the driver, that DCPC channel is currently assigned to the driver. The driver can then assume that it has been recalled with a DCPC channel assigned and can initiate the transfer on that DCPC channel. If neither value matches, no DCPC channel is assigned to the driver, and it must return to IOC to request that one be assigned. Note that in this case the driver cannot use Base Page Communications Area word CHAN as an indication of whether or not it has been assigned a DCPC channel. This is because CHAN indicates only which channel was last assigned to any driver, not to whom it was assigned.

The following code illustrates the DCPC assignment check technique. In reviewing this and subsequent examples, remember that the Base Page Communications Area word INTBA contains the address of the Interrupt Table and that base page word EQT1 contains the address of the EQT entry currently being serviced by the driver.

```
CHDCP  EQU  *          Execute this code if DCPC required
       DLD  INTBA,I    Access DCPC Channel Assignment Words
       CPA  EQT1       Is DCPC channel 1 assigned to this driver?
       JMP  CH1        Yes, configure and initiate transfer on channel 1
       CPB  EQT1       Is DCPC channel 2 assigned assigned to this driver?
       JMP  CH2        Yes, configure and initiate transfer on channel 2
       LDA  =B5        No.  A DCPC channel is not assigned.  Set
       JMP  Isnn,I     A = 5 to request one from IOC, and return
```

Note that if a driver obtains a DCPC channel in this way, a special procedure must also be followed to return the DCPC channel to RTE. The return procedure is discussed in the "Returning DCPC Channels to RTE" subsection.

Alternate Method II: Continuation Request

Alternately, in RTE-IV and RTE-6/VM, the driver can request a DCPC channel from IOC by returning via a jump indirect through C*xnn* with the A-Register equal to 5 as follows:

```
(Set A-Register to 5)
ISZ C xnn
ISZ C xnn
JMP C xnn,I
```

IOC then assigns a DCPC channel and recalls the driver, entering at I*xnn*. In this case, IOC does not reset the EQT before entering at Ixnn.

The Initiation section of the driver can determine if this method is being used to acquire a DCPC channel by examining bit 15 of word 3 in the EQT. If this bit is set, then the Initiation section was entered as a result of a DCPC request from the driver's Continuation section. The DCPC channel assigned is available in Base Page Communications Area word CHAN.

Note that a DCPC channel obtained this way must be returned to RTE by a special procedure. The return procedure is discussed in the "Returning DCPC Channels to RTE" subsection.

Determining the DCPC Assignment Method

These alternate methods of obtaining a DCPC channel are more complex than the preferred method and should only be used by drivers that: 1) process a mixture of DCPC and non-DCPC operations and 2) cannot afford to tie up a DCPC channel during the non-DCPC operations.

It should also be noted that the Initiation section of the driver may have to use different methods to determine if a DCPC channel was assigned by IOC. Figure 3-7 summarizes these methods.

Regardless of the method used to obtain a DCPC channel, RTE records the assignment by putting the address of the EQT entry being assigned the DCPC channel into the appropriate DCPC Channel Assignment Word in the Interrupt Table.

If a DCPC channel is not available, the requesting EQT is set into a "waiting for DCPC" state. As soon as another driver releases a DCPC channel, the lowest-numbered EQT waiting for DCPC is assigned the DCPC channel, and its I/O request is initiated.

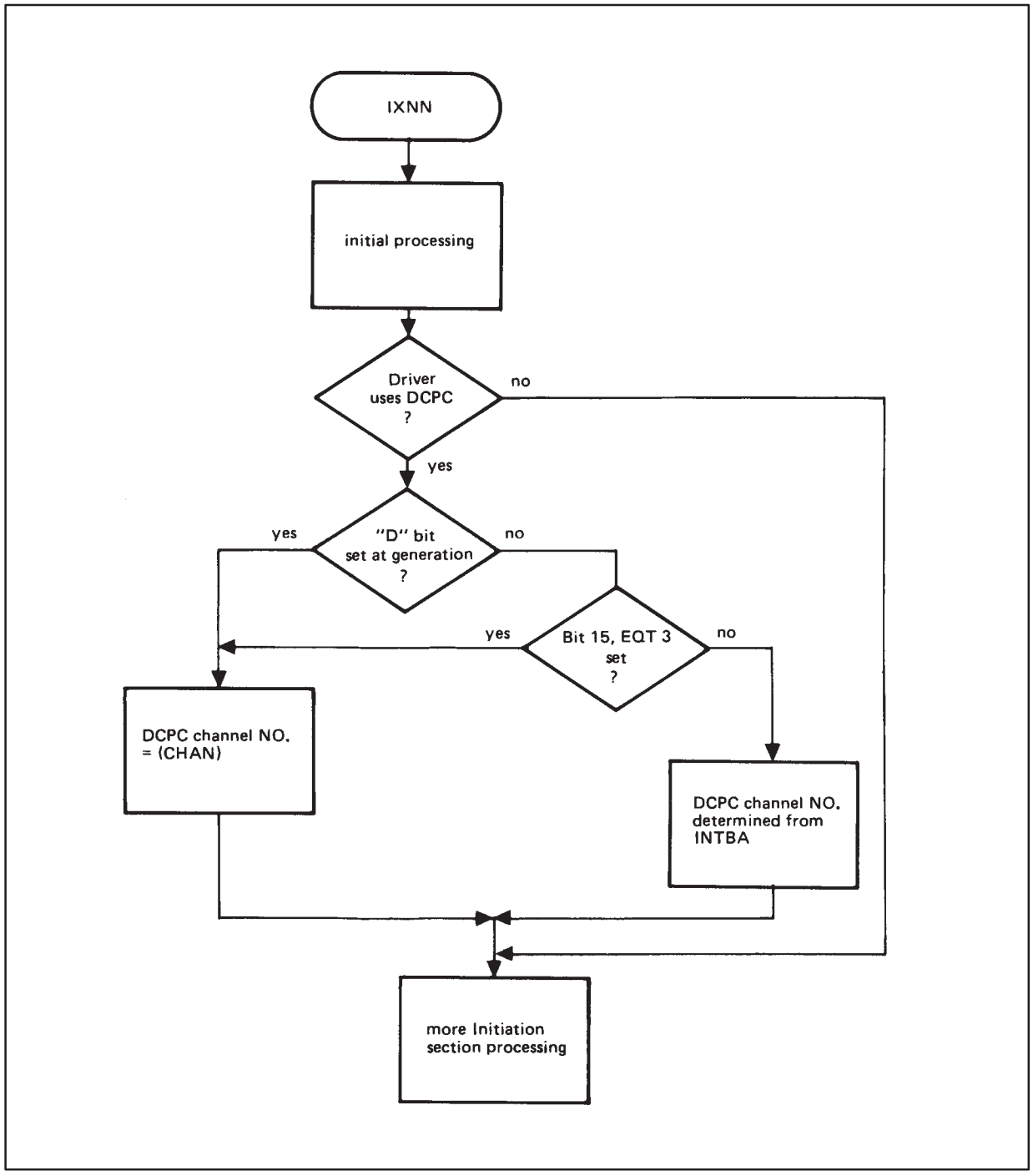

Figure 3-7. Determining DCPC Assignment

Returning DCPC Channels to RTE

As soon as a driver completes a DCPC transfer and the associated I/O request, the DCPC channel must be returned to the available pool so that it can be used by other drivers. This occurs in two different ways, depending on how the DCPC channel was assigned:

1. If the DCPC channel was assigned automatically (Preferred Method discussed previously), the DCPC channel is returned automatically by RTE when the driver makes its completion return to CIC. No special driver processing is required.

2. If the DCPC channel was assigned as the result of a specific request by the driver (Alternate Method discussed previously), the driver must explicitly inform RTE of this fact when the I/O request is completed. This is done by setting the sign bit in the A-Register on the completion return to CIC. This bit may be set at all times, even when the driver has not been assigned a DCPC channel. However, some extra system overhead is created if the sign bit is set when not required. Note that the sign bit is set in addition to the normal completion code, as illustrated below:

```
LDA  COMCD          Set A = completion code determined earlier
IOR  =B100000       Set sign bit to indicate dynamic DCPC assignment
JMB  Cxnn,I         Return to CIC
```

In either of the above cases, RTE implements the return of the DCPC channel by clearing the appropriate DCPC Channel Assignment Word in the Interrupt Table. DCPC Channel 1 Assignment Word (Interrupt Table word 1) is cleared if DCPC channel 1 was assigned to the driver; DCPC Channel 2 Assignment Word (Interrupt Table word 2) is cleared if DCPC channel 2 was assigned to the driver. No action is taken if a DCPC channel was not assigned to the driver.

In RTE-IV and RTE-6/VM, a driver may also return a DCPC channel assigned to it through an Initiation or a Continuation return. In each case, the A-Register having value 6 selects this deallocate DCPC option. IOC will deallocate the DCPC channel, if one was indeed allocated, and then proceed as for a normal Initiation or Continuation return. In addition, if a DCPC channel is deallocated from a Continuation return, IOC will check to see if another EQT is waiting for a DCPC channel. If so, IOC assigns the DCPC channel to that EQT and initiates its I/O request, thus increasing DCPC utilization.

Handling the DCPC Interrupt

An end-of-operation interrupt is generated by the DCPC hardware when a DCPC transfer is complete. Depending upon the nature of the device under control, the associated driver may or may not wish to recognize the DCPC interrupt.

If the driver does not require or use the DCPC completion interrupt, it can be disabled by issuing a clear control instruction (CLC) to the DCPC select code (6 or 7) after initializing the DCPC transfer. No further special processing is necessary.

If the driver uses the DCPC completion interrupt, some special processing must be included in the driver to ensure that the completion interrupt occurs only at the correct time in systems using a Privileged Interrupt card. These systems are described more fully in Chapter 4.

The following potential problem exists: In systems using a Privileged Interrupt card, the interrupt system is always ON, even when a driver is executing. It is therefore possible that a driver

using DCPC could start the DCPC transfer and be interrupted by its own DCPC completion interrupt before it has a chance to complete the initiation procedure and return to IOC.

To eliminate this problem, a scheme has been designed to hold off the DCPC completion interrupt until the standard driver using DCPC completes the initiation procedure and returns to IOC. This scheme requires the cooperation of the standard driver utilizing DCPC and of both RTE and any privileged drivers present in the system, as follows: After disabling the interrupt system and initializing the DCPC transfer, the standard driver clears control on the DCPC select code (6 or 7) to inhibit the completion interrupt while the standard driver completes the initiation procedure. The standard driver also sets a flag (F in the interrupt table, Figure 3-6) to inform RTE that the standard driver actually needs the interrupt, and that RTE should re-enable the interrupt later, after the driver returns to IOC.

The flag is also used by privileged drivers. A privileged driver disables the DCPC completion interrupts upon entry so that the privileged driver will not be interrupted while processing the privileged interrupt. A privileged driver will re-enable a DCPC completion interrupt before exiting only if it is needed by a standard driver (as indicated by the flag being set).

Bit 15 of each DCPC Channel Assignment word in the Interrupt Table is used as the flag for the respective DCPC channel. If this flag is set, RTE and the privileged drivers will enable the DCPC interrupt on the correct DCPC channel at the appropriate time. No action is taken if the flag is not set.

The section of code listed below illustrates the special processing required when a standard driver uses the DCPC completion interrupt. Note that although this processing must be included in all drivers using DCPC, it need only be executed when the driver is operating in a privileged system. The type of system in which a driver is operating can be determined by examining base page word DUMMY. If DUMMY is 0, the system is non-privileged (that is, no Privileged Interrupt card is present); otherwise the system is privileged (a Privileged Interrupt card is present).

```
      CLF 0              Disable the interrupt system

      STC DCPC,C         Initiate transfer on DCPC channel

      CLA                Bypass section below if
      CPA DUMMY          DUMMY = 0 (non-privileged system)
      JMP X              and special processing not needed.

      CLC DCPC           Clear DCPC control to inhibit DCPC
      LDB INTBA          interrupt. Set B = address of the appropriate
      LDA CHAN           DCPC Channel Assignment word in the
      CPA =D7            Interrupt Table.

      LDA B,I            Set bit 15 of DCPC channel assignment entry
      IOR =B100000       equal to 1 as flag to system to turn DCPC
      STA B,I            interrupts back on later. Reenable the
      STF 0              interrupt system.

    X EQU *              Continue processing.
```

Intermixed DCPC and Non-DCPC Operations

Occasionally a driver may have a special requirement to intermix a series of non-DCPC operations with DCPC operations during the same I/O request. If it is necessary or desirable to retain assignment of the DCPC channel throughout the non-DCPC operation, the F bit should be cleared prior to beginning the non-DCPC operations. This prevents the system from re-enabling the DCPC completion interrupt when it is not desired. Note that this processing need only be done if the flag was previously set under the conditions discussed in the preceding paragraphs.

Driver Automatic "Up"

A driver has the capability of automatically returning its EQT entry and all associated LUs to the "up" state through a JMP instruction. For example, if a driver makes a not ready, parity error, end-of-information, or timeout return to the system, the system sets the associated LU and EQT entry into the "down" state. If the driver subsequently detects an interrupt (or timeout) entry that signals that the controller is now ready, it may return the EQT entry and associated LUs to the "up" state as follows:

```
JMP $UPIO
```

The device controller's EQT entry and all associated LUs are reset to the up (available) state by $UPIO. If an I/O request is pending, $UPIO restarts the request by entering the driver at the initiation entry point I*xnn*. If there are no requests pending, $UPIO goes to the dispatcher to start the next program.

Powerfail Processing

When an RTE system is generated, the user has the option of including DVP43, the powerfail/auto-restart driver, and AUTOR, the automatic restart program. If DVP43 is not included, the system executes a HALT 4 when power is restored to the computer.

If the powerfail/auto-restart modules are included in the generation, they enable the system to recover automatically from a power failure. Powerfail/auto-restart processing can be divided into three parts:

1. The power down sequence.
2. The power up sequence.
3. The sequence required to restart any I/O transfers that were in progress when the powerfail occurred. A driver has the option of restarting its own I/O or of letting the system restart it from the beginning of the request. These two alternatives are discussed below.

Power Down Sequence

When a powerfail occurs, a powerfail interrupt is generated and DVP43 is entered to process the interrupt. In the brief period of time available before the system becomes completely inoperable, DVP43 performs the following steps to save the state of the machine:

1. Stops all DCPC transfers.
2. Saves all user-accessible registers (A, B, X, Y, E, O ...).
3. Saves the status of the memory protect fence. Also saves the status of the Dynamic Mapping System (DMS) in systems that include the DMS feature.
4. Saves all map registers (System Map, User Map, and the two DCPC maps). This step is performed in systems with Dynamic Mapping only.

DVP43 then executes a HLT 4 instruction before power fails completely.

Power Up Sequence

When power is restored to the computer, an interrupt is generated and DVP43 is re-entered to process the interrupt. DVP43 performs the following steps to restart the system:

1. Sets a software flag to prevent resaving the state of the machine if a subsequent power failure occurs before the system is completely restored.
2. Re-enables the powerfail hardware.
3. Restores the state of the memory protect fence. Also restores all map registers and the status of the Dynamic Mapping System in systems that include the DMS feature.

4. Saves the time of the powerfail.

5. Finds the powerfail EQT entry (that is, the EQT entry associated with DVP43) and sets up a very short timeout on this EQT entry by setting EQT entry word 15 (the timeout clock) to −1. This causes DVP43 to be re-entered after one tick of the real-time clock. DVP43 can then begin to restart any I/O transfers that were in progress at the time of the power failure.

6. Restarts the real-time clock.

7. Restores all user-accessible registers.

8. Clears the software flag that was set in step "1", so that the state of the machine will be saved as usual on any subsequent power failures.

9. Returns to the suspended process (that is, the process that was in operation when the power-fail occurred) at the point of interruption.

Restart I/O Sequence

As soon as the powerfail EQT entry times out, DVP43 is entered again because it previously set the "driver processes timeout" bit. DVP43 now attempts to restart any I/O transfers that were in progress at the time of the powerfail by performing the following steps:

1. Makes the following checks for each I/O controller:

   a. Checks bits 14 and 15 of EQT entry word 5. The value of bits 14 and 15 indicate whether the I/O controller was "down" or "busy" at the time of the power failure.

   b. Checks bit 13 of EQT entry word 4 to see if the driver associated with the EQT entry is prepared to process a powerfail/auto-restart entry. Drivers that are prepared to process powerfail/auto-restart entries will have previously set bit 13 to one. Otherwise this bit is zero. Note that the system never clears bit 13, so a driver only needs to set it once.

   c. Checks to see if any EQT entries are currently waiting for a DCPC channel.

2. Depending upon the above information, one of the following three actions is taken for each controller or device in the system:

   Case 1. Controller (EQT entry) busy and "driver processes timeout" bit set:

   If the controller was reading or writing data when the powerfail occurred and the driver is designed to handle powerfail, the driver has the responsibility to recover from the powerfail in the best possible manner. The system simply sets bit 15 of EQT entry word 5 to 1 to indicate that a powerfail has occurred and enters the driver at the initiation entry point I*xnn*.

   Case 2. Controller waiting for a DCPC channel.

   If the controller was waiting for a DCPC channel when the power failure occurred, no action is taken. The I/O transfer will be initiated as usual when a DCPC channel is released by another driver.

Case 3. All other EQT entries.

For all EQT entries not falling under Case 1 or Case 2 above, DVP43 makes a call to $UPIO to up the EQT entry and all associated LUs. (See the "Driver Automatic Up" subsection of this manual.) $UPIO restarts any I/O requests that were in progress (EQT entry was busy) or pending (EQT entry or LU was down) at the time of the power failure. This is done by resetting the parameters of the original call into the EQT entry and reentering the driver at the initiation point I*xnn*.

After the above action is taken for each I/O controller in the system, an HP-supplied program called AUTOR (auto-restart) is scheduled. AUTOR sends the time of power failure to all user consoles on the system (thereby re-enabling all terminals).

AUTOR is written in FORTRAN, and the source file is supplied so that it can be easily modified for site-specific applications.

Program Scheduling by Drivers

Occasionally some I/O applications may require that a driver schedule a program to perform a certain task. The system list processor, $LIST, has several calls available that provide drivers with this capability. These calls are illustrated below. All of these calls cause a program to be scheduled. They differ only in the format of the calling sequence and in the type of information that each call may specify to be stored in the ID segment of the program to be scheduled. In a session environment, drivers schedule programs outside of session, so files that they access must be on cartridges mounted outside of session.

Method 1. Puts five parameters in the ID segment and then schedules the program.

```
            EXT $LIST
               .
               .
            JSB $LIST
            OCT 701
            DEF RTN          Return point.  Must be immediately after the call.
            DEF PNAME        Address of 3-word array containing program name.
            DEF P1           Addresses of up to five optional parameters to be
            DEF P2           placed in program's ID segment.
            DEF P3
            DEF P4
            DEF P5
RTN            .             Return point.  Must be located immediately after call.
               .             (See below for error status return in A/B-Registers.)
               .
PNAME       ASC 3,XXXXX      Name of program to be scheduled.
P1          OCT A            Up to five optional parameters to be copied to the
P2          OCT B            program's ID segment (temporary storage area)
P3          OCT C            prior to scheduling it.
P4          OCT D
P5          OCT E
```

This call causes the system to copy whatever number (1-5) of optional parameters are supplied into the temporary storage area of the ID segment of the program whose ASCII name is contained in the variable PNAME. The system then schedules the program to run at its own priority.

Caution You must pass at least one parameter (P1). The driver will cause a system crash if it calls $LIST with no parameters.

Method 2. Same as Method 1, except that the ID segment address rather than the program name is supplied. Note that in some RTE systems a driver may not be able to search the ID Segment Table for a program's ID segment address. It is therefore recommended that drivers scheduling programs do so by specifying the program's name (function code 701 call to $LIST), rather than the ID segment address.

Note that if function codes 001 or 601 are used, the driver should schedule only permanently loaded programs because RTE re-uses the ID segment of a temporary program.

```
          EXT $LIST
            .
            .
            .
          JSB $LIST
          OCT 001
          DEF RTN          Return point.  (Must be immediately after call.)
          OCT IDADR        ID segment address of program to be scheduled.
          DEF P1           Address of up to five optional parameters to be
          DEF P2           placed in program's ID segment.
          DEF P3
          DEF P4
          DEF P5
RTN         .              Return point.  Must be located immediately after call.
            .              (See below for error status return in A/B-Registers)
            .
P1        OCT A            Up to five optional parameters to be placed in
P2        OCT B            program's ID segment (temporary storage area)
P3        OCT C            prior to scheduling it.
P4        OCT D
P5        OCT E
```

This call causes the system to place whatever number (1-5) of optional parameters are supplied into the temporary storage area of the ID segment specified by IDADR. The system then schedules the program associated with the ID segment to run at its own priority.

Method 3. Puts a value into the "B-Register at suspension" word in the ID segment and then schedules the program. This call can be used to set the B-Register to point to a scheduling parameter storage area. The scheduled program can then recover the parameters via a call to subroutine RMPAR. The driver should make sure that the parameters are placed in a memory area that is mapped with the user program.

```
        EXT $LIST
            .
            .
            .
        JSB $LIST
        OCT 601
        OCT IDADR       ID segment address of program to be scheduled.
        OCT BVAL        Value to be put into "B-Register at suspension" word.
RTN         .           Return is always made to here by $LIST.
            .           (See below for error status return in A/B-Registers.)
            .
```

This call causes the system to place the value BVAL into the "B-Register at suspension" word of the ID segment specified by IDADR. If this value is an address that points to a set of scheduling parameters, the program can recover the parameters by making a call to subroutine RMPAR. The system then schedules the program associated with the ID segment to run at its own priority.

Error Conditions:

When $LIST returns from any of the program schedule calls described above, the A and B-Registers indicate whether or not the program was successfully scheduled, as follows:

If A = 0 The program was successfully scheduled. The B-Register contains the ID segment address of the scheduled program,

If A ≠ 0 The program could not be scheduled. The B-Register indicates the reason, as follows:

B = 3 Illegal status (program not dormant)

B = 5 No such program.

Determination of Operating System Environment

There are times when it may be necessary for a driver to know the operating system within which it is executing. The system entry point $OPSY provides this and other information in the form of a one-word table, as illustrated in Table 3-1.

Table 3-1. $OPSY Word Format

System	$OPSY Value	Bit 15	Bit 3	Bit 2	Bit 1	Bit 0
		1=RTE	Type	0=Memory-Based 1=Disk-Based	0=No DMS 1=DMS	0= 64-Word Disk 1=128-Word Disk
RTE-M/I	−7	1	1	0	0	1
RTE-M/II	−15	1	0	0	0	1
RTE-M/III	−5	1	1	0	1	1
RTE-II	−3	1	1	1	0	1
RTE-III	−1	1	1	1	1	1
RTE-IV	−9	1	0	1	1	1
RTE-IVE	−13	1	0	0	1	1
RTE-6/VM	−17	1	1	1	1	1

$OPSY can be referenced by loading it into a register and testing the appropriate bits. This technique is illustrated below:

```
EXT $OPSY
     .
     .
     .
LDA $OPSY    Access $OPSY information.
AND MASK     Isolate appropriate bits.
     .
     .       Take appropriate action.
     .
```

In Dynamic Mapping System ($OPSY bit 1 = 1) mode, it may also be necessary to determine whether the System Map or User Map is currently enabled. This can be done in a driver by accessing the status of the Dynamic Mapping System via an RSA instruction and looking at bit 12. Bit 12 is 0 if the System Map is enabled, and 1 if the User Map is enabled. The following code illustrates this procedure.

```
RSA          Access Dynamic Mapping System Status.
ALF,SLA      Position Bit 12 into Bit 0.
             ; System Map enabled?
JMP USER     No, User Map Enabled.
JMP SYSTM    Yes, System Map Enabled.
```

Subroutines for Special Mapping Function (DMS Systems Only)

By using the Dynamic Mapping System (DMS) feature of the HP 1000 M/E/F-Series of computers, RTE provides the capability for addressing memory configurations larger than 32K words. This is accomplished by translating memory addresses through one of four "memory maps". A memory map consists of a set of hardware registers. These registers provide the interface between memory addresses used by programs (logical memory addresses) and actual memory addresses (physical memory addresses). There are four distinct maps: the System Map, the User Map, and the two DCPC maps. The DCPC Maps are loaded (that is, set up) by the system as necessary to describe the logical memory configuration required by the currently executing program or DCPC transfer.

Prior to entering a driver to process an I/O request, RTE loads and enables the correct map (System or User) needed to describe the buffer of the request. The driver can access the buffer through the request buffer address word in the EQT without having to consider which map contains the buffer.

Certain drivers may, however, need to access a buffer in addition to the primary request buffer. If a driver is entered with the System Map enabled (for example, when the primary request buffer is in System Available Memory or System Common), it must access the second buffer through the User Map if the second buffer is located in the program making the I/O request. In addition, the driver must load the User Map because the current contents of the map may or may not describe the desired program.

Subroutine $XDMP can be called by standard drivers to reload the User Map to describe the program containing the second buffer. The driver uses $XDMP as follows:

1. Save the contents of the current User Map.
2. Load the A-Register with the ID Segment address of the program containing the second buffer.
3. Call $XDMP (JSB $XDMP) to load the User Map for the program specified in the A-Register.
4. Check for an error return. $XDMP returns with the A-Register set to zero if the specified program was not in memory and the User Map could not be set up.
5. If no error was detected, access the second buffer through the newly loaded User Map.
6. After all accesses to the second buffer have been made, restore the User Map to its original contents and continue with normal operation.

Notice that the use of $XDMP requires that the driver know the ID Segment address of the program making the I/O request. Because the driver cannot be sure that this ID Segment address is available on base page or in the EQT, the program making the I/O request should cooperate with the driver and pass its ID Segment address. This could be done by using one of the optional parameters and allowing the driver to retrieve the address through the appropriate optional parameter word in the EQT or by including the address in one of the words of the primary request buffer.

The recommended procedure for using $XDMP and accessing the second buffer is somewhat different in RTE-III, RTE-IV and RTE-6/VM. Each of these procedures is described in a sepa-

rate section below. Note that subroutine $XDMP can be called by standard RTE drivers only. Privileged drivers wishing to perform the same function must use subroutine $PVMP, which is described in Chapter 4 of this manual.

Mapping in RTE-III and RTE-M/III

Any standard driver operating in RTE-III may use subroutine $XDMP to perform the memory map switching discussed above. The driver first saves the current state of the User Map registers and the Dynamic Mapping System, and then calls $XDMP to reload the User Map registers to describe the user program. The driver can then enable the User Map and access the memory described by it. After all accesses have been made, the driver re-enables the System Map and restores the original state of the User Map registers before continuing with its normal processing under the System Map.

Note that subroutine $XDMP need only be used to reload the User Map registers when the driver is entered with the System Map enabled. The calling sequence is as follows:

```
        EXT $XDMP
           :
        RSA                 Get Dynamic Mapping System (DMS) status.
        ALF,SLA             Position bit 12 into bit 0.
                            ; Is System Map Enabled?
        JMP USER            No, so do not need to execute code below.
            .
            .              Normal driver processing under System Map.
            .
        RSA                 Get Dynamic Mapping System (DMS) status.
        RAL,RAL             Position current status in upper bits.
        STA DMSST           Save status for later.

        LDA MAPAD           Set A = address of User Map storage area.
        IOR SIGN            Set sign bit indicating STORE Map in memory.
        USA                 Save current User Map in memory for later.

        LDA IDADR           Get ID address of program that contains buffer.
        JSB $XDMP           Call $XDMP to set up User Map for this program.
        SZA,RSS             Check for error return.
        JMP ERROR           Error exists, go handle it.

        UJP CONT            No errors.  Enable new User Map and continue.

CONT        .
            .              Process buffer under new User Map.
            .
        SJP NEXT            Re-enable System Map.
NEXT    LDA MAPAD           Access address of User Map storage area.
        USA                 Restore original contents of User Map.
        JRS DMSST NXT       Restore original DMS status (that is, System Map).

NXT         .
            .              Proceed with normal processing under System Map.
            .
MAPAD   DEF MAP             Address of User Map storage area.
```

```
MAP     BSS 32              User Map storage area.
SIGN    OCT 100000
IDADR   BSS 1               Storage for ID segment address.
DMSST   BSS 1               Temporary storage for DMS status.
```

Remember that any driver using $XDMP must save the original contents of the User Map before calling $XDMP. It must also restore the original User Map before continuing with its normal operation under the System Map. Also remember that if the driver accesses the memory described by the User Map by enabling the map, it must save and restore the DMS status in addition to the original contents of the User Map. The example above illustrates this procedure.

Note The driver could also access the buffer in the user program through a series of cross-map loads and stores without actually enabling the User Map. This is in fact the recommended procedure for using $XDMP in an RTE-IV or RTE-6/VM system, and use of it by a driver would allow the same driver to be used in either type of system.

Mapping in RTE-IV and RTE-6/VM

Drivers in RTE-IV and RTE-6/VM may be located in one of the Driver partitions or in the System Driver Area of memory. If a driver resides in a driver partition, RTE automatically includes the correct driver partition as it loads and enables the correct map (System or User) prior to entering the driver. Drivers generated into a driver partition are always in the same map as their I/O buffer(s).

The other location for a driver in RTE-IV and RTE-6/VM is the System Driver Area (SDA) of memory. Drivers are placed in SDA by specifying either “M” or “S” in the EQT entry definition phase during system generation. The “S” option specifies that the driver will be entered under the appropriate map (System or User) depending on the location of the I/O request buffer. The “M” option specifies that the driver will always be entered under the System Map regardless of the location of the I/O buffer.

The “S” and “M” options also differ in the types of checks performed by RTE prior to entering the driver. If the “S” option is selected and the I/O buffer is in the user program, RTE will check to see if SDA is included in the program’s logical address space (map). If not, RTE aborts the program with an IO11 error, “Type 4 program made an unbuffered I/O request to a driver that did not do its own mapping”. If the “M” option is selected, RTE will not check to see if the I/O request buffer and SDA are in the same logical address space; rather, RTE enters the driver under the System Map, assuming that the driver will perform any needed mapping. This is why the “M” option is sometimes called the “driver does its own mapping” option.

Thus drivers in SDA with the S option are always in the same map as their I/O buffer(s).

Drivers using the M option are always in the System Map. However, for non-privileged drivers, RTE-6/VM will always reload the User Map registers to describe the requesting program if the I/O buffers are located in the program.

In RTE-6/VM $XDMP never needs to be used.

The procedure for using $XDMP is as follows: The driver first saves the current contents of the User Map registers and then calls $XDMP to reload the User Map to describe the desired program. The driver can then use a series of cross-map loads and stores to access the buffer described by the User Map. Note that the User Map should not be enabled because drivers in SDA are not necessarily included in all user maps. After all accesses have been made, the driver restores the original state of the User Map registers before continuing with its normal processing.

The following example shows how an SDA driver operating under the System Map might use the $XDMP subroutine to access a buffer in a user program.

```
        EXT $XDMP
            .
            .               Normal SDA driver processing under System Map.
            .
        LDA MAPAD           Set A = address of User Map storage area.
        IOR SIGN            Set sign bit indicating STORE Map in memory.
        USA                 Save current User Map in Memory for later.

        LDA IDADR           Get ID Segment address of program that contains
                            buffer.
        JSB $XDMP           Call $XDMP to set up User Map for this program.
        SZA,RSS             Check for error return.  If A-Register = 0,
                            specified program was not found in memory and
                            user map is not set up.
        JMP ERROR           Error exists, go handle it.
            .
            .               No errors.  Access buffer via series of
            .               cross-map loads and stores, since System
            .               Map is still enabled.
            .
        LDA MAPAD           Access address of User Map storage area.
        USA                 Restore original contents of User Map.
            .
            .               Proceed with normal processing under System Map.
            .

MAPAD   DEF MAP             Address of User Map storage area.
MAP     BSS 32              User map storage area.
SIGN    OCT 100000
IDADR   BSS 1               Storage for ID segment address.
```

Remember that the driver using this routine must save the current contents of the User Map registers before calling $XDMP, and must restore the registers to their original value after all accesses have been made to the buffer. The example above illustrates this procedure.

Obtaining the Subchannel

There are two ways to obtain the subchannel number. The first is to pull it out of the EQT. The second, for RTE-6/VM only, is to use the system subroutine $SUBC. $SUBC will return the subchannel number in the low bits of the A-Register.

```
EXT $SUBC
    :
JSB $SUBC
```
subchannel # returned in the A-Register

Operating System Trap Cell Instructions

In RTE-6/VM, for E- and F-Series computers only, there are four instructions available for handling interrupts, excluding powerfail. These instructions replace the JSB LINK,I that is in the trap cell in RTE-M, -II, -III and IV systems.

The four interrupt-handling instructions perform some of the operations handled by the central interrupt processor, thereby speeding up the interrupt processing and reducing overhead.

The four instructions speed up interrupt processing in two ways. First, the driver does not need to execute code to determine what type of interrupt occurred (memory protect, DCPC, TBG, or normal device). Second, the operations are performed in microcode, allowing faster execution of the functions.

Operating system trap cell instructions do not have to be generated into the system. On the E- and F-Series computers, the system places them in the trap cells for you.

Sample Standard RTE Driver

The sample driver illustrated in Figure 3-8 demonstrates some of the principles involved in writing a standard I/O driver for the RTE operating system. Note that this driver is for tutorial purposes only and is not one of the drivers supplied with the system.

```
               ASMB,Q
                      NAM DVR70   **STANDARD RTE DRIVER EXAMPLE **
*
*
                      ENT I.70,C.70
*
*
* DRIVER 70 OPERATES UNDER THE CONTROL OF THE I/O CONTROL (IOC)
* AND THE CENTRAL INTERRUPT CONTROL (CIC) MODULES OF RTE.
* THIS DRIVER IS RESPONSIBLE FOR CONTROLLING OUTPUT
* TRANSMISSION TO A 16 BIT EXTERNAL DEVICE.
* I.70 IS THE ENTRY POINT FOR THE *INITIATION* SECTION
* AND C.70 IS THE ENTRY POINT FOR THE *CONTINUATION/COMPLETION*
* SECTION.
*
* NOTE THAT THIS DRIVER DOES NOT PROCESS TIME-OUTS OR
* POWER FAIL.  THESE PROCEDURES ARE LEFT ENTIRELY UP TO
* THE SYSTEM.
*
* REMEMBER THAT RTE SETS THE ADDRESSES OF EACH WORD OF
* THE 15 WORD EQT ENTRY FOR THE DEVICE BEING SERVICED INTO
* THE BASE PAGE COMMUNICATIONS AREA ON EACH ENTRY TO THE
* DRIVER.
* THIS DRIVER REFERENCES THESE ADDRESSES THROUGH VARIABLES
* EQT1 THROUGH EQT15.
*
* **********************
* * INITIATION SECTION *
* **********************
*
* THE INITIATION SECTION IS CALLED FROM I/O CONTROL (IOC) TO
* INITIALIZE A DEVICE AND INITIATE AN OUTPUT OPERATION
*
* THE CALLING SEQUENCE FOR THE INITIATION SECTION IS:
*
*              (SET A = SELECT CODE OF I/O DEVICE)
*         P    JSB I.70
*         P+1  (RETURN POINT)
*
*         ON RETURN, A REGISTER INDICATES STATUS, AS FOLLOWS:
*
*               A = 0,   OPERATION SUCCESSFULLY INITIATED
*               A NOT 0, OPERATION REJECTED FOR THE FOLLOWING
*                        REASON:
*
*                        A = 1 = ILLEGAL READ REQUEST
*                        A = 2 = ILLEGAL CONTROL REQUEST
*
* (NOTE, HOWEVER, THAT A “CLEAR” CONTROL REQUEST FROM THE
*  SYSTEM WILL BE PROCESSED BY THE DRIVER, AS REQUIRED.)
*
```

Figure 3-8. Standard RTE Driver Example

```
* **********************************
* * CONTINUATION/COMPLETION SECTION *
* **********************************
*
* THE CONTINUATION/COMPLETION SECTION IS CALLED BY CENTRAL
* INTERRUPT CONTROL (CIC) TO CONTINUE OR COMPLETE AN OPERATION WHEN
* AN INTERRUPT IS DETECTED ON THE DEVICE
*
* THE CALLING SEQUENCE FOR THE COMPLETION SECTION IS:
*
*              (SET A = SELECT CODE OF I/O DEVICE)
*         P    JSB C.70
*         P+1  COMPLETION RETURN
*         P+2  CONTINUATION RETURN
*
*         ON RETURN, A & B REGISTERS INDICATE STATUS, AS FOLLOWS:
*
*               ON A COMPLETION RETURN:
*
*               A = 0, SUCCESSFUL COMPLETION, WITH
*                      B = NUMBER OF WORDS TRANSMITTED
*
*               A = 2, TRANSMISSION ERROR DETECTED
*
*               ON A CONTINUATION RETURN, THE REGISTERS ARE
*               MEANINGLESS
*
* RECORD FORMAT:
*
*     THIS DRIVER PROVIDES A 16 BIT BINARY WORD
*     TRANSFER ONLY.
*
*     **********************
*     * INITIATION SECTION *
*     **********************
  00000 000000  I.70  NOP           ENTRY FROM IOC
*
  00001 000100R       JSB SETIO     CONFIGURE I/O INSTRUCTNS FOR DEVICE

  00002 001665        LDA EQT6,I    GET CONTROL WORD OF REQUEST, AND
  00003 000117R       AND =B3       ISOLATE THE REQUEST TYPE
*
  00004 000115R       CPA =B1       IF REQUEST IS FOR INPUT
  00005 000000R       JMP I.70,I    THEN REJECT (A=1=ILLEGAL READ)
  00006 000116R       CPA =B2       IF REQUEST IS FOR OUTPUT
  00007 000017R       JMP D.X1      THEN GO PROCESS WRITE REQUEST
*
* CONTROL REQUEST CHECK IF IT IS A "CLEAR" CONTROL REQUEST
* IF SO, ASSUME IT WAS ISSUED BY SYSTEM, CLEAR DEVICE, AND RETURN
*
  00010 001665        LDA EQT6,I    ACCESS CONTROL WORD
  00011 000124R       AND =B3700    ISOLATE SUBFUNCTION
  00012 002002        SZA           "CLEAR" REQUEST?
  00013 000015R       JMP REJCT     NO, SO REJECT REQUEST AS ILLEGAL
```

Figure 3-8. Standard RTE Driver Example (continued)

```
*
  00014 106700  I.0   CLC SC        YES, CLEAR DEVICE AND RETURN
*
* REQUEST ERROR - CAUSE REJECT RETURN TO IOC
*
  00015 000116R REJCT LDA =B2       SET A=2=ILLEGAL CONTROL REQUEST
  00016 000000R       JMP I.70,I    AND RETURN
*                   (A=2=ILLEGAL CONTROL REQUEST)
*
* WRITE REQUEST PROCESSING
*
  00017 001666  D.X1  LDA EQT7,I    GET REQUEST BUFFER ADDRESS
  00020 001670        STA EQT9,I    AND SET IT AS CURRENT ADDRESS
  00021 001667        LDA EQT8,I    GET REQUEST BUFFER LENGTH
  00022 003004        CMA,INA       MAKE NEGATIVE AND
  00023 001671        STA EQT10,I   SAVE AS REMAINING BUFFER LENGTH
  00024 002002        SZA           IS BUFFER LENGTH = 0?
  00025 000031R       JMP D.X3      NO, PROCESS AS USUAL
  00026 000120R       LDA =B4       YES, MAKE IMMEDIATE COMPLETN RETURN
  00027 006400        CLB           SET TRANSMISSION LOG = 0 INTO B
  00030 000000R       JMP I.70,I    AND RETURN (A=4=IMMED. COMPLETION)
*
* CALL THE CONTINUATION/COMPLETION SECTION TO WRITE FIRST WORD
*
  00031 000114R D.X3  LDA P2        ADJUST RETURN ADDRESS SO WILL
  00032 000036R       STA C.70      RETURN HERE (INITIATION SECTION)
  00033 000047R       JMP D.X2      GO TO COMPLETION SECTION
*
  00034 002400  IEXIT CLA           NOW RETURN TO IOC WITH
  00035 000000R       JMP I.70,I    OPERATION INITIATED (A = 0 = OK)
*
*     **********************************
*     * CONTINUATION/COMPLETION SECTION *
*     **********************************
*
  00036 000000  C.70  NOP           CONTINUATION/COMPLETION ENTRY POINT
*
  00037 000100R       JSB SETIO     CONFIGURE I/O INSTRUCTIONS
*
  00040 001660        LDA EQT1,I    CHECK FOR SPURIOUS INTERRUPT
  00041 000126R       AND =B77777   ISOLATE I/O REQST LIST PTR (15 BITS)
  00042 002002        SZA           IS REQST IN PROGRESS?
  00043 000047R       JMP D.X2      YES, GO PROCESS REQUEST
*
  00044 001774        STA EQT15,I   NO SPURIOUS INTRUPT-ZERO TO CLOCK
  00045 000036R       ISZ C.70      ADJUST RETURN TO P+2 (CONTINUATION)
  00046 000036R       JMP C.70,I    MAKE CONTINUATION RETURN TO CIC
*
  00047 002400  D.X2  CLA           IF CURRENT BUFFER LENGTH  = 0,
  00050 001671        CPA EQT10,I   THEN GO TO STATUS
  00051 000063R       JMP I.3       SECTION.  (I.E., TRANSFER DONE NOW)
*
  00052 001670        LDB EQT9,I    GET CURRENT BUFFER ADDRESS
```

Figure 3-8. Standard RTE Driver Example (continued)

```
*
  00053 001670        ISZ EQT9,I    ADD 1 FOR NEXT WORD
  00054 000001        LDA B,I       GET WORD TO BE WRITTEN TO DEVICE
  00055 001671        ISZ EQT10,I   INCREMENT WORD COUNT ALSO
  00056 000000        NOP           IGNORE P+1 SKIP IF LAST WORD
*
  00057 102600  I.1   OTA SC        OUTPUT WORD TO INTERFACE
  00060 103700  I.2   STC SC,C      TURN DEVICE ON
*
  00061 000036R       ISZ C.70      ADJUST RETURN TO P+2 (CONTINUATION)
  00062 000036R       JMP C.70,I    MAKE CONTINUATION RETURN
*
* STATUS AND COMPLETION SECTION
*
  00063 102500  I.3   LIA SC        GET STATUS WORD FROM DEVICE
  00064 000121R       AND =B77      STRIP OFF UNUSED BITS
  00065 000001        STA B         SAVE IN B TEMPORARILY
  00066 001664        LDA EQT5,I    REMOVE PREVIOUS STATUS
  00067 000127R       AND =B177400  BITS IN EQT WORD 5
  00070 000001        IOR B         OR IN NEW BITS
  00071 001664        STA EQT5,I    AND RESET INTO EQT WORD 5
*
  00072 002400        CLA           SET A = 0 = OK RETURN CODE
  00073 000120R       CPB =B4       ERROR STATUS BIT ON?
  00074 000116R       LDA =B2       YES, SET A = 2 = ERROR RETURN
*
  00075 001667        LDB EQT8,I    SET B = TRANSMISSION LOG
*
  00076 106700  I.4   CLC SC        CLEAR DEVICE CONTROLLER
*
  00077 000036R       JMP C.70,I    MAKE COMPLETION RETURN TO CIC
*
*
*     ********************
*     * SUBROUTINE SETIO *
*     ********************
*
* SUBROUTINE <SETIO> CONFIGURES ALL I/O INSTRUCTIONS IN DRIVER
*
  00100 000000  SETIO NOP           ENTRY POINT (A = SELECT CODE)
*
  00101 000113R       IOR LIA       COMBINE LIA WITH I/O
  00102 000063R       STA I.3       SELECT CODE AND SET IN CODE
*
  00103 000122R       ADA =B100     CONSTRUCT OTA INSTRUCTION
  00104 000057R       STA I.1
*
  00105 000123R       ADA =B1100    CONSTRUCT STC,C INSTRUCTION
  00106 000060R       STA I.2
*
  00107 000125R       XOR =B5000    CONSTRUCT CLC INSTRUCTION
  00110 000014R       STA I.0
  00111 000076R       STA I.4
*
  00112 000100R       JMP SETIO,I   RETURN
```

Figure 3-8. Standard RTE Driver Example (continued)

```
*
*
*     *************
*     * DATA AREA *
*     *************
*
* CONSTANT AND STORAGE AREA
*
        000000  A      EQU 0          A-REGISTER
        000001  B      EQU 1          B-REGISTER
*
        000000  SC     EQU 0          DUMMY I/O SELECT CODE NUMBER
  00113 102500  LIA    LIA 0          CODE FOR LIA INSTRUCTION
  00114 000033R P2     DEF IEXIT-1    RETURN POINT IN INITIATION SECTION
*
* ** BASE PAGE COMMUNICATIONS AREA DEFINITIONS **
*
        001650  .      EQU 1650B
*
        001660  EQT1   EQU .+8
        001661  EQT2   EQU .+9
        001662  EQT3   EQU .+10
        001663  EQT4   EQU .+11
        001664  EQT5   EQU .+12
        001665  EQT6   EQU .+13
        001666  EQT7   EQU .+14
        001667  EQT8   EQU .+15
        001670  EQT9   EQU .+16
        001671  EQT10  EQU .+17
        001672  EQT11  EQU .+18
        001771  EQT12  EQU .+81
        001772  EQT13  EQU .+82
        001773  EQT14  EQU .+83
        001774  EQT15  EQU .+84
*
*
  00115 000001
  00116 000002
  00117 000003
  00120 000004
  00121 000077
  00122 000100
  00123 001100
  00124 003700
  00125 005000
  00126 077777
  00127 177400
                       END
```

Figure 3-8. Standard RTE Driver Example (continued)

4

Writing Privileged RTE Drivers

Introduction

Peripheral devices that are synchronous in nature, that generate interrupts at very high rates, or that cannot tolerate long latency time, need special attention in an RTE system. Such devices cannot be controlled by standard RTE drivers on a word-by-word transfer basis, because this method cannot guarantee that the interrupts generated by such device controllers will be processed within the required response time. There are two reasons why the response time may be exceeded:

1. An interrupt is not recognized immediately by RTE if the interrupt system is disabled at the time the interrupt occurs. For example, this happens if the interrupt occurs while a standard driver is processing a previous interrupt or while RTE itself is executing.

2. Once an interrupt is recognized, the system overhead required to direct the interrupt to the appropriate driver for processing may be too long.

One way to guarantee a fast interrupt response time is to utilize DCPC transfers for synchronous and high-speed devices. The special DCPC hardware allows the transfer to occur simultaneously with other RTE operations, thereby eliminating the above problems.

However, DCPC transfers do not allow the driver to perform processing that might be required on each data word as it is transferred. For example, it might be necessary to check a parity bit on each word as it is received from the device. Thus, a special interrupt processing method is needed for a high-speed or synchronous device that requires driver interaction on each data word transferred. This interrupt processing method must have the following properties:

a. The ability to recognize interrupts immediately, regardless of what other RTE operation is in progress.

b. A means to eliminate the system overhead associated with processing an interrupt.

c. Driver interaction on each data word transferred.

Privileged interrupt processing was specifically designed to meet these criteria. This method requires that a special I/O card, known as the Privileged Interrupt Fence card, be present in the system. The Privileged Interrupt Fence card is inserted in the computer such that it physically separates the I/O cards into two groups. All devices whose I/O cards are in lower-numbered

(higher priority) select codes are known as privileged devices; these are the high-speed and synchronous devices that require driver interaction on each word transferred. The I/O cards of all other devices in the system are placed in higher-numbered (lower priority) select codes and are known as non-privileged devices.

Systems with Privileged Interrupt Fence cards are referred to as privileged systems, and a special type of RTE driver (known as a privileged driver) is required for each privileged controller present in the system. Standard RTE drivers are used for the remaining non-privileged devices.

The Privileged Interrupt Fence card can be almost any standard I/O card that contains the normal control and flag flip-flop circuitry, for example the HP 12620A Breadboard, HP 12966A BACI, HP 12531D Terminal, or HP 12566C Microcircuit Interface Cards. Because of the position of the Privileged Interrupt Fence card in the I/O priority chain, setting of the control flip-flop on the card holds off all interrupts from the non-privileged device controllers, while at the same time allowing the privileged device controllers to interrupt.

When a Privileged Interrupt Fence card is present in the system and a non-privileged interrupt occurs (or when the system is requested to perform some function via an EXEC call or operator request), RTE performs the following functions before entering the standard driver (or system routine) to process the interrupt:

1. Disables the interrupt system and saves the state of the machine.

2. Sets the control flip-flop on the Privileged Interrupt Fence card to hold off any further non-privileged interrupts.

3. Disables the DCPC completion interrupts. These interrupts are not held off by the Privileged Interrupt Fence card because the DCPC completion interrupts occur on the highest priority select codes (6 and 7).

4. Re-enables the interrupt system and enters the driver to process the interrupt.

The above means that a privileged system processes standard (that is, non-privileged) interrupts and requests for system functions with the interrupt system in a held-off state, rather than with the interrupt system disabled (as it does in non-privileged systems). The privileged interrupts are always enabled, and they can interrupt any process taking place and be serviced almost immediately.

When servicing of the non-privileged interrupt is completed, RTE clears the control flip-flop on the Privileged Interrupt Fence card and re-enables the DCPC completion interrupts if they are needed by the standard driver using DCPC. This returns the system to a state where any interrupt (privileged or non-privileged) can occur and be recognized almost immediately.

RTE records a system as privileged by storing at generation time (or, for RTE-IV, at reconfiguration time) the I/O select code address of the Privileged Interrupt card in Base Page Communications Area word DUMMY. Systems without a Privileged Interrupt card have a a zero in base page word DUMMY.

In general, privileged drivers are very similar to standard drivers, thus most of the material presented in Chapter 3 for standard drivers also applies to privileged drivers. Because only the differences are pointed out in this section, you should be familiar with the material presented in Chapter 3 before continuing with the privileged driver considerations described here.

General Privileged Driver Structure and Operation

Privileged drivers are responsible for the initiation, continuation, and completion of all I/O requests for privileged devices just as in a standard driver. Because privileged drivers operate independently of RTE, there are several additional requirements and restrictions that must be followed to ensure the integrity of the operating system and the proper operation of the driver. These restrictions and requirements are described in subsequent subsections.

Privileged drivers are generally designed in three sections: 1) an initiation section, 2) a privileged section, and 3) a completion section. The driver must have a name in the form DV*ynn*, and the initiation and completion entry points must have names in the forms I*xnn* and C*xnn* respectively. The rules for the choice of x, *y*, and *nn* are the same as those given previously for standard drivers. There are no special rules for the entry point name of the privileged section. For consistency with I*xnn* and C*xnn*, it is suggested that a name such as P*xnn* be chosen where *x* and *nn* agree with the characters chosen for I*xnn* and C*xnn*.

IOC calls the initiation section of a privileged driver when an I/O request for the privileged device is made. This call has the same format as the call to the initiation section of a standard driver, described in the previous chapter.

The privileged section of a privileged driver is somewhat similar to the continuation section of a standard driver. The privileged section is entered on each interrupt from the privileged device controller and is responsible for reading and writing the next data word and restarting the device. Because the privileged section is entered directly from the trap cell on interrupt (rather than from CIC), it must save and restore the state of the computer on entry and exit. These tasks are performed by CIC for standard drivers.

The completion section of a privileged driver has an entry point named C*xnn* and is responsible for returning to RTE when the I/O transfer is complete.

The overall operation of a privileged I/O request from initiation to completion is summarized below:

1. The privileged driver is called by a standard EXEC I/O call.

2. If the request is being made by a user program and the call is not buffered, the calling program is placed into I/O suspension.

3. Each time the device controller interrupts, the system overhead is circumvented because the privileged section P*xnn* is entered directly.

4. After each interrupt, if another data transfer is still required to satisfy the buffer length, the device controller is restarted and the privileged section is exited.

5. When the entire data buffer has been filled, the driver needs a way to inform RTE that the transfer is complete. This is accomplished by allowing the driver to timeout, which causes IOC to re-enter the driver at C*xnn*.

6. C*xnn* returns the transmission log (via the B-Register) and a successful completion indication (via the A-Register) to IOC.

7. IOC then reschedules the program that made the I/O request.

Initiation Section

The initiation section of a privileged driver performs the functions listed below. The list is similar to the one given earlier for standard drivers with the exception that no DCPC processing can be done by privileged drivers. See the "Privileged Driver Design Considerations" subsection of this manual.

1. Checks for powerfail/auto-restart entry by examining bit 15 of EQT entry word 5, which is set to 1 only on this type of entry. If bit 15 is set, the appropriate powerfail/auto-restart processing should be done. This check need only be made by drivers that are designed to process powerfail interrupts (as described in the "Powerfail Processing" subsection).

2. Configures all I/O instructions in the driver to reference the specific I/O select code of the device controller. This step is done only on the first entry to the driver because there is one privileged driver for each privileged device controller in the system. See the "Privileged Driver Design Considerations" subsection.

3. Clears bit 12 in EQT entry word 4 if timeouts are to be handled by the system. This bit will be reset to 1 by the privileged section when it sets up to complete the call.

4. Rejects the request and follows the procedure described in step "6" if:

    1. A status check of the device or controller indicates that it is inoperable, or

    2. The request code or other parameters are illegal.

5. Initializes software flags and activates the device controller. All variable information pertinent to the transmission can be saved in the EQT entry associated with the controller, providing that the driver saves the addresses of the EQT entry internally in the driver itself at initiation. These addresses are not available on base page on subsequent entries to the driver. See the "Privileged Driver Design Considerations" subsection.

6. Returns to IOC (via JMP I*xnn*,I) with the A-Register set to indicate initiation or rejection (and the cause of the rejection) as follows:

A-Reg	Status
0	The operation was initiated successfully.
1	Operation rejected: read or write illegal for device.
2	Operation rejected: control request illegal or undefined.
3	Operation rejected: equipment malfunction or not ready.
4	The operation was immediately completed. This means that the driver was able to completely satisfy the request without the need of a subsequent interrupt and that the program making the I/O call can be rescheduled immediately. The B-Register should be set to the number of words or characters (depending upon which the user specified) transferred. This value is known as the transmission log.
5	This return must NOT be used by privileged drivers.

6 Return DCPC − should NOT be used.

7 to 99 The program making the I/O request is aborted (unless the no-abort bit was set in the call), and an I/O error message is printed on the system console. Note that this return can be used for unbuffered I/O requests only. This, therefore, excludes the use of return codes 7 through 99 on any Class, buffered, or system I/O request. The error message has the following format:

```
IOxx yyyyy
NNNNN ABORTED
```

where:

xx = the return code from the driver (decimal 07 to decimal 99),

yyyyy = the address of the aborted I/O request in program *NNNNN*, and

NNNNN = the name of the program that made the I/O request.

This type of return can be used by drivers to generate their own I/O error messages at the system console. Note that certain codes are reserved for system use, as follows:

Return Code	Reserved for
7 − 59	HP system modules and system drivers.
60 − 99	User-written drivers.

The method of setting up the trap cell is to point the trap cell directly to the privileged section entry point when the system is generated. This is done by entering *sc*,ENT,P*xnn* (where *sc* is the select code of the privileged controller) during the Interrupt Table definition phase of the generation. When the generator detects an entry of this form it places a JSB $JPNN,I instruction (where $JPNN contains the address of P*xnn*) into the appropriate interrupt trap cell. The generator also places a zero in the corresponding Interrupt Table entry to indicate that interrupts on the select code are not handled by RTE. The privileged section entry point, P*xnn*, is declared as an entry point in the privileged driver (via the ENT pseudo-instruction).

Privileged Section

When a privileged interrupt occurs, the operation currently in progress is suspended, and the privileged section of the driver is entered directly via the JSB $JPNN,I instruction in the trap cell. In addition to the normal tasks associated with continuing the data transfer, the privileged section is required to perform several housekeeping functions that are normally performed by CIC. This includes saving and restoring the state of the computer on entry and exit and disabling the DCPC completion interrupts so that the driver's operation is not interrupted.

The privileged section of a privileged driver performs the following functions:

1. Executes the following tasks done by CIC for standard drivers:

   a. Disables the entire interrupt system with a CLF 0 instruction so that the driver is not interrupted while performing the housekeeping functions.

b. Disables the DCPC completion interrupts by issuing a CLC 6 instruction and a CLC 7 instruction. The DCPC completion interrupts are associated with I/O select codes 6 and 7 and, therefore, precede the Privileged Interrupt card in the I/O priority chain. Interrupts from these device controllers are not held off by the Privileged Interrupt card and must be disabled to prevent an interrupt from occurring while the privileged driver is executing.

c. Saves the current contents of all program accessible registers (A, B, E, O, and if present, X, and Y) in a local buffer. These registers must be restored to their original contents before exiting the driver.

d. Saves the previous state of the memory protect fence. When an interrupt occurs, the memory protect fence (if ON) is automatically turned off. The driver can determine the previous state of the memory protect fence (which is the state to which it should be restored after processing the interrupt) by examining Base Page Communications Area word MPTFL. If MPTFL equals zero, memory protect was ON when the privileged interrupt occurred and the privileged section must turn the fence back on before exiting. If MPTFL is non-zero, memory protect was OFF, and the privileged section must not restore the memory protect fence.

e. Sets base page word MPTFL to 1 to indicate that the memory protect fence is now OFF.

f. Saves the status of the Dynamic Mapping System so that it can be restored to its original state before returning to the point of interruption. This is done by executing an SSM instruction. (Applicable to systems with Dynamic Mapping only.)

g. Re-enables the interrupt system by executing an STF 0 instruction. This allows a higher priority privileged controller (if one exists) to interrupt the driver. All lower priority (higher select code) privileged interrupts and non-privileged interrupts are held off because the flag is still set on the card that caused the privileged interrupt.

2. Checks whether bits 14 through 0 of EQT word 1 (the controller I/O request list pointer) equal zero. If so, a spurious interrupt has occurred (that is, no I/O operation was in progress at the time of the interrupt). The driver ignores the interrupt as follows:

   a. Disables the interrupt system via a CLF 0 instruction so that the driver is not interrupted while clearing the controller.

   b. Clears the control and flag flip-flops on the controller (usually via a CLC DEVIC,C instruction).

   c. Proceeds to step “5d” below to restore the computer to its original state before exiting.

3. Performs the input or output of the next data item. One of the following three actions is then taken:

   a. If the transfer is not complete, the driver follows the procedure described in step “5” to return to the suspended process at the point of interruption.

   b. If the transfer is complete, the driver follows the procedure described in step “4” to set up for a completion return to IOC.

c. If the driver detects a transmission error, it may reinitiate the transfer and attempt a retransmission. A counter for the number of retry attempts can be kept in the EQT entry. After initiating each retry, the driver follows the procedure described in step "5" to return to the suspended process at the point of interruption.

4. Once the transfer is complete, the driver needs a way to indicate this fact to RTE so that the program that made the I/O request can be rescheduled. This is accomplished by forcing the driver to time out. To do this, the driver performs the following steps:

   a. Disables the interrupt system with a CLF 0 instruction so that no interruptions occur while the timeout is being set up and the computer is being restored to its original state.

   b. Turns off the device controller to prevent further interrupts (usually with a CLC instruction).

   c. Sets the timeout clock (EQT entry word 15) to −1 to cause a timeout of the privileged device controller at the next tick of the real-time clock. This will cause the completion section of the driver to be entered from IOC so that a normal completion return can be made to RTE.

   d. Sets the "driver processes timeout" bit in EQT entry word 4 to one so that the driver will be re-entered when the timeout occurs.

   e. Follows the procedure described in step "5d" below to restore the computer to its original state before exiting.

5. Before returning to the point of interruption, the privileged section performs the following steps to restore the computer to its original state upon entry:

   a. Disables the interrupt system so that no interruptions occur while the computer is being restored to its original state.

   b. Encodes the device controller to initiate the next data transfer, usually via a STC DEVIC,C instruction. Note that the device controller must not be encoded until the interrupt system is disabled and the driver is about to return to the point of interruption. If the device controller were encoded earlier, the driver might be re-entered at P*xnn* by the next interrupt before completely servicing the previous interrupt. Clearing of the flag on the privileged device controller's I/O card will also allow any lower priority interrupts to be recognized by the system when the interrupt system is re-enabled.

   c. Copies EQT word 14 to EQT word 15; this is optional, refer to the section "Timeout Values for Privileged Drivers" in this chapter for more information.

   d. Checks to see if either of the DCPC completion interrupts must be re-enabled, as follows:

      - If the memory protect fence was initially OFF, the driver must not re-enable the DCPC completion interrupts. If the memory protect fence was initially OFF, the privileged driver interrupted the operation of the system or another privileged driver. These routines operate with the DCPC completion interrupts disabled and assume that the completion interrupts will remain disabled if they are interrupted.

- If the memory protect fence was initially ON, the DCPC completion interrupts are turned back on only if the standard driver currently using the DCPC channel needs the interrupt. Standard drivers that need the DCPC completion interrupt set bit 15 of the appropriate DCPC Assignment Control Word (in the Interrupt Table) to 1 as a flag. See the "DCPC Processing" subsection earlier in this manual. If bit 15 is set, a privileged driver must re-enable the appropriate DCPC completion interrupt by issuing a STC 6 or STC 7 instruction. If bit 15 is not set, the privileged driver must not re-enable the interrupt.

d. Restores all saved registers to their original values.

e. Restores base page word MPTFL to its original value. This word is used to indicate the current status (ON/OFF) of the memory protect fence.

f. Turns the interrupt system back on via a STF 0 instruction to allow other interrupts to occur.

g. Turns the memory protect fence back on via a STC 5 instruction if the fence was ON initially.

h. Performs one of the following actions depending on whether or not the system includes the Dynamic Mapping feature:

   - Restores the Dynamic Mapping System to its original value at interrupt and returns to the suspended process at the point of interruption by executing a jump and restore status (JRS) instruction indirect through the entry point P*xnn*. (Performed only in systems with the Dynamic Mapping feature.)
   - Returns to the suspended process at the point of interruption via a jump indirect through the entry point P*xnn*. (Performed only in systems without the Dynamic Mapping feature.)

Note If the memory protect fence was turned on in step "g", execution of the JRS instruction (in step "h") to restore DMS status can only be performed if the System Map is currently enabled. An attempt to execute it with the User Map enabled will result in a DMS violation. Thus, if the driver switches to operation under the User Map for any reason, the System Map must be re-enabled before executing the JRS instruction. The explanation of map switching given in the "Subroutines for Special Mapping Functions" subsection illustrates this procedure.

Completion Section

When the timeout set up by the privileged section occurs, IOC enters the continuation section of the privileged driver at entry point C*xnn*. The continuation section sets the A-Register equal to the appropriate completion code and the B-Register equal to the transmission log. It then returns to IOC via a jump indirect through the entry point C*xnn* (JMP C*xnn*,I). The return point (P+1) and allowable completion codes (0-4) are the same as those described earlier for the completion section of a standard driver.

Privileged Driver Design Considerations

Privileged drivers operate independently of RTE. In fact, the operation of the RTE operating system itself may be suspended while a privileged interrupt is being serviced. As a result, the writer of a privileged driver must adhere to the following design requirements:

1. Privileged drivers must not use any of the features or request calls of RTE. Calling a system process might involve entering RTE while it is processing another request. This cannot be allowed because RTE is not re-entrant.

2. Privileged drivers cannot use either DCPC channel because it is very difficult to coordinate the use of DCPC with the operating system and other drivers that may be using DCPC.

3. The initiation section must read the EQT entry addresses from the Base Page Communications Area and save them internally in the driver. These addresses are not available in base page on subsequent interrupts because the privileged driver is entered directly from the trap cell instead of from CIC. (CIC is the module that places these addresses into the base page before calling a standard RTE driver to process an interrupt.)

4. Because privileged drivers are required to keep information relating to the I/O request internally (see "3" above), a separate privileged driver is required for each privileged device controller present in the system. For each additional controller of the same type, an additional copy of the privileged driver must be generated into the system. Each copy of the driver must have unique names for DV*ynn*, I*xnn*, P*xnn*, and C*xnn*.

Communication with User Programs (DMS Systems Only)

Privileged drivers are automatically entered with the System Map enabled when a privileged interrupt occurs. If the I/O request buffer for the privileged call is located in a user program, the driver must switch maps before it can access the buffer. Any privileged driver in a DMS system should therefore be designed for user communication through SYSTEM COMMON or the Subsystem Global Area (SSGA) to avoid the overhead of map switching. These areas can be specified at generation to be included in both the System Map and User Map, and hence can be accessed directly by both user programs and privileged drivers without any map switching.

Otherwise, if the I/O request buffer is located in a user program, some map switching will have to be done before the privileged driver can access the buffer. This map switching procedure is described in detail in the "Subroutines for Special Mapping Functions" subsections of this manual.

Discussion of Sample DMS Privileged Driver

The following discussion describes Figure 4-1, an example of a privileged driver written specifically for use in a system with the Dynamic Mapping feature. Figure 4-2 shows a similar driver written specifically for a system that does not include the Dynamic Mapping feature. For the purposes of the discussion, this driver has been given the generalized name of DVYNN.

The device controller transfers one word of data each time it interrupts, and the data is stored in a buffer passed to the driver via the call parameters. Note that the design of the DMS privileged driver assumes that the I/O request buffer is located in SYSTEM COMMON for two reasons: 1) it ensures that the driver's initiation section is entered with the System Map enabled (this is necessary for the proper operation of the trap cell modification technique used in that section), and 2), it allows the driver to place data values directly in the I/O request buffer without any map switching.

Note that the driver does not process powerfail interrupts nor does it process timeouts, except for the timeout it creates as a means to complete the I/O request and return to IOC.

Initiation Section

Refer to the partial listing of the sample privileged driver in Figure 4-1 (or Figure 4-2). A standard I/O call to input from the device causes the calling program to be I/O suspended and the driver to be entered at Ixnn.

Because this driver can control just one device controller, there is no need to configure the I/O instructions more than once. Therefore, the driver is configured the first time it is entered, and the switch at "FIRST" is set so that the configuration code is not executed on any subsequent entry to the driver. The initiation section also saves the addresses of those EQT entry words that will be used by the privileged section because these addresses will not be available in base page on subsequent interrupts.

The request code is checked for validity. All write and control requests (except a "clear" control request from the system) are rejected. For read requests, a counter is established for the number of readings to be taken, and the buffer address for the storage of the data is saved. The "driver processes timeout" bit in EQT entry word 4 is cleared so that any unexpected timeouts are handled by the system. This bit is later reset to 1 by the privileged section when it sets up a timeout as a means of returning to IOC at the end of the I/O request. Finally, the initiation section sets up and encodes the device controller to begin a read operation and returns to IOC.

Privileged Section

When the device controller interrupts, the privileged section (Pxnn) is entered directly as a result of the controller's trap cell modification.

Because entry is made directly into Pxnn, Pxnn must do the housekeeping that is normally done by CIC when a standard interrupt occurs. Thus, before Pxnn can turn the interrupt system back on to allow higher priority privileged interrupts to be recognized, Pxnn must ensure that the DCPC channels cannot interrupt, save the user-programmable registers, save the old memory protect status, and set its new status. For systems with Dynamic Mapping, Pxnn must also save the Dynamic Mapping System status at the time of interrupt.

Pxnn then loads and stores the data in the next unfilled buffer word. If there is yet another data point to be taken, Pxnn sets up the device controller for the next reading, disables the interrupt system, encodes the device controller, restores memory protect status and its flag, restores the user programmable registers, turns the interrupt system back on, and exits. For systems with Dynamic Mapping, Pxnn must also restore the Dynamic Mapping System status to its original value. All of this basically resets the system to its state before Pxnn was entered.

When the privileged section wants to perform system operations such as scheduling a program, and still continue with the privileged section, then Pxnn should set EQT word 15 to −1 so a timeout will occur at the next tick of the real-time clock. The continuation section can perform the required system task and then do a continuation exit.

When the last reading is taken, Pxnn disables the interrupt system, turns off the device controller, and sets up the privileged controller's EQT so that a timeout will occur at the next tick of the real-time clock. Pxnn then resets the system to its original state and returns to the suspended process at the point of interruption.

Completion Section

The status of the device controller and the driver is now unchanged until the Time Base Generator (TBG) interrupts. The TBG causes a timeout of the privileged controller (because a −1 was set into EQT entry word 15), which in turn causes IOC to pass control to the completion section at Cxnn. The completion section simply sets the A- and B-Registers to the status and transmission log, respectively, and returns to IOC. IOC then reschedules the calling program and initiates any remaining requests for the controller as if it were a standard (non-privileged) controller.

Timeout Values for Privileged Drivers

If you wish to specify a timeout value for the privileged controller (to prevent indefinite suspension in the event that the controller malfunctions), the timeout value must be long enough to cover the entire period from I/O initiation to completion. This is different from the timeout value for a standard driver, which is normally only long enough to cover the expected time between interrupts from the standard device controller.

Each time IOC or CIC enters a standard driver to initiate or continue an I/O request, it resets the timeout clock (EQT entry word 15) to the value specified at generation. However, because privileged drivers are not entered from CIC on interrupt, the timeout value is inserted into the privileged controller's timeout clock only at initiation. If this value is not long enough to cover the entire I/O transfer period, a timeout will occur while the data transfer is still in progress, and the transfer will be prematurely terminated. This can be prevented by specifying a suitably long timeout value or by specifying a timeout value of zero (which disables the timeout feature entirely), or by setting EQT word 15 on each privileged entry.

The timeout value set by the user to prevent indefinite suspension should not be confused with the timeout set up by the privileged driver to complete the call and return to IOC. In the latter case, the driver overrides the user-specified timeout by inserting its own value (−1) directly into the timeout clock before returning.

Subroutines for Special Mapping Functions (DMS Systems Only)

The mapping considerations for the initiation and continuation/completion sections of privileged drivers are the same as those for standard RTE drivers (see Chapter 3 of this manual). Prior to entering either of these sections, RTE will load and enable the correct map (System or User) needed to describe the I/O request buffer.

The privileged section of a privileged driver requires special mapping considerations, however. The System Map will always be enabled when the privileged section is entered because this section is entered directly from the interrupt trap cell. If the I/O buffer is not accessible under the System Map, the privileged section must reload the User Map to describe the appropriate program before accessing the buffer. Therefore, if the privileged driver only processes I/O requests from buffers in System Common or System Available Memory, the privileged section can access the I/O buffer directly without making special mapping considerations.

Subroutine $PVMP can be used by privileged drivers to perform any needed map switching. The driver uses $PVMP as follows:

1. Save the contents of the current User Map.

2. Load the A-Register with the ID Segment address of the program containing the buffer.

3. Call $PVMP (JSB $PVMP) to load the User Map for the program specified in the A-Register.

4. Check for an error return. $PVMP returns with the A-Register set to zero if the specified program was not in memory and the User Map could not be set up.

5. If no error was detected, access the buffer through the newly loaded User Map.

6. After all accesses to the buffer have been made, restore the User Map to its original contents and continue with normal operation.

The only time a privileged driver must set up the user map is when the I/O buffers are in the program (that is, a user normal request, T field − 00). In this case, the driver can obtain the program's ID segment address from EQT word 1 during driver initiation.

The recommended procedure for using $PVMP is somewhat different in RTE-III and RTE-IV. Each of these procedures is described in a separate section below.

Note Subroutine $PVMP can be called by privileged RTE drivers only. Standard drivers wishing to perform the same function must use subroutine $XDMP, which is described in Chapter 3 of this manual.

Mapping in RTE-III and RTE-M/III

Before entering the initiation or continuation section of a privileged driver, IOC enables the correct map needed to process the call, as follows:

1. When the I/O request buffer is located in SYSTEM COMMON or System Available Memory, IOC enables the System Map before entering the driver.

2. When the I/O request buffer is located in the calling program, IOC enables the User Map before entering the driver.

However, the System Map is always enabled upon entry to the privileged section, because the privileged section is entered directly from the interrupt trap cell.

Because the buffer, in general, may be located in either the User Map (program not swappable) or the System Map, the driver needs to identify which map is used. If necessary, the driver can then use $PVMP to access the buffer while in the privileged section.

In the following example, the initiation section of the privileged driver checks the status of the Dynamic Mapping System and sets a flag to indicate whether the System or User Map is enabled. The initiator also retrieves the ID Segment address of the program making the I/O request. Note that the driver is assuming that the program making the I/O request has cooperated with the driver and placed its ID Segment address in the first optional parameter.

The privileged section of the driver then saves the current state of the User Map registers and Dynamic Mapping System. After the driver has determined that the User Map is needed, it calls $PVMP to reload the User Map registers to describe the calling program. The driver then switches to operation under this map and accesses the memory described by it. After all accesses have been made, the driver re-enables the System Map and restores the original state of the User Map registers before continuing with its normal processing under the System Map.

```
        EXT $PVMP

IXNN    NOP             Initiation Section entry point
                        (System Map or User Map enabled depending on
            .           location of I/O request buffer.)
            .
            .
        CLA
        STA IDADR       Clear IDADR signifying System Map used
        RSA             Access Dynamic Mapping System Status

        ALF             Position bit 12 into bit 0
        SLA,RSS         System Map enabled?
        JMP PROCD       Yes, System map enabled
                        No, user map enabled

        LDA EQT9,I      Access address of program making request
        STA IDADR       Save for use of Privileged Section later

PROCD   NOP
            .
            .           Continue Initiation Section processing
```

```
              .
        JMP IXNN,I      Return to IOC
PXNN    NOP             Privileged Section entry point (System Map Enabled)
              .
              .         Normal driver processing under System Map
              .
        RSA             Get Dynamic Mapping System (DMS) status
        RAL,RAL         Position current status in upper bits
        STA DMSST       Save status for later

        LDA MAPAD       Set A = address of User Map storage area
        IOR SIGN        Set sign bit indicating Store Map in Memory
        USA             Save current User Map in memory for later

        LDA IDADR       Access IDADR to determine if User or System Map used
        SZA,RSS         System Map used?
        JMP CONT        Yes, System Map used
                        No, User Map used
*
        JSB $PVMP       Call $PVMP to set up User Map for this program
        SZA,RSS         Check for error return
        JMP ERROR       Error exists, go handle it

        UJP CONT        No, errors.  Enable new User Map and continue.
CONT          .
              .         Process buffer under new User Map.
              .
        LDA IDADR       Determine if System or User Map is used
        SZA,RSS         System Map used?
        JMP NXT         Yes, System Map used
*                       No, User Map used

        SJP NEXT        Re-enable System Map
NEXT    LDA MAPAD       Access address of User Map storage area
        USA             Restore original contents of User Map

        JRS DMSSTNXT    Restore original DMS status (i.e., System Map)
NXT           .         Enable Interrupt System.
              .
              .         Proceed with normal processing under System Map
              .
MAPAD   DEF MAP         Address of User Map storage area
MAP     BSS 32          User Map storage area
SIGN    OCT 100000
IDADR   BSS 1           ID segment address saved by initiation section
DMSST   BSS 1           Temporary storage for DMS status
```

Remember that any driver using this routine must save the original contents of the User Map registers before calling $PVMP. It must also restore the original User Map after all accesses to the buffer have been made. Also remember that if the driver accesses the memory described by the User Map by enabling the map, it must save and restore the DMS status in addition to the original contents of the User Map. The example above illustrates this procedure.

Note The privileged driver could also access the buffer in the user program through a series of cross-map loads and stores without actually enabling the User Map. This in fact is the recommended procedure for using $PVMP in an RTE-IV system, and use of it by a privileged driver would allow the same privileged driver to be used in either type of system.

Mapping in RTE-IV and RTE-6/VM

Privileged drivers in RTE-IV must reside in the System Driver Area (SDA) of memory because the privileged section is always entered under the System Map directly from the trap cell. The driver can be placed in SDA by selecting either the "M" or "S" option during the EQT entry definition phase of system generation (see Chapter 3 of this manual for a discussion of the "M" and "S" options). The initiation and continuation/completion sections of the driver may be entered under either the System Map or the User Map depending on the location of the I/O buffer and the option selected at generation. The privileged section, however, will always be entered under the System Map.

If the I/O buffer being processed by the driver is located in System Common on System Available Memory, the privileged driver will always have access to the buffer under the System Map. If, however, the I/O buffer is located in the user program, the driver must consider two possible situations in which it is operating under the System Map while the buffer is in the User Map.

1. The initiation and continuation/completion sections of the driver must access the buffer through the User Map if the "M" option was selected.

2. The privileged section must access the buffer through the User Map regardless of the option selected.

A privileged driver that needs to access a buffer in a user program while operating under the System Map can use the $PVMP subroutine to reload the User Map to describe the desired program.

In the following example of an "S" option privileged driver, the initiation section of the driver checks the request type field in EQT6 (bits 14 and 15) and, if the request type is 0, saves the ID segment address of the program making the request. This is the only request type that has the user's buffer in its map. For these requests, the ID segment address is in the system map, so the driver saves 0 for the ID segment address.

The privileged section of the driver then saves the current contents of the User Map. If the User Map is needed to access the buffer, the privileged section then calls $PVMP to reload the User Map registers to describe the calling program. The driver then uses a series of cross-map loads and stores to access the buffer described by the User Map. Note that the User Map should not be enabled because drivers in SDA are not necessarily included in all user maps. After all accesses have been made, the driver restores the original state of the User Map registers before continuing with its normal processing under the System Map.

```
        EXT $PVMP

IXNN    NOP             Initiation Section entry point
           .            (System Map or User Map enabled depending on
           .            location of I/O request buffer.)
           .
        CLB
        LDA EQT6,I      Get Control word
        AND =B140000    Isolate T field
        SZA,RSS         User normal request
        LDB EQT1,I      Yes, get ID segment address
        STB IDADR       Save for use of Privileged Section later

PROCD   NOP
           .
           .            Continue Initiation Section processing
           .
        JMP IXNN,I      Return to IOC
PXNN    NOP             Privileged Section entry point (System Map Enabled)
           .
           .            Normal driver processing under System Map.
           .
        LDA MAPAD       Set A = address of User Map storage area
        IOR SIGN        Set sign bit indicating Store Map in memory
        USA             Save current User Map in memory for later

        LDA IDADR       Access IDADR to determine if User or System Map used
        SZA,RSS         System Map used?
        JMP SYACC       Yes, System Map used − access buffer directly
*                       No, User Map used
        JSB $PVMP       Call $PVMP to set up User Map for this program
        SZA,RSS         Check for error return
        JMP ERROR       Error exists, go handle it

           .            No errors.  System Map is still enabled.
           .            Access buffer via a series of cross-map loads and stores
           .            because SDA drivers are not included in all User maps.

         LDA MAPAD      Access address of User Map storage area
        USA             Restore original contents of User Map


           .
           .            Proceed with normal processing under System Map
           .

MAPAD   DEF MAP         Address of User Map storage area
MAP     BSS 32          User Map storage area
SIGN    OCT 100000
IDADR   BSS 1           ID segment address saved by initiation section
```

Remember that any driver using this routine must save the original contents of the User Map registers before calling $PVMP and must restore the registers to their original value after all accesses to the buffer have been made. The example above illustrates this procedure.

Sample Privileged Drivers

The sample drivers illustrated in Figures 4-1 and 4-2 demonstrate some of the principles involved in writing a privileged I/O driver for use in an RTE system with Dynamic Mapping (Figure 4-1) and without Dynamic Mapping (Figure 4-2). Note that the drivers are examples only and are not drivers supplied with the system.

```
                ASMB,Q
*
                      NAM DVYNN       **DMS PRIVILEGED DRIVER EXAMPLE**

                      SUP
*
                      ENT IXNN,CXNN
*

*********************************************************
* SAMPLE RTE PRIVILEGED DRIVER DVYNN - FOR DMS SYSTEMS *
*********************************************************
*
* HANDLES USER PROGRAM REQUESTS TO READ FROM A PRIVILEGED
* CONTROLLER
*
* USER PROGRAM CALLING SEQUENCE:
*
*   JSB EXEC       CALL EXEC
*   DEF *+5        RETURN POINT
*   DEF RCODE      REQUEST CODE (MUST BE READ REQUEST)
*   DEF CONWD      CONTROL WORD
*   DEF BUFFR      ADDRESS OF BUFFER (MUST BE IN SYSTEM COMMON)
*   DEF LENTH      LENGTH OF BUFFER
*
* CAUTION:
*
* THIS DRIVER WILL NOT WORK WITH MORE THAN ONE PRIVILEGED
* CONTROLLER.  IF MORE THAN ONE PRIVILEGED CONTROLLER
* EXISTS IN A SYSTEM, DVYNN MUST BE
* RE-ASSEMBLED WITH ALL NAMES CONTAINING "NN" CHANGED SO
* THAT EACH COPY OF THE DRIVER HAS UNIQUE ENTRY POINTS.
* THEN ONE DRIVER PER CONTROLLER MUST BE PUT
* INTO THE SYSTEM AT GENERATION TIME.
*
* NOTE:
*
* 1.) THE DESIGN OF THIS DRIVER ASSUMES THAT THE I/O
*     BUFFER BEING PROCESSED IS LOCATED IN SYSTEM COMMON.
*     CONSEQUENTLY, THE DRIVER IS ENTERED WITH THE
*     SYSTEM MAP ENABLED.  THIS IS NECESSARY FOR THE
*     CORRECT OPERATION OF THE TRAP CELL MODIFICATION
*     TECHNIQUE ILLUSTRATED BELOW.  IN ADDITION, THE
*     BUFFER IN SYSTEM COMMON ALLOWS THE DRIVER TO PUT THE
*     DATA VALUES DIRECTLY INTO THE BUFFER, WITHOUT
*     THE NEED FOR MAP SWITCHING
*
```

Figure 4-1. DMS Privileged RTE Driver Example

```
* 2.) THIS DRIVER DOES NOT PROCESS POWER-FAIL INTERRUPTS.
*
* 3.) THIS DRIVER DOES NOT PROCESS ANY TIME-OUTS EXCEPT
*     FOR THE TIME-OUT THAT IT CREATES AS A MEANS TO
*     COMPLETE THE I/O REQUEST AND RETURN TO IOC
*
*   **********************
*   * INITIATION SECTION *
*   **********************
*
  00000 000000  IXNN  NOP           INITIATION SECTION ENTRY POINT
  00001 000201R       STA SCODE     SAVE SELECT CODE OF CONTROLLER
*
  00002 000204R       LDB FIRST     ACCESS FIRST TIME THROUGH FLAG
  00003 006002        SZB           IS THIS THE FIRST TIME THROUGH?
  00004 000020R       JMP INIT      NO, SO SKIP CONFIGURATION CODE
*
* CONFIGURE I/O INSTRUCTIONS
*
  00005 000220R       IOR LIA       CREATE LIA INSTRUCTION
*
* SAVE EQT ADDRESSES
*
  00010 001774        LDA EQT15     SAVE EQT15
  00011 000216R       STA EQ15
  00012 001663        LDA EQT4      EQT4
  00013 000215R       STA EQ4
  00014 001660        LDA EQT1      AND EQT1
  00015 000214R       STA EQ1       ADDRESSES
*
  00016 002404        CLA,INA       SET FLAG TO PREVENT CONFIGURING ON
  00017 000204R       STA FIRST     SUBSEQUENT INITIATIONS
*
* CLEAR THE "DRIVER PROCESSES TIME-OUT" BIT TO ALLOW
* NORMAL TIME-OUT OPERATION
*
  00020 001663  INIT  LDA EQT4,I    ACCESS EQT WORD 4
  00021 000227R       AND =B167777  CLEAR BIT 12
  00022 001663        STA EQT4,I    AND RESET EQT WORD 4
*
* CHECK THE REQUEST CODE
*
  00023 001665        LDA EQT6,I    ACCESS REQUEST CODE
  00024 000224R       AND =B3       ISOLATE REQUEST TYPE
  00025 000222R       CPA =B1       READ REQUEST?
  00026 000041R       JMP PROC      YES, GO PROCESS READ REQUEST
*
  00027 000224R       CPA =B3       CONTROL REQUEST?
  00030 000033R       JMP CNTRL     YES, GO PROCESS CONTROL REQUEST
*
  00031 002404        CLA,INA       NO, REJECT AS ILLEGAL WRITE REQUEST
  00032 000000R       JMP IXNN,I
```

Figure 4-1. DMS Privileged RTE Driver Example (continued)

```
*
* CONTROL REQUEST.  CHECK IF IT IS A "CLEAR" CONTROL REQUEST
* IF SO, ASSUME IT WAS ISSUED BY SYSTEM, CLEAR DEVICE, AND RETURN.
*
  00033 001665  CNTRL LDA EQT6,I    ACCESS CONTROL WORD
  00034 000225R       AND =B3700    ISOLATE SUBFUNCTION
  00035 002002        SZA           "CLEAR" REQUEST?
  00036 000037R       JMP REJCT     NO, REJECT AS ILLEGAL CONTRL REQUEST
*       .
*       .          EXECUTE CODE TO CLEAR CONTROLLER
*       .
  00037 000223R REJCT LDA =B2       REJECT AS ILLEGAL CONTROL REQUEST
  00040 000000R       JMP IXNN,I
*
* SET UP FOR THE DATA TRANSFER
*
  00041 001667  PROC  LDA EQT8,I    ACCESS $ OF CONVERSIONS REQUIRED
  00042 003004        CMA,INA       NEGATE FOR CONVERSION COUNTER
  00043 000202R       STA CVCTR     AND SAVE
  00044 002021        SSA,RSS       REJECT IF
  00045 000037R       JMP REJCT     NUMBER <0
  00046 001666        LDA EQT7,I    SAVE DATA BUFFER ADDRESS
  00047 000203R       STA DAPTR     FOR PXNN
*
* INITIATE A READ AND RETURN
*
  00050 000054R       JSB READ      START A READ
  00051 103700  I.1   STC SC,C      ENCODE DEVICE
  00052 002400        CLA
  00053 000000R       JMP IXNN,I    RETURN TO IOC
*
* SUBROUTINE TO INITIATE A READ
*
  00054 000000  READ  NOP           ROUTINE CONTAINING
*       .                           CONFIGURED I/O
*       .                           INSTRUCTION TO
*       .                           SET UP THE DEVICE
*       .                           TO INITIATE ONE READING
  00055 000054R       JMP READ,I
*
*   **********************
*   * PRIVILEGED SECTION *
*   **********************
*
* SAVE STATE OF COMPUTER AT INTERRUPT
*
  00056 000000  PXNN  NOP           PRIVILEGED SECTION ENTRY POINT
*
  00057 103100        CLF 0         TURN OFF INTERRUPT SYSTEM
*
  00060 106706        CLC 6         TURN OFF DCPC COMPLETION INTERRUPTS
  00061 106707        CLC 7
```

Figure 4-1. DMS Privileged RTE Driver Example (continued)

```
*
  00062 000205R       STA ASV       SAVE REGISTERS
  00063 000206R       STB BSV
  00064 001520        ERA,ALS
  00065 102201        SOC
  00066 002004        INA
  00067 000207R       STA EOSV
  00070 105743        STX XSV       SAVE X REGISTER
  00072 105753        STY YSV       SAVE Y REGISTER
  00074 105714        SSM DMSTS     SAVE DYNAMIC MAPPING SYSTEM STATUS
*
  00076 001770        LDA MPTFL     SAVE OLD MEMORY PROTECT FLAG
  00077 000213R       STA MPFSV
  00100 002404        CLA,INA       SET MEMORY PROTECT FLAG TO OFF
  00101 001770        STA MPTFL     BECAUSE MEMORY PROTECT IS NOW OFF
*
  00102 102100        STF 0         TURN INTERRUPT SYSTEM BACK ON
*
* CHECK FOR SPURIOUS INTERRUPT
*
  00103 000214R       LDA EQ1,I     ACCESS REQUEST LIST POINTER WORD
  00104 000226R       AND =B77777   ISOLATE REQUEST LIST POINTER
  00105 002002        SZA           IS A REQUEST IN PROGRESS?
  00106 000112R       JMP PREAD     YES, GO PROCESS INTERRUPT
*
  00107 103100        CLF 0         NO, TURN OFF INTERRUPT SYSTEM
  00110 107700  I.2   CLC SC,C      RESET CONTROLLER, AND
  00111 000122R       JMP EXIT      IGNORE SPURIOUS INTRUPT BY RETURNING
*
* PROCESS READ REQUEST
*
        000112R PREAD EQU *
*         .         LOAD IN DATA FROM DEVICE
*         .         VIA CONFIGURED I/O INSTRUCTIONS
*         .
*
  00112 000203R       STA DAPTR,I   STORE WORD IN DATA BUFFER
  00113 000202R       ISZ CVCTR     IS THIS THE LAST CONVERSION?
  00114 002001        RSS           NO
  00115 000165R       JMP DONE      YES, GO SET UP TO TERMINATE CALL
*
  00116 000203R       ISZ DAPTR     NO, SET UP FOR NEXT CONVERSION
  00117 000054R       JSB READ      INITIATE IT
*
* RESTORE MACHINE TO ORIGINAL STATE ON INTERRUPT
*
  00120 103100        CLF 0         TURN OFF INTRUPT SYSTEM TEMPORARILY
*
  00121 103700  I.3   STC SC,C      ENCODE DEVICE
*
  00122 000213R EXIT  LDA MPFSV     ACCESS PREVIOUS STATE OF MEM PROTECT

  00123 002002        SZA           WAS MEMORY PROTECT ON?
  00124 000135R       JMP EXIT1     NO, DO NOT TURN ON DCPC INTERRUPTS
```

Figure 4-1. DMS Privileged RTE Driver Example (continued)

```
*
  00125 001654        LDB INTBA     YES, TURN DCPC COMPLETION INTERRUPTS
  00126 000001        LDA B,I       BACK ON IF THEY WERE ON INITIALLY.
  00127 002020        SSA           ON/OFF STATUS IS INDICATED BY BIT 15
  00130 102706        STC 6         OF EACH DCPC ASSIGNMENT WORD IN THE
  00131 006004        INB           INTERRUPT TABLE
  00132 000001        LDA B,I
  00133 002020        SSA
  00134 102707        STC 7
*
  00135 000207R EXIT1 LDA EOSV      RESTORE E AND O REGISTERS
  00136 103101        CLO
  00137 000036        SLA,ELA
  00140 102101        STF 1
  00141 000206R       LDB BSV       RESTORE B-REGISTER
  00142 105745        LDX XSV       RESTORE X REGISTER
  00144 105755        LDY YSV       RESTORE Y REGISTER
*
  00146 000213R       LDA MPFSV     RESTORE MEMORY PROTECT FLAG
  00147 001770        STA MPTFL     IN BASE PAGE
  00150 002002        SZA           WAS MEMORY PROTECT ON AT INTERRUPT?
  00151 000160R       JMP EXIT2     NO
*
  00152 000205R       LDA ASV       YES, RESTORE A-REGISTER
  00153 102100        STF 0         TURN ON INTERRUPT SYSTEM
  00154 102705        STC 5         SET MEMORY PROTECT ON
  00155 105715        JRS DMSTS PXNN,I     RESTORE DMS STATUS AND RETURN
*
*                          (NOTE: EXECUTION OF A "JRS"
*                          INSTRUCTION AFTER TURNING THE
*                          MEMORY PROTECT FENCE ON IS
*                          ALLOWED ONLY IF SYSTEM MAP
*                          IS CURRENTLY ENABLED.  THIS
*                          DRIVER HAS BEEN DESIGNED SUCH
*                          THAT THIS IS ALWAYS THE CASE.
*
  00160 000205R EXIT2 LDA ASV       NO, RESTORE A-REGISTER
  00161 102100        STF 0         TURN ON INTERRUPTS
  00162 105715        JRS DMSTS PXNN,I     RESTORE DMS STATUS AND RETURN
*
* THIS CODE SETS UP THE TIME OUT TO COMPLETE THE CALL
*
  00165 103100  DONE  CLF 0         TURN OFF THE INTERRUPT SYSTEM
  00166 106700  I.4   CLC SC        TURN OFF PRIVILEGED DEVICE
  00167 003400        CCA           SET TIME OUT FOR
  00170 000216R       STA EQ15,I    ONE TICK AND SET
  00171 000215R       LDA EQ4,I     BIT12 IN EQT4 SO
  00172 000217R       IOR BIT12     RTIOC WILL
  00173 000215R       STA EQ4,I     CALL CXNN ON TIMEOUT
  00174 000122R       JMP EXIT      GO TO EXIT ROUTINE
*
*   **********************
*   * COMPLETION SECTION *
*   **********************
```

Figure 4-1. DMS Privileged RTE Driver Example (continued)

```
*
  00175 000000  CXNN  NOP           COMPLETION SECTION ENTRY POINT
*
  00176 002400        CLA           SET A = 0 = NORMAL RETURN
  00177 001667        LDB EQT8,I    SET B = TRANSMISSION LOG
  00200 000175R       JMP CXNN,I    RETURN TO IOC
*
* CONSTANT AND STORAGE AREA
*
        000000  A     EQU 0
        000001  B     EQU 1
        000000  SC    EQU 0         DUMMY I/O SELECT CODE NUMBER
*
  00201         SCODE BSS 1
  00202         CVCTR BSS 1
  00203         DAPTR BSS 1
  00204         FIRST BSS 1
  00205         ASV   BSS 1
  00206         BSV   BSS 1
  00207         EOSV  BSS 1
  00210         XSV   BSS 1
  00211         YSV   BSS 1
  00212         DMSTS BSS 1
  00213         MPFSV BSS 1
  00214         EQ1   BSS 1
  00215         EQ4   BSS 1
  00216         EQ15  BSS 1
  00217 010000  BIT12 OCT 10000
  00220 102500  LIA   LIA 0
*
* BASE PAGE COMMUNICATIONS AREA DEFINITION
*
        001650  .     EQU 1650B
        001654  INTBA EQU .+4
        001660  EQT1  EQU .+8
        001663  EQT4  EQU .+11
        001665  EQT6  EQU .+13
        001666  EQT7  EQU .+14
        001667  EQT8  EQU .+15
        001774  EQT15 EQU .+84
        001770  MPTFL EQU .+80
*
                      END
```

Figure 4-1. DMS Privileged RTE Driver Example (continued)

```
                ASMB,Q
*
                      NAM DVYNN  **NON-DMS PRIVILEGED DRIVER EXAMPLE **
*
                      ENT IXNN,CXNN
*
************************************************************
* SAMPLE RTE PRIVILEGED DRIVER DVYNN - FOR NON-DMS SYSTEMS *
************************************************************
*
* HANDLES USER PROGRAM REQUESTS TO READ FROM A PRIVILEGED
* CONTROLLER
*
* USER PROGRAM CALLING SEQUENCE:
*
*   JSB EXEC       CALL EXEC
*   DEF *+5        RETURN POINT
*   DEF RCODE      REQUEST CODE (MUST BE READ REQUEST)
*   DEF CONWD      CONTROL WORD
*   DEF BUFFR      ADDRESS OF BUFFER (MUST BE IN COMMON)
*   DEF LENTH      LENGTH OF BUFFER
*
* CAUTION:
*
* THIS DRIVER WILL NOT WORK WITH MORE THAN ONE PRIVILEGED
* CONTROLLER.  IF MORE THAN ONE PRIVILEGED CONTROLLER
* EXISTS IN A SYSTEM, DVYNN MUST BE
* RE-ASSEMBLED WITH ALL NAMES CONTAINING "NN" CHANGED SO
* THAT EACH COPY OF THE DRIVER HAS UNIQUE ENTRY POINTS.
* THEN ONE DRIVER PER CONTROLLER MUST BE PUT
* INTO THE SYSTEM AT GENERATION TIME.
*
* NOTE:
*
* 1.) THIS DRIVER DOES NOT PROCESS POWER-FAIL INTERRUPTS.
*
* 2.) THIS DRIVER DOES NOT PROCESS ANY TIME-OUTS EXCEPT
*     FOR THE TIME-OUT THAT IT CREATES AS A MEANS TO
*     COMPLETE THE I/O REQUEST AND RETURN TO IOC
*
*   **********************
*   * INITIATION SECTION *
*   **********************
*
  00000 000000  IXNN  NOP           INITIATION SECTION ENTRY POINT
  00001 000163R       STA SCODE     SAVE SELECT CODE OF CONTROLLER
*
  00002 000166R       LDB FIRST     ACCESS FIRST TIME THROUGH FLAG
  00003 006002        SZB           IS THIS THE FIRST TIME THRU?
  00004 000020R       JMP INIT      NO, SO SKIP CONFIGURATION CODE
*
* CONFIGURE I/O INSTRUCTIONS
```

Figure 4-2. Non-DMS Privileged RTE Driver Example

```
*
  00005 000177R       IOR LIA       CREATE LIA INSTRUCTION
*
* MODIFY TRAP CELL
*
  00006 000200R       LDA $JSB      SET TRAP CELL TO
  00007 000163R       STA SCODE,I   JSB $JPNN,I ($JPNN=ADDR OF PXNN)
*
* SAVE EQT ADDRESSES
*
  00010 001774        LDA EQT15     SAVE EQT15
  00011 000175R       STA EQ15
  00012 001663        LDA EQT4      EQT4
  00013 000174R       STA EQ4
  00014 000173R       LDA EQ1       AND EQT1
  00015 000173R       STA EQ1       ADDRESSES
*
  00016 002404        CLA,INA       SET FLAG TO PREVENT CONFIGURATING ON

  00017 000166R       STA FIRST     SUBSEQUENT INITIATIONS
*
* CLEAR THE "DRIVER PROCESSES TIMEOUT" BIT TO ALLOW
* NORMAL TIMEOUT OPERATION
*
  00020 001663  INIT  LDA EQT4,I    ACCESS EQT WORD 4
  00021 000206R       AND =B167777  CLEAR BIT 12
  00022 001663        STA EQT4,I    AND RESET EQT WORD 4
*
* CHECK THE REQUEST CODE
*
  00023 001665        LDA EQT6,I    ACCESS REQUEST CODE
  00024 000203R       AND =B3       ISOLATE REQUEST TYPE
  00025 000201R       CPA =B1       READ REQUEST?
  00026 000041R       JMP PROC      YES, GO PROCESS READ REQUEST
*
  00027 000203R       CPA =B3       CONTROL REQUEST?
  00030 000033R       JMP CNTRL     YES, GO PROCESS CONTROL REQUEST
*
  00031 002404        CLA,INA       NO, REJECT AS ILLEGAL WRITE REQUEST
  00032 000000R       JMP IXNN,I
*
* CONTROL REQUEST.  CHECK IF IT IS A "CLEAR" CONTROL REQUEST
* IF SO, ASSUME IT WAS ISSUED BY SYSTEM, CLEAR DEVICE, AND RETURN.
*
  00033 001665  CNTRL LDA EQT6,I    ACCESS CONTROL WORD
  00034 000204R       AND =B3700    ISOLATE SUBFUNCTION
  00035 002002        SZA           "CLEAR" REQUEST?
  00036 000037R       JMP REJCT     NO, REJECT AS ILLEGAL CNTRL REQUEST
*         .
*         .           EXECUTE CODE TO CLEAR CONTROLLER
*         .
```

Figure 4-2. Non-DMS Privileged RTE Driver Example (continued)

```
*         .
  00037 000202R REJCT LDA =B2       REJECT AS ILLEGAL CONTROL REQUEST
  00040 000000R       JMP IXNN,I
*
* SET UP FOR THE DATA TRANSFER
*
  00041 001667  PROC  LDA EQT8,I    ACCESS $ OF CONVERSIONS REQUIRED
  00042 003004        CMA,INA       NEGATE FOR CONVERSION COUNTER
  00043 000164R       STA CVCTR     AND SAVE
  00044 002021        SSA,RSS       REJECT IF
  00045 000037R       JMP REJCT     NUMBER <0
  00046 001666        LDA EQT7,I    SAVE DATA BUFFER ADDRESS
  00047 000165R       STA DAPTR     FOR PXNN
*
* INITIATE A READ AND RETURN
*
  00050 000054R       JSB READ      START A READ
  00051 103700  I.1   STC SC,C      ENCODE DEVICE
  00052 002400        CLA
  00053 000000R       JMP IXNN,I    RETURN TO IOC
*
* SUBROUTINE TO INITIATE A READ
*
  00054 000000  READ  NOP           ROUTINE CONTAINING
*         .                       CONFIGURED I/O
*         .                       INSTRUCTIONS TO
*         .                       SET UP THE DEVICE
*         .                       TO INITIATE ONE READING
  00055 000054R       JMP READ,I
*
*   **********************
*   * PRIVILEGED SECTION *
*   **********************
*
* SAVE STATE OF COMPUTER AT INTERRUPT
*
  00056 000000  PXNN  NOP           PRIVILEGED SECTION ENTRY POINT
*
  00057 103100        CLF 0         TURN OFF INTERRUPT SYSTEM
*
  00060 106706        CLC 6         TURN OFF DCPC COMPLETION INTERRUPTS
  00061 106707        CLC 7
*
  00062 000167R       STA ASV       SAVE REGISTERS
  00063 000170R       STB BSV
  00064 001520        ERA,ALS
  00065 102201        SOC
  00066 002004        INA
  00067 000171R       STA EOSY
```

Figure 4-2. Non-DMS Privileged RTE Driver Example (continued)

```
*
  00070 001770        LDA MPTFL     SAVE OLD MEMORY PROTECT FLAG
  00071 000172R       STA MPFSV
  00072 002404        CLA,INA       SET MEMORY PROTECT FLAG TO OFF,
  00073 001770        STA MPTFL     BECAUSE MEMORY PROTECT IS NOW OFF
*
  00074 102100        STF 0         TURN INTERRUPT SYSTEM BACK ON
*
* CHECK FOR SPURIOUS INTERRUPT
*
  00075 000173R       LDA EQ1,I     ACCESS REQUEST LIST POINTER WORD
  00076 000205R       AND =B77777   ISOLATE REQUEST LIST POINTER
  00077 002002        SZA           IS A REQUEST IN PROGRESS?
  00100 000104R       JMP PREAD     YES, GO PROCESS INTERRUPT
*
  00101 103100        CLF 0         NO, TURN OFF INTERRUPT SYSTEM
  00102 107700  I.2   CLC SC,C      RESET CONTROLLER, AND
  00103 000114R       JMP EXIT      IGNORE SPURIOUS INTRUPT BY RETURNING
*
* PROCESS READ REQUEST
*
        000104R PREAD EQU *
*         .         LOAD IN DATA FROM DEVICE
*         .         VIA CONFIGURED I/O INSTRUCTIONS
*         .
  00104 000165R       STA DAPTR,I   STORE WORD IN DATA BUFFER
  00105 000164R       ISZ CVCTR     IS THIS THE LAST CONVERSION?
  00106 002001        RSS           NO
  00107 000147R       JMP DONE      YES, GO SET UP TO TERMINATE CALL
*
  00110 000165R       ISZ DAPTR     NO, SET UP FOR NEXT CONVERSION
  00111 000054R       JSB READ      INITIATE IT
*
* RESTORE MACHINE TO ORIGINAL STATE ON INTERRUPT
*
  00112 103100        CLF 0         TURN OFF INTRUPT SYSTEM TEMPORARILY
*
  00113 103700  I.3   STC SC,C      ENCODE DEVICE
*
  00114 000172R EXIT  LDA MPFSV     ACCESS PREVIOUS STATE OF MEM PROTECT
  00115 002002        SZA           WAS MEMORY PROTECT ON?
  00116 000127R       JMP EXIT1     NO, DO NOT TURN ON DCPC INTERRUPTS
*
  00117 001654        LDB INTBA     YES, TURN DCPC COMPLETION INTERRUPTS
  00120 000001        LDA B,I       BACK ON IF THEY WERE ON INITIALLY.
  00121 002020        SSA           ON/OFF STATUS IS INDICATED BY BIT 15
  00122 102706        STC 6         OF EACH DCPC ASSIGNMENT WORD IN THE
  00123 006004        INB           INTERRUPT TABLE
  00124 000001        LDA B,I
  00125 002020        SSA
  00126 102707        STC 7
```

Figure 4-2. Non-DMS Privileged RTE Driver Example (continued)

```
*
  00127 000171R EXIT1 LDA EOSY      RESTORE E AND O-REGISTERS
  00130 103101        CLO
  00131 000036        SLA,ELA
  00132 102101        STF 1
  00133 000170R       LDB BSV       RESTORE B-REGISTER
*
  00134 000172R       LDA MPFSV     RESTORE MEMORY PROTECT FLAG
  00135 001770        STA MPTFL     IN BASE PAGE
  00136 002002        SZA           WAS MEMORY PROTECT ON A1 INTERRUPT?
  00137 000144R       JMP EXIT2     NO
*
  00140 000167R       LDA ASV       YES, RESTORE A-REGISTER
  00141 102100        STF 0         TURN ON INTERRUPT SYSTEM
  00142 102705        STC 5         SET MEMORY PROTECT ON
  00143 000056R       JMP PXNN,I    RETURN TO POINT OF INTERRUPTION
*
  00144 000167R EXIT2 LDA ASV       NO, RESTORE A-REGISTER
  00145 102100        STF 0         TURN ON INTERRUPT SYSTEM
  00146 000056R       JMP PXNN,I    RETURN TO POINT OF INTERRUPTION
*
* THIS CODE SETS UP THE TIME OUT TO COMPLETE THE CALL
*
  00147 103100  DONE  CLF 0         TURN OFF THE INTERRUPT SYSTEM
  00150 106700        CLC SC        TURN OFF PRIVILEGED DEVICE
  00151 003400        CCA           SET TIME OUT FOR
  00152 000175R       STA EQ15,I    ONE TICK AND SET
  00153 000174R       LDA EQ4,I     BIT12 IN EQT4 SO
  00154 000176R       IOR BIT12     RTIOC WILL
  00155 000174R       STA EQ4,I     CALL CXNN ON TIME-OUT
  00156 000114R       JMP EXIT      GO TO EXIT ROUTINE
*
*   **********************
*   * COMPLETION SECTION *
*   **********************
*
  00157 000000  CXNN  NOP           COMPLETION SECTION ENTRY POINT
*
  00160 002400        CLA           NO, SET A = 0 = NORMAL POINT
  00161 001667        LDB EQT8,I    SET B = TRANSMISSION LOG
  00162 000157R       JMP CXNN,I    MAKE COMPLETION RETURN (P+1) TO IOC
*
* CONSTANT AND STORAGE AREA
*
        000000  A     EQU 0
        000001  B     EQU 1
        000000  SC    EQU 0         DUMMY I/O SELECT CODE NUMBER
```

Figure 4-2. Non-DMS Privileged RTE Driver Example (continued)

```
*
  00163          SCODE BSS 1
  00164          CVCTR BSS 1
  00165          DAPTR BSS 1
  00166          FIRST BSS 1
  00167          ASV   BSS 1
  00170          BSV   BSS 1
  00171          EOSY  BSS 1
  00172          MPFSV BSS 1
  00173          EQ1   BSS 1
  00174          EQ4   BSS 1
  00175          EQ15  BSS 1
  00176 010000   BIT12 OCT 10000
  00177 102500   LIA   LIA 0
*
* BASE PAGE COMMUNICATIONS AREA DEFINITION
*
         001650  .     EQU 1650B
         001654  INTBA EQU .+4
         001660  EQT1  EQU .+8
         001663  EQT4  EQU .+11
         001665  EQT6  EQU .+13
         001666  EQT7  EQU .+14
         001667  EQT8  EQU .+15
         001774  EQT15 EQU .+84
         001770  MPTFL EQU .+80
*
* CODE TO SET UP JSB $JPNN,I INSTRUCTION ON BASE PAGE
*
  00000                ORB           RESET LOCATION COUNTER TO BASE PAGE
  00000 000056R $JPNN DEF PXNN       PRIV. SECTION ENTRY POINT ADDR
*
  00200                ORR           RESET LOC COUNTER TO RELOCATABLE
  00200 000000B $JSB   JSB $JPNN,I   JSB INSTR. TO PRIV SECTION, INDIRECT

*
  00201 000001
  00202 000002
  00203 000003
  00204 003700
  00205 077777
  00206 167777
                       END
```

Figure 4-2. Non-DMS Privileged RTE Driver Example (continued)

Index

R

S

T

U

W