RTE Operating System

Driver Writing Manual

This manual reflects information that is compatible with
software revision code 1805.

HEWLETT hp PACKARD

HEWLETT-PACKARD COMPANY
11000 WOLFE ROAD, CUPERTINO, CALIFORNIA, 95014

Library Index Number
2RTE.340.92200-93005

PART NO. 92200-93005

Printed in U.S.A. 10/78

PUBLICATION NOTICE

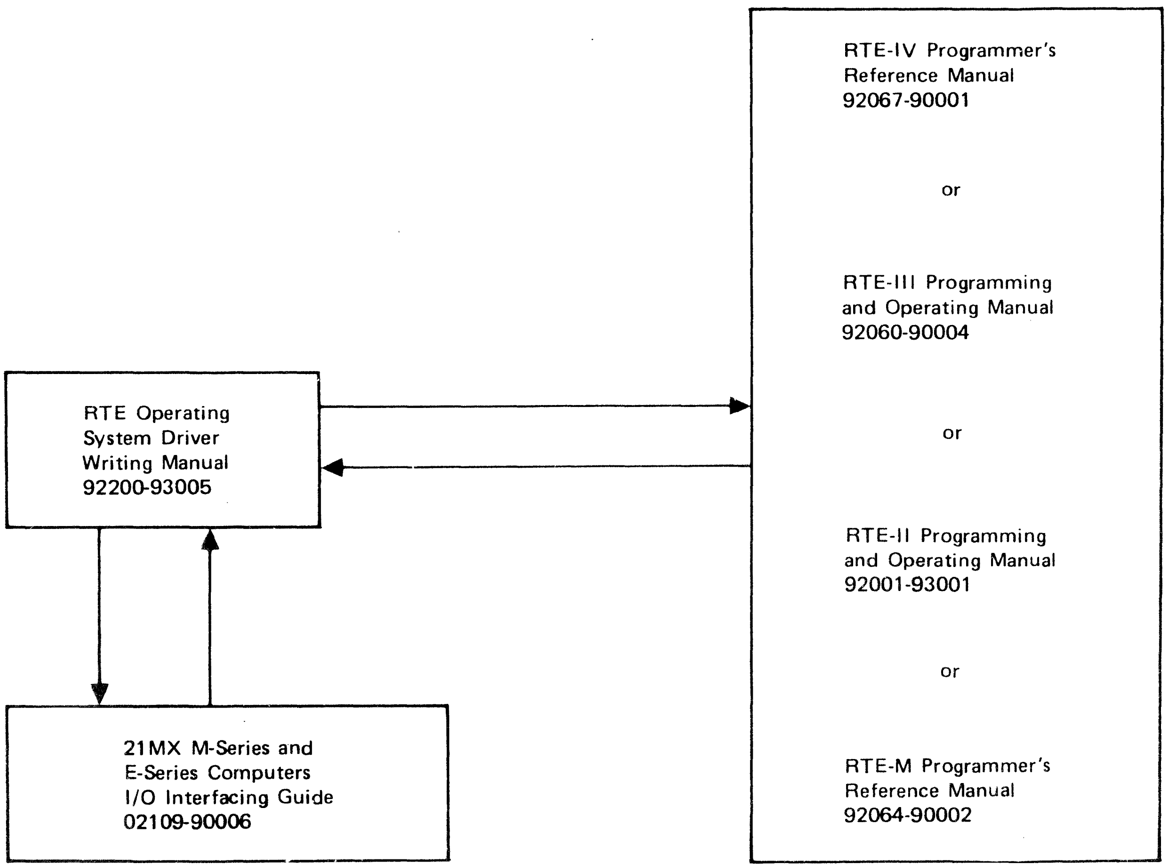
Information in this manual describes RTE driver writing techniques. Changes in text to document software updates subsequent to the initial release are supplied in manual update notices and/or complete revisions to the manual. The history of any changes to this edition of the manual is given below under "Publication History." The last change itemized reflects the software currently documented in the manual.

Any changed pages supplied in an update package are identified by a change number adjacent to the page number. Changed information is specifically identified by a vertical line (revision bar) on the outer margin of the page.

PUBLICATION HISTORY

Third Edition .... May 78 (Software Rev. Code 1805)
Change 1 .... Oct 78 (Software Rev. Code 1805)

DOCUMENTATION MAP



(Refer to the Documentation Map in the appropriate operating system manual for a complete list of related manuals.)

CONTENTS

ILLUSTRATIONS

TABLES

INTRODUCTION

SECTION I

1-1. 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 HP 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 any detail. Each of these is described in a separate manual specific to the driver.

1-2. SCOPE

The manual first provides the reader with a general description of the input/output (I/O) characteristics of the RTE family of operating systems. The techniques and requirements for developing device drivers are then presented in subsequent sections.

Since 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 (disc-based system) and RTE-M/III (memory-based equivalent of RTE-III).

1-3. SUPPORTING DOCUMENTATION

To use this manual effectively, the reader 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.

RTE INPUT/OUTPUT STRUCTURE

SECTION II

2-1. 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, the user 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 disc 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.

2-2. 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 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 Section III. Privileged drivers are more complex and are generally used for high speed and synchronous devices that require driver interaction on each data word transferred (i.e., DCPC transfers cannot be used); these drivers are discussed in Section IV.

SYSTEM I/O PROCESSOR

The system I/O processor (RTIOC) provides the software interface between user programs that perform I/O and the drivers that actually handle the I/O operations. RTIOC 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 RTIOC and user programs is the EXEC call and its associated parameters. Communication between RTIOC and drivers is handled directly via parameters and indirectly through the various I/O tables discussed in this section.

Two general areas within RTIOC 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, etc. The RTE Assembler allows 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 (disc-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 4 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 entry

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 (e.g., DCPC, buffering, time-out, power fail, etc.). 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 RTE-IV, at reconfiguration time) and never changes during on-line operation of the system. Words marked <B> are also fixed at generation time (or, for RTE-IV, at reconfiguration time) but can be changed on-line 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.

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	R	DRIVER "CONTINUATION/COMPLETION" SECTION ADDRESS <A>														
4	<A> D	<B> B	<E> P	<E> S	<C> T	SUBCHANNEL #<C>					I/O SELECT CODE # <A>					
5	AV <F>		EQUIPMENT TYPE CODE <A>						STATUS <E>							
6	CONWD (CURRENT I/O REQUEST WORD) <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 TIME-OUT RESET VALUE <B>															
15	DEVICE TIME-OUT CLOCK <C>															

WHERE THE LETTERS IN BRACKETS (<>) INDICATE THE NATURE OF EACH DATA ITEM, AS FOLLOWS:

<A> = FIXED AT GENERATION TIME (OR, FOR RTE-IV, AT RECONFIGURATION TIME); NEVER CHANGES.

<B> = FIXED AT GENERATION TIME (OR, FOR RTE-IV, AT RECONFIGURATION TIME); CAN BE CHANGED ON-LINE.

<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.

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.

D = 1 IF DCPC REQUIRED

B = 1 IF AUTOMATIC OUTPUT BUFFERING USED

P = 1 IF DRIVER IS TO PROCESS POWER FAIL

S = 1 IF DRIVER IS TO PROCESS TIME-OUT

T = 1 IF DEVICE TIMED OUT (SYSTEM SETS TO ZERO BEFORE EACH I/O REQUEST)

SUBCHANNEL # = LAST SUBCHANNEL ADDRESSED

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

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 = DISABLED (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 EQUIPMENT TYPE CODES ARE:

00 TO 07	=	PAPER TAPE DEVICES OR CONSOLES
00	=	TELEPRINTER (OR KEYBOARD CONTROL DEVICE)
01	=	PHOTOREADER
02	=	PAPER TAPE PUNCH
05	=	264X 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	=	MAGNETIC TAPE/MASS STORAGE DEVICES
23	=	9 TRACK MAGNETIC TAPE
31	=	7900 MOVING HEAD DISC
32	=	7905/06/20 MOVING HEAD DISC
33	=	FLEXIBLE DISC DRIVES
36	=	WRITABLE CONTROL STORE
37	=	HPIB
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 BELOW).

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits 15–14:
- 00 = STANDARD CALL
- 01 = BUFFERED CALL
- 11 = CLASS CALL

SUBFUNCTION (bits 10–6):
- 00000 = CLEAR CONTROLLER (IF FUNCTION = 11 = CONTROL CALL)

OTHER SUBFUNCTIONS ARE DRIVER SPECIFIC AND MAY OR MAY NOT BE DEFINED.

FUNCTION (bits 1–0):
- 01 = READ CALL
- 10 = WRITE CALL
- 11 = CONTROL CALL

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

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 (since 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.

For programming convenience, the addresses of each word 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. 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.

LOGICAL UNIT NUMBERS

Logical unit numbers (LU's) 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 on-line by the LU operator request. This scheme allows the programmer to reference changeable logical units instead of fixed physical units.

The function of Logical Units 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 disc subchannel in disc-based systems)

3 — reserved for system (auxiliary disc subchannel in disc-based systems)

4 — standard output device

5 — standard input device

6 — standard list device

Logical Unit 8 is recommended for the magnetic tape device, if one is present in the system. Peripheral discs must be assigned logical units greater than 6. Additional logical units may be assigned for any functions desired.

DEVICE REFERENCE TABLE

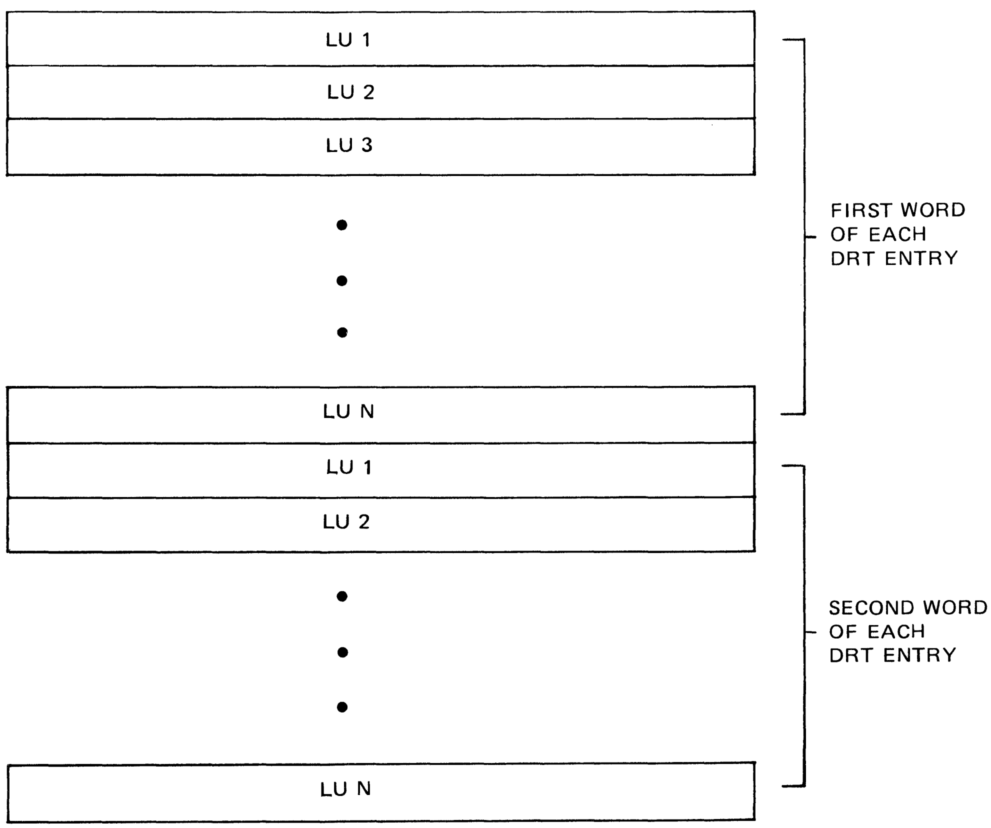
The Device Reference Table (DRT) is part of the mechanism by which logical unit numbers for I/O are implemented. RTE users request I/O by specifying a logical unit number. The DRT is used to translate this logical unit number into a physical device, as specified by an EQT entry number and subchannel. 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 (i.e., up) devices originates from word 1 of the EQT entry as illustrated in Figure 2-1.)

Each DRT entry is two words long. 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 first 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.

15–11	10–6	5–0	BIT NO.
SUBCHANNEL NO.	(RESERVED)	EQT ENTRY NUMBER	WORD 1
F (bit 15) / DOWNED I/O REQUEST LIST POINTER (bits 14–0)			WORD 2

F (UP/DOWN FLAG) = 0 IF DEVICE IS UP
= 1 IF DEVICE IS DOWN

Figure 2-3. Device Reference Table Entry Format



WHERE:

N = NUMBER OF LOGICAL UNITS IN SYSTEM

Figure 2-4. Device Reference Table

The DRT table is actually split into two separate parts. The first part contains word 1 of each DRT entry, and the second part contains word 2 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.

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 cells. 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. 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.

Since 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 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, state of the memory protect fence, etc.

Specifically, I/O instructions and NOP instructions must not be put into trap cells because they do not provide any way to restore the system to its original state. Microcode macro's (i.e., jumps to microcoded routines) may be used if a microcoded driver is used to process the interrupts.

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 power fail/automatic restart feature have a HALT 4 instruction inserted by the generator into the power fail interrupt trap cell (memory location 4). As a result, the computer will halt when a power fail 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 action 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 boot-up (e.g., RTE-IV) 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 of the entry 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.

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 (power fail interrupts, memory protect violations, etc.) 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

The reader should 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.

WORD 1	ENTRY FOR I/O SELECT CODE 6
WORD 2	ENTRY FOR I/O SELECT CODE 7
WORD 3	ENTRY FOR I/O SELECT CODE 10
WORD 4	ENTRY FOR I/O SELECT CODE 11

WORD N-4	ENTRY FOR I/O SELECT CODE N-1
WORD N-5	ENTRY FOR I/O SELECT CODE N

WHERE:

N = THE HIGHEST I/O SELECT CODE KNOWN TO THE SYSTEM.

Figure 2-5. Interrupt Table

DRIVER MAPPING TABLE (RTE-IV ONLY)

In the RTE-IV Operating System, 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. (For more information on driver placement, refer to the appropriate operating system reference and/or generation manual.)

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 driver 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 later in this manual.) 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 disc resident program, memory resident program, or system area. If a disc resident program is making the request and the I/O request buffer is located within the program (i.e., an unbuffered request), word 2 also indicates the physical memory page number of the disc 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.

		15	14 13 12 11 10	9 8 7 6 5 4 3 2 1 0
WORD 1 OF DMT ENTRY FOR EQT ENTRY:	1	SD	(RESERVED)	M
	2	SD	(RESERVED)	M
	3	SD	(RESERVED)	M
	•			
	•			
	•			
	N	SD	(RESERVED)	M
WORD 2 OF DMT ENTRY FOR EQT ENTRY:	1	MR	(RESERVED)	P
	2	MR	(RESERVED)	P
	•			
	•			
	•			
	N	MR	(RESERVED)	P

WHERE:

SD = 0 IMPLIES DRIVER RESIDES IN A DRIVER PARTITION, AND
M = STARTING PAGE NUMBER OF PARTITION IN BITS 0-9

SD = 1 IMPLIES DRIVER RESIDES IN SYSTEM DRIVER AREA, AND
M = 0 IMPLIES DRIVER NOT DOING ITS OWN MAPPING
M = 1 IMPLIES DRIVER DOING ITS OWN MAPPING

MR = 1 IMPLIES THAT THE I/O REQUEST BUFFER IS LOCATED IN A MEMORY RESIDENT PROGRAM.
(P VALUE NOT SIGNIFICANT – RESERVED FOR FUTURE USE)

MR = 0 IMPLIES THAT THE I/O REQUEST BUFFER IS NOT LOCATED IN A MEMORY RESIDENT PROGRAM. BUFFER LOCATION IS INDICATED BY THE VALUE OF P, AS FOLLOWS:
P = 0 IMPLIES BUFFER IS IN THE SYSTEM AREA
P NOT ZERO IMPLIES BUFFER IS LOCATED IN A DISC RESIDENT PROGRAM. P IS THE PHYSICAL PAGE NUMBER OF THE PROGRAM'S BASE PAGE

N = NUMBER OF EQT ENTRIES IN SYSTEM

Figure 2-6. Driver Mapping Table

2-3. 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.)

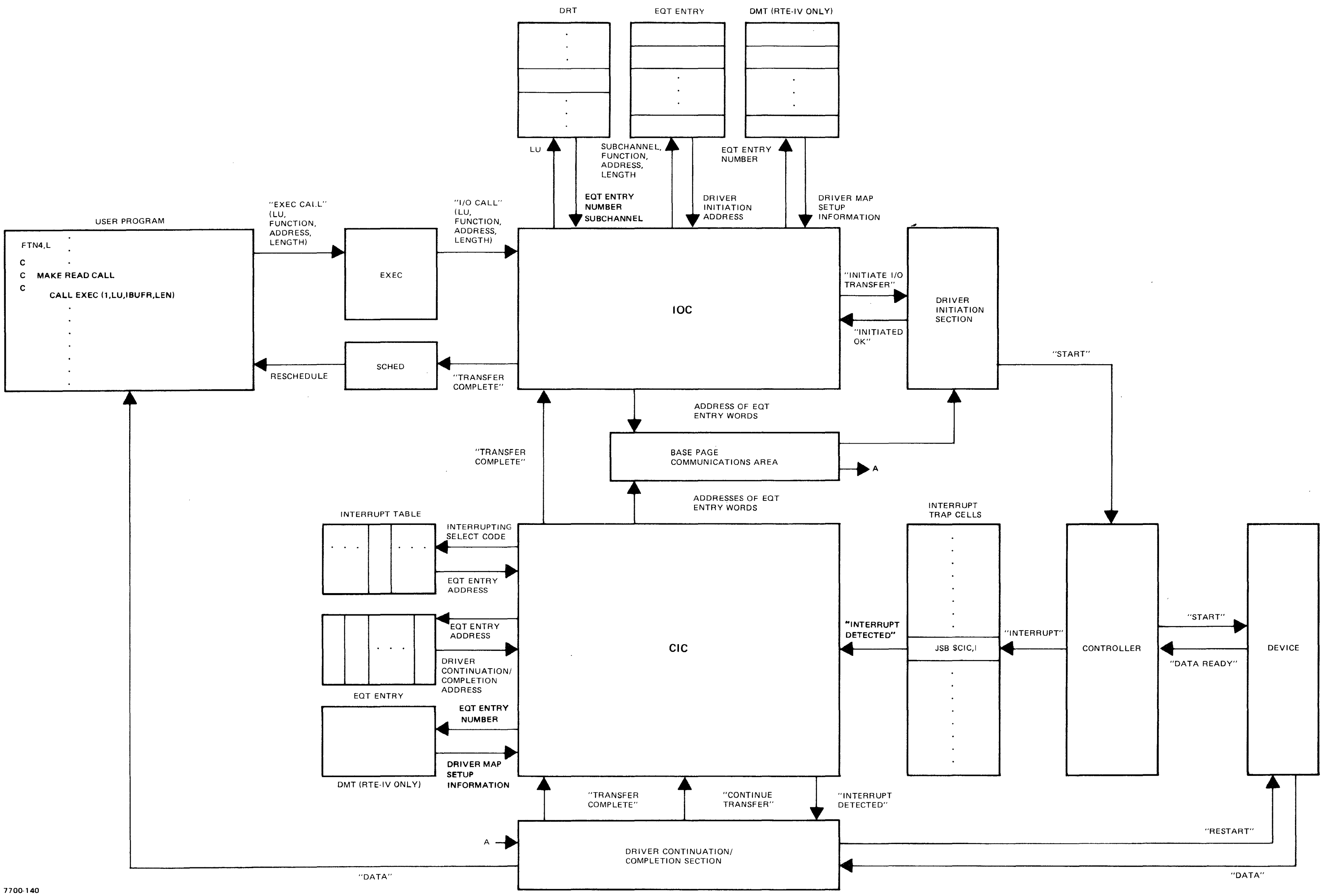


Figure 2-7. Unbuffered I/O READ Request

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.

If the device controller is available (i.e., 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 for convenience, 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 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.

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 I/O 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.

WRITING STANDARD RTE DRIVERS

SECTION III

3-1. 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. Section IV of this manual describes the differences in the design of privileged drivers versus standard drivers. Thus, if the user wishes to design a privileged driver, the material in this section should be read and understood before continuing with the privileged driver discussion in Section IV.

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. The reader should keep this requirement in mind when deciding upon the type of driver to be written.

3-2. 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.

3-3. 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 DVR07 has entry points I.07 and C.07. A driver named DVA07 has entry points IA07 and CA07.

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.

3-4. INITIATION SECTION

The IOC module of RTE calls the driver initiation section when an I/O transfer is initiated. Prior to actually entering the driver, it sets up all the 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 addresses 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 select code 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.

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.)

After performing these tasks, IOC enters the driver directly via a jump subroutine to the initiation entry point **I*xnn*** (**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 **I*xnn*** 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 and 14 must not be altered. 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 set a time-out value for the device. (Refer to the "I/O Controller Time-Out" subsection in this manual.)

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 3-1. A more detailed description of the initiation section functions is given below.

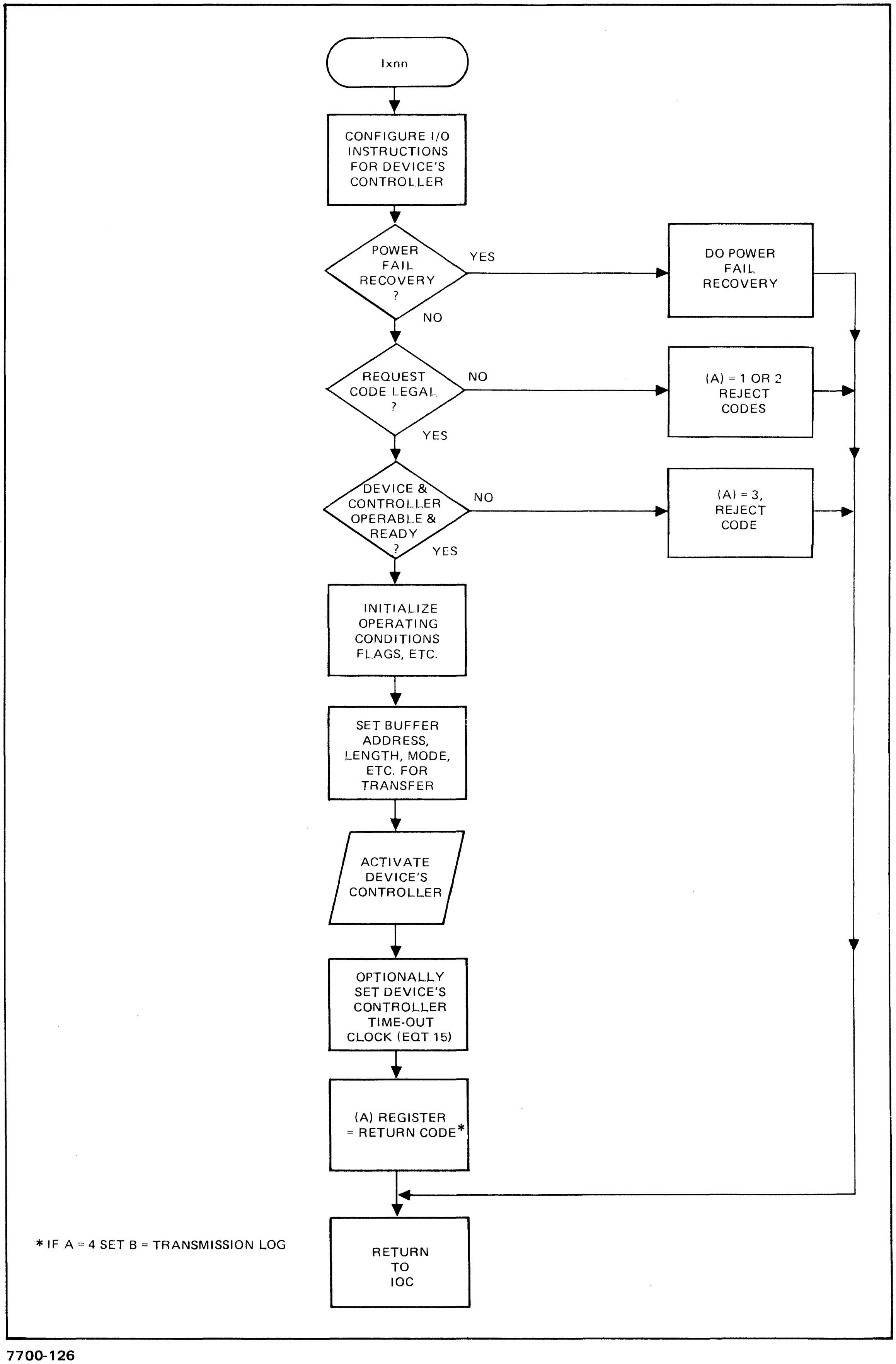


Figure 3-1. I/O Driver Initiation Section

a. Checks for power fail/automatic 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 power fail/automatic restart processing should be done. This check need only be made by drivers that are designed to process power fail interrupts (as described in the "Power Fail Processing" subsection of this manual).

b. Rejects the request by following the procedure described in step "g" 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.

c. 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.

d. Initializes DCPC, if used. (Refer to the "DCPC Processing" subsection of this manual.)

e. 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.

f. Optionally sets the device controller time-out clock (EQT entry word 15) to modify the time-out value inserted there by the system. (Refer to the "I/O Controller Time-Out" subsection of this manual.)

g. 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:

If A = 0	the operation was initiated successfully.
If A = 1,2,3	The operation was rejected, where: 1 = read or write illegal for device 2 = control request illegal or undefined, 3 = equipment malfunction or not ready
If A = 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.
If A = 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.)

If A = 6 – 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 user I/O requests only. This therefore excludes the use of return codes 6 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 06 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
6 - 59	HP system modules and drivers
60 - 99	user written drivers

3-5. CONTINUATION/COMPLETION SECTION

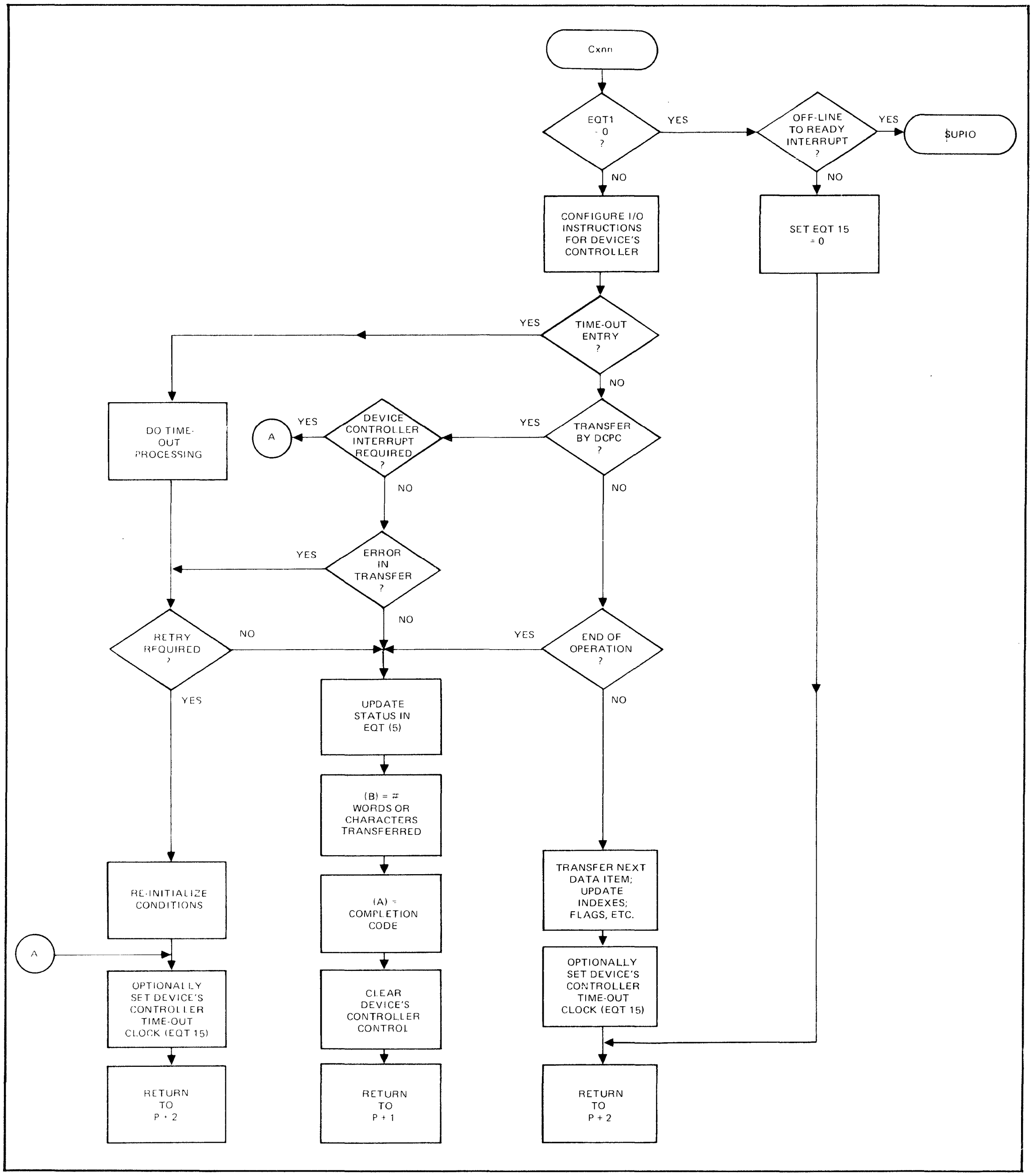
The CIC module of RTE calls the continuation/completion section of a driver whenever an interrupt is recognized on an I/O controller associated with the driver. Before calling the driver to process the interrupt, CIC issues a clear flag instruction (CLF) to the interrupting select code, sets the addresses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

The return points from **C*xnn*** to CIC indicate whether the transfer is continuing (i.e., further interrupts are expected from the device controller) or has been completed.

The continuation/completion section of the driver is flowcharted in Figure 3-2 and performs the following functions. Note that steps "a" through "e" 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 "f" (completion return) or step "g" (continuation return) respectively.



7700-127

Figure 3-2. I/O Driver Continuation/Completion Section

a. 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 (i.e., no I/O operation was in progress on the controller). The driver ignores the interrupt, sets EQT entry word 15 (the time-out clock) to zero to prevent time-out, and makes a continuation return as described in step "g" below.

b. If the interrupt is valid (i.e., 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.

c. Checks to see if a time-out 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 time-out processing should be done. Note that this check need only be made by drivers that are designed to process time-out interrupts (as described in the "I/O Controller Device Time-Out" subsection of this manual). Drivers not processing time-out are not entered on device time-out.

d. 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 "g" below. (Refer to the "DCPC Processing" subsection of this manual for a method to suppress the DCPC interrupt entirely.)

e. 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:

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

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

    3. 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 "g" below.

f. (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:

    1. 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.

    2. 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.

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

    4. Sets the A-register to indicate successful or unsuccessful 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
2 = end-of-tape or end-of-information
3 = transmission parity error
4 = device time-out

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

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

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

2. Optionally sets the device time-out clock (EQT entry word 15) to modify the value inserted there by the system. (Refer to the "I/O Controller Time-Out" subsection of this manual.)

3. Returns to CIC at P+2 as follows (the registers are not significant):

ISZ C*xnn*
JMP C*xnn*,I

3-6. 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 (i.e., 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 (i.e., 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 equivalently 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 one second time-out 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 one second time-out, 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 time-out even if it had previously set the "driver processes time-out" bit in EQT entry word 4. (Refer to the "I/O Controller Time-Out" subsection in this manual.)

3-7. I/O CONTROLLER TIME-OUT

Each I/O controller can have a time-out 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 time-out, 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 time-out 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 (i.e., 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:

a. 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. Word 14 can be preset to "m" by entering a time-out value during the EQT entry phase of generation, or it can be set or modified on-line with the TO operator request.

b. 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 exception: if an initiation call is being made and word 15 is already non-zero, it is not reset. In any case, a time-out value inserted by the driver directly into word 15 overrides any value previously set by the system.

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

DRIVER PROCESSING OF TIME-OUT

When a controller times-out, a driver has the option of performing its own time-out processing, or of letting the system handle it entirely. A driver that processes its own time-outs indicates this to the system by setting bit 12 of EQT entry word 4. Since 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 time-out:

a. Bit 11 in EQT entry word 4 is set by the system.

b. 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.)

c. The driver recognizes that the entry is for time-out by examining bit 11 of EQT entry word 4. When bit 11 is set, a time-out 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 prior to completion of the operation. This enables the driver to distinguish between a normal continuation entry (bit 11 = 0) and another time-out entry (bit 11 = 1). Note that IOC only clears bit 11 prior to entering the driver at **I*xnn*** on an initiation call.

d. The driver may decide to continue (i.e., restart the device) or complete (i.e., terminate) the operation, as follows:

   1. If the driver decides to complete the operation, it sets the A-register equal to 4 (to indicate that a time-out 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 subchannel associated with this LU.

      It is also 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.

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

SYSTEM PROCESSING OF TIME-OUT

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

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

b. 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.

c. A clear control (CLC) 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 time-out 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 time-out need to check for a time-out entry.

Since 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.

3-8. DCPC PROCESSING

The Dual Channel Port Controller (DCPC) feature of the HP 21XX 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 the reader is 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 3-3. 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.)

DCPC Channel 1 Assignment Word	Interrupt Table Word 1 (I/O Select Code 6)
DCPC Channel 2 Assignment Word	Interrupt Table Word 2 (I/O Select Code 7)

Where each DCPC Channel Assignment Word has the format:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Bit No.
F	Address															

Where:

F = 1, if the driver assigned to the channel needs the DCPC completion interrupt (set only in systems with a privileged interrupt card).

= 0, otherwise

Address = the address of the EQT entry of the driver to which the DCPC channel is assigned.

= 0, if the DCPC channel is currently not assigned.

Figure 3-3. DCPC Channel Assignment Words

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.

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 since it is not available in CHAN on later entries to the driver for the same I/O request.

ALTERNATE METHOD

If a driver needs a DCPC channel only for a certain subset of the functions that it performs, it can 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 hence the "DCPC channel required" bit is not set in the EQT entry. Thus, the driver is entered without a DCPC channel being assigned on I/O initiation. 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 (i.e., 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 to this driver?
      JMP  CH2        Yes, configure and initiate transfer on channel 2
      LDA  =B5        No. A DCPC channel is not assigned. Set
      JMP  I.xnn,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.

This method of obtaining a DCPC channel is more complex 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.

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.

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 above), 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 above), 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
JMP C*xnn*,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.

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 Section IV.)

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 had 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 to inform RTE that the standard driver actually needs the interrupt, and that RTE should reenable 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 reenable 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 reenable 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 (i.e., no Privileged Interrupt card is present); otherwise the system is privileged (i.e., 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
          INB

          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 following special processing is required: the software flag in bit 15 of the appropriate DCPC channel assignment word should be cleared prior to beginning the non-DCPC operations. This prevents the system from reenabling 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 preceeding paragraphs.

3-9. DRIVER AUTOMATIC "UP"

A driver has the capability of automatically returning its EQT entry and all associated LU's to the "up" state through a JMP instruction. For example, if a driver makes a not ready, parity error, end-of-information, or time-out 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 time-out) entry that signals that the controller is now ready, it may return the EQT entry and associated LU's to the "up" state as follows:

```
     JMP  $UPIO
```

The device controller's EQT entry and all associated LU's 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 **Lx*nn***.

3-10. POWER FAIL PROCESSING

When an RTE system is generated, the user has the option of including DVP43, the power fail/automatic 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 power fail/automatic restart modules are included in the generation, they enable the system to recover automatically from a power failure. Power fail/automatic restart processing can be divided into three parts:

a. The power down sequence.

b. The power up sequence.

c. The sequence required to restart any I/O transfers that were in progress when the power fail 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 power fail occurs, a power fail 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:

a. Stops all DCPC transfers.

b. Saves all user accessible registers (A,B,E,O . . .).

c. Saves the status of the memory protect fence. Also saves the status of the Dynamic Mapping System (DMS) in systems which include the DMS feature.

d. 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 HALT 4 instruction before power fails completely.

POWER UP SEQUENCE

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

a. Sets a software flag to prevent resaving the state of the machine if a subsequent power failure occurs before the system is completely restored.

b. Reenables the power fail hardware.

c. Restores the state of the memory protect fence. Also restores all map registers and the status of the Dynamic Mapping System in systems which include the DMS feature.

d. Saves the time of the power fail.

e. Finds the power fail EQT entry (i.e., the EQT entry associated with DVP43) and sets up a very short time out on this EQT entry by setting EQT entry word 15 (the time-out clock) to −1. This causes DVP43 to be reentered 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.

f. Restarts the real-time clock.

g. Restores all user-accessible registers.

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

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

RESTART I/O SEQUENCE

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

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

   1. 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.

   2. Checks bit 13 of EQT entry word 4 to see if the driver associated with the EQT entry is prepared to process a power fail/automatic restart entry. Drivers that are prepared to process power fail/automatic 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.)

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

b. 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 time-out" bit set:

   If the controller was reading or writing data when the power fail occurred and the driver is designed to handle power fail, the driver has the responsibility to recover from the power fail in the best possible manner. The system simply sets bit 15 of EQT entry word 5 to 1 to indicate that a power fail 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 LU's. (See the "Driver Automatic Up" subsection of this manual.) $UPIO restarts any I/O requests that were in progress (i.e., EQT entry was busy) or pending (i.e., 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 reenabling all terminals).

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

3-11. 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 is to be stored in the ID segment of the program to be scheduled.

Method 1. (Used to put five parameters in the ID segment and then schedule the program.)

```
          EXT  $LIST
                .
                .
                .
          JSB  $LIST
          OCT  701
          DEF  RTN      Return point. (Must be immediately after the call)
          DEF  PNAME    Address of three 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 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 (0-5) 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.

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.)

```
        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 (0-5) 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. (Used to put a value into the "B-register at suspension" word in the ID segment and then schedule 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 is non-zero	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.

3-12. 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 one-word table, as illustrated in Table 3-1.

Table 3-1. $OPSY Word Format

System	$OPSY Value*	Bit 15 1=RTE	Bit 3 Type	Bit 2 0=RTE-M 1=RTE	Bit 1 0=No DMS 1=DMS	Bit 0 0 =64 Word Disc 1=128 Word Disc
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
*Note: All unspecified bits are set to 1.						

$OPSY can be referenced simply 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 Systems ($OPSY bit 1 = 1), 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           Position Bit 12 into Bit 0
SLA           System Map Enabled?
JMP USER      No, User Map Enabled.
JMP SYSTM     Yes, System Map Enabled
```

3-13. SUBROUTINES FOR SPECIAL MAPPING FUNCTIONS (DMS SYSTEMS ONLY)

By using the Dynamic Mapping System (DMS) feature of the 21MX 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 (i.e., 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, the system loads and enables the correct map (System or User) needed to describe the buffer of the request. However, when a driver is entered with the System Map enabled (e.g., when the buffer is in System Available Memory), it may also need to reference a second buffer located in the user program. Subroutine $XDMP can be called by standard drivers to reload the User Map to describe the desired program. The driver can then access the second buffer. After all accesses have been made, the driver restores the User Map to its original contents and continues with its normal operation under the System Map. The recommended procedure for using $XDMP is somewhat different in RTE-III and RTE-IV, and each of these procedures is discussed in a separate section below.

NOTE

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 Section IV 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 switch to operation under this map and access the memory described by it. After all accesses have been made, the driver reenables 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 Malping System (DMS) status
          ALF                Position bit 12 into bit 0
          SLA                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
               .

          LDA MAPAD          Access address of User Map storage area
          USA                Restore original contents of User Map
          JRS  DMSST NXT     Restore original DMS status (i.e., System Map)

NXT            .
               .             Proceed with normal processing under System Map
               .
MAPAD     DEF MAP            Addrees 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
```

When called, $XDMP checks to see if the referenced program is resident in memory. If it is not, the User Map registers are not reloaded and the A-register is zero on return. If the program is resident in memory, the User Map registers are reloaded to describe the program and the A-register is non-zero on return.

Remember that any driver using this routine must save the original contents of the User map and the DMS status before calling $XDMP. It must also restore the original User Map and DMS status before continuing with its normal operation under the System 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 is the recommended procedure for using $XDMP in an RTE-IV 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

Drivers that do their own mapping in RTE-IV must have the "M" ("driver does its own mapping") option specified at generation time (during the EQT entry definition phase). Selection of the "M" option causes the driver to be placed in the System Driver Area (SDA) of memory and sets the "M" bit in the driver's DMT (Driver Mapping Table) entry. Drivers for which the "M" bit is set are always entered with the System Map enabled, regardless of the location of the buffer specified in the I/O request. If such a driver needs to access a buffer within the calling program, it can call subroutine $XDMP to reload the User Map registers to describe the program.

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 calling 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 since drivers in the SDA are not necessarily included in all users' maps. After all accesses have been made, the driver restores the original state of the User Map registers before continuing with its normal processing.

Note that subroutine $XDMP need only be used when the driver is entered with the System Map enabled but needs to access a buffer in a user program. Furthermore, $XDMP can only be called by drivers resident in the System Driver Area of an RTE-IV system. The following example assumes that both of these conditions are true.

```
           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 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
                .
                .          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
```

When called, $XDMP checks to see if the referenced program is resident in memory and whether or not the driver making the call resides in the System Driver Area (SDA). if either condition is not met, the User map registers are not reloaded and the A-register is zero on return. If the program is resident in memory and the driver is in the SDA, the User Map registers are reloaded to describe the program and the A-register is non-zero on return.

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.

NOTE

Before a standard driver in the SDA is entered to process a user program I/O request, the system automatically reloads the User Map registers to describe the calling program. This is done regardless of whether the driver is entered under the System Map or the User Map. Thus, if the driver is entered under the System Map, and if the second buffer that the driver wishes to access is located within the calling program, some processing time can be saved by not calling $XDMP to reload the User Map registers. A call to $XDMP is not needed in this case since the system has already reloaded the User Map registers with the correct information.

3-14. SAMPLE STANDARD RTE DRIVER

The sample driver illustrated in Figure 3-4 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.

```
PAGE 0002 #01  ** STANDARD RTE DRIVER EXAMPLE **

                   ASMB,L
       00000              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.)
*
* ***********************************
* * CONTINUATION/COMPLETION SECTION *
* ***********************************
*
* THE CONTINUATION/COMPLETION SECTION IS CALLED BY CENTRAL
```

Figure 3-4. Standard RTE Driver Example

```
PAGE 0003 #01   ** STANDARD RTE DRIVER EXAMPLE **

* 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.
*
```

Figure 3-4. Standard RTE Driver Example (Continued)

```
PAGE 0004 #01  ** STANDARD DRIVER - INITIATION SECTION **

*
*                         *********************
*                         * INITIATION SECTION *
*                         *********************
*
      00000 000000  I.70  NOP            ENTRY FROM IOC
*
      00001 016100R       JSB SETIO      CONFIGURE I/O INSTRUCTIONS FOR DEVICE
*
      00002 161665        LDA EQT6,I     GET CONTROL WORD OF REQUEST, AND
      00003 012115R       AND =B3        ISOLATE THE REQUEST TYPE
*
      00004 052116R       CPA =B1        IF REQUEST IS FOR INPUT
      00005 126000R       JMP I.70,I     THEN REJECT IT (A = 1 = ILLEGAL READ)
      00006 052117R       CPA =B2        IF REQUEST IS FOR OUTPUT
      00007 026017R       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 161665        LDA EQT6,I     ACCESS CONTROL WORD
      00011 012120R       AND =B3700     ISOLATE SUBFUNCTION
      00012 002002        SZA            "CLEAR" REQUEST?
      00013 026015R       JMP REJCT      NO, SO REJECT REQUEST AS ILLEGAL
*
      00014 106700  I.0   CLC SC         YES, CLEAR DEVICE AND RETURN
*
* REQUEST ERROR - CAUSE REJECT RETURN TO IOC
*
      00015 062117R REJCT LDA =B2        SET A = 2 FOR ILLEGAL CONTROL REQUEST
      00016 126000R       JMP I.70,I     AND RETURN (A = 2 = ILLEGAL CONT. REQ.)
*
* WRITE REQUEST PROCESSING
*
      00017 161666  D.X1  LDA EQT7,I     GET REQUEST BUFFER ADDRESS
      00020 171670        STA EQT9,I     AND SET IT AS CURRENT ADDRESS
      00021 161667        LDA EQT8,I     GET REQUEST BUFFER LENGTH
      00022 003004        CMA,INA        MAKE NEGATIVE AND
      00023 171671        STA EQT10,I    AND SAVE AS REMAINING BUFFER LENGTH
      00024 002002        SZA            IS BUFFER LENGTH = 0?
      00025 026031R       JMP D.X3       NO, PROCESS AS USUAL
      00026 062121R       LDA =B4        YES, SO MAKE IMMEDIATE COMPLETION RETURN
      00027 006400        CLB            SET TRANSMISSION LOG = 0 INTO B
      00030 126000R       JMP I.70,I     AND RETURN (A = 4 = IMMED. COMPLETION)
*
* CALL THE CONTINUATION/COMPLETION SECTION TO WRITE FIRST WORD
*
      00031 062114R D.X3  LDA P2         ADJUST RETURN ADDRESS SO WILL
      00032 072036R       STA C.70       RETURN HERE (INITIATION SECTION)
      00033 026047R       JMP D.X2       GO TO COMPLETION SECTION
*
      00034 002400  IEXIT CLA            NOW RETURN TO IOC WITH
      00035 126000R       JMP I.70,I     OPERATION INITIATED (A = 0 = OK)
*
```

Figure 3-4. Standard RTE Driver Example (Continued)

```
PAGE 0005 #01  ** STANDARD DRIVER - CONTINUATION/COMPLETION SECTION **

*
*                         *********************************
*                         * CONTINUATION/COMPLETION SECTION *
*                         *********************************
*
      00036 000000  C.70  NOP              CONTINUATION/COMPLETION ENTRY POINT
*
      00037 016100R       JSB SETIO        CONFIGURE I/O INSTRUCTIONS
*
      00040 161660        LDA EQT1,I       CHECK FOR SPURIOUS INTERRUPT
      00041 012122R       AND =B77777      ISOLATE I/O REQUEST LIST PTR (15 BITS)
      00042 002002        SZA              IS A REQUEST IN PROGRESS?
      00043 026047R       JMP D.X2         YES, GO PROCESS REQUEST
*
      00044 171774        STA EQT15,I      NO, SPURIOUS INTERRUPT-ZERO TIME-OUT CLK
      00045 036036R       ISZ C.70         ADJUST RETURN TO P+2 (CONTINUATION)
      00046 126036R       JMP C.70,I       MAKE CONTINUATION RETURN TO CIC
*
      00047 002400  D.X2  CLA              IF CURRENT BUFFER LENGTH  = 0,
      00050 151671        CPA EQT10,I      THEN GO TO STATUS
      00051 026063R       JMP I.3          SECTION. (I.E., TRANSFER DONE NOW)
*
      00052 165670        LDB EQT9,I       GET CURRENT BUFFER ADDRESS
*
      00053 135670        ISZ EQT9,I       ADD 1 FOR NEXT WORD
      00054 160001        LDA B,I          GET WORD TO BE WRITTEN TO DEVICE
      00055 135671        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 036036R       ISZ C.70         ADJUST RETURN TO P+2 (CONTINUATION)
      00062 126036R       JMP C.70,I       MAKE CONTINUATION RETURN
*
* STATUS AND COMPLETION SECTION
*
      00063 102500  I.3   LIA SC           GET STATUS WORD FROM DEVICE
      00064 012123R       AND =B77         STRIP OFF UNUSED BITS
      00065 070001        STA B            SAVE IN B TEMPORARILY
      00066 161664        LDA EQT5,I       REMOVE PREVIOUS STATUS
      00067 012124R       AND =B177400     BITS IN EQT WORD 5
      00070 030001        IOR B            OR IN NEW BITS
      00071 171664        STA EQT5,I       AND RESET INTO EQT WORD 5
*
      00072 002400        CLA              SET A = 0 = OK RETURN CODE
      00073 056121R       CPB =B4          ERROR STATUS BIT ON?
      00074 062117R       LDA =B2          YES, SET A = 2 = ERROR RETURN
*
      00075 165667        LDB EQT8,I       SET B = TRANSMISSION LOG
*
      00076 106700  I.4   CLC SC           CLEAR DEVICE CONTROLLER
*
      00077 126036R       JMP C.70,I       MAKE COMPLETION RETURN TO CIC
*
```

Figure 3-4. Standard RTE Driver Example (Continued)

```
PAGE 0006 #01  ** STANDARD DRIVER - SUBROUTINE SETIO **

*
*                          ********************
*                          * SUBROUTINE SETIO *
*                          ********************
*
* SUBROUTINE <SETIO> CONFIGURES ALL I/O INSTRUCTIONS IN DRIVER
*
 00100 000000  SETIO NOP           ENTRY POINT
*
 00101 032113R       IOR LIA       COMBINE LIA WITH I/O
 00102 072063R       STA I.3       SELECT CODE AND SET IN CODE
*
 00103 042125R       ADA =B100     CONSTRUCT OTA INSTRUCTION
 00104 072057R       STA I.1
*
 00105 042126R       ADA =B1100    CONSTRUCT STC,C INSTRUCTION
 00106 072060R       STA I.2
*
 00107 032127R       IOR =B4000    CONSTRUCT CLC INSTRUCTION
 00110 072014R       STA I.0
 00111 072076R       STA I.4
*
 00112 126100R       JMP SETIO,I   RETURN
*
```

Figure 3-4. Standard RTE Driver Example (Continued)

```
PAGE 0007 #01  ** STANDARD DRIVER - DATA AREA **

0222*
0223*                         *************
0224*                         * DATA AREA *
0225*                         *************
0226*
0227* CONSTANT AND STORAGE AREA
0228*
0229  00000          A      EQU 0         A-REGISTER
0230  00001          B      EQU 1         B-REGISTER
0231*
0232  00000          SC     EQU 0         DUMMY I/O SELECT CODE NUMBER
0233  00113 102500   LIA    LIA 0         CODE FOR LIA INSTRUCTION
0234  00114 000033R  P2     DEF IEXIT-1   RETURN POINT IN INITIATION SECTION
0235*
0236* ** BASE PAGE COMMUNICATIONS AREA DEFINITIONS **
0237*
0238  01650                 EQU 1650B
0239*
0240  01660          EQT1   EQU .+8
0241  01661          EQT2   EQU .+9
0242  01662          EQT3   EQU .+10
0243  01663          EQT4   EQU .+11
0244  01664          EQT5   EQU .+12
0245  01665          EQT6   EQU .+13
0246  01666          EQT7   EQU .+14
0247  01667          EQT8   EQU .+15
0248  01670          EQT9   EQU .+16
0249  01671          EQT10  EQU .+17
0250  01672          EQT11  EQU .+18
0251  01771          EQT12  EQU .+81
0252  01772          EQT13  EQU .+82
0253  01773          EQT14  EQU .+83
0254  01774          EQT15  EQU .+84
0255*
0256*
      00115 000003
      00116 000001
      00117 000002
      00120 003700
      00121 000004
      00122 077777
      00123 000077
      00124 177400
      00125 000100
      00126 001100
      00127 004000
0257                        END
**  NO ERRORS *TOTAL **RTE ASMB 760924**
```

Figure 3-4. Standard RTE Driver Example (Continued)

WRITING PRIVILEGED RTE DRIVERS

SECTION IV

4-1. INTRODUCTION

Peripheral devices that are synchronous in nature, or that generate interrupts at very high rates, 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 any 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 any 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 card, be present in the system. The Privileged Interrupt 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 (i.e., 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 (i.e., lower priority) select codes and are known as non-privileged devices.

Systems with Privileged Interrupt 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 card can be any standard I/O card that contains the normal control and flag flip-flop circuitry. Because of the position of the Privileged Interrupt 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 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:

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

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

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

d. Reenables the interrupt system and enters the driver to process the interrupt.

The above means that a privileged system processes standard (i.e., non-privileged) interrupts and requests for system functions with the interrupt system in a hold-off state, rather than with the interrupt system disabled (as it does in non-privileged systems). The privileged interrupts are always enabled and 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 card and reenables 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.

To eliminate the system overhead associated with processing interrupts, the contents of the interrupt trap cells for the privileged device controllers are changed from a JSB LINK,I instruction (where LINK contains the address of the entry point to CIC), to a JSB $JPNN,I instruction (where $JPNN contains the address of the entry point of the privileged interrupt routine). When a privileged interrupt occurs, the privileged routine is entered directly, rather than from CIC. This eliminates the system overhead. The tradeoff is that the privileged driver must be somewhat more complex than a standard driver, since it must perform some of the housekeeping duties normally handled by CIC.

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 zero in base page word DUMMY.

In general, privileged drivers are very similar to standard drivers; thus most of the material presented in Section III for standard drivers also applies to privileged drivers. Since only the differences are pointed out in this section, the reader should be familiar with the material presented in Section III before continuing with the privileged driver considerations described below.

4-2. GENERAL PRIVILEGED DRIVER STRUCTURE AND OPERATION

Privileged drivers are responsible for the initiation, continuation, and completion of all I/O requests for privileged devices. Since 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 a privileged device is made. This call has the same format as the call to the initiation section of a standard driver (see Section III of this manual).

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 or writing the next data word and restarting the device. Since 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:

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

b. 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.

c. The interrupt trap cell for the privileged device controller is changed by the privileged driver from a JSB LINK,I instruction (where LINK contains the address of the entry point to CIC) to a JSB $JPNN,I instruction (where $JPNN contains the address of the privileged section entry point, **P*xnn***).

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

e. 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.

f. 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 time-out, which causes IOC to reenter the driver at **C*xnn***.

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

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

4-3. 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.)

a. Checks for power fail/automatic 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 power fail/automatic restart processing should be done. This check need only be made by drivers that are designed to process power fail interrupts (as described in the "Power Fail Processing" subsection of this manual).

b. 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 since there is one privileged driver for each privileged device controller in the system. (See the "Privileged Driver Design Considerations" subsection of this manual.)

c. Clears bit 12 in EQT entry word 4 if time-outs 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.

d. Rejects the request and follows the procedure described in step "f" 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.

e. 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 of this manual.)

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

   If A = 0 the operation was initiated successfully.

   If A = 1,2,3 the operation was rejected, where:

   1 = read or write illegal for device
   2 = control request illegal or undefined,
   3 = equipment malfunction or not ready

If A = 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.

If A = 5 this return must NOT be used by privileged drivers.

If A = 6 − 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 6 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 06 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
6 - 59	HP system modules and drivers
60 - 99	user written drivers

Before returning to IOC, the initiation section modifies the trap cell for the privileged device controller to contain a JSB $JPNN,I instruction, where $JPNN contains the address of the privileged section entry point **P*xnn***. This causes all interrupts from the privileged device controller to be directed immediately to the privileged section of the driver. The driver needs to perform this step only once since the contents of the trap cell are not modified by any other program or system routine.

In setting up the trap cell, the driver must be sure that it is operating with the System Map enabled so that it has access to the trap cells. This can be ensured by having the calling program reference a buffer in SYSTEM COMMON when making the I/O request to the driver. (See the "Communication with User Programs" subsection later in this manual.)

An alternative method of setting up the trap cell is to point the trap cell directly at 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. This method requires that the privileged section entry point, **P*xnn***, be declared as an entry point in the privileged driver (via the ENT pseudo-instruction).

4-4. 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:

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

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

   2. 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.

   3. 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.

   4. 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.

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

   6. 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.)

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

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

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

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

   3. Proceeds to step "e-3" below to restore the computer to its original state before exiting.

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

   1. If the transfer is not complete, the driver follows the procedure described in step "e" below to return to the suspended process at the point of interruption.

   2. If the transfer is complete, the driver follows the procedure described in step "d" below to set up for a completion return to IOC.

   3. 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 "e" below to return to the suspended process at the point of interruption.

d. 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 letting the driver time-out. To do this, the driver performs the following steps:

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

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

   3. Sets the time-out clock (EQT entry word 15) to -1 to cause a time-out 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.

   4. Sets the "driver processes time-out" bit in EQT entry word 4 to one so that the driver will be reentered when the time-out occurs.

   5. Follows the procedure described in step "e-3" below to restore the computer to its original state before exiting.

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

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

2. 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 reentered at ***Pxnn*** 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 reallow any lower priority interrupts to be recognized by the system when the interrupt system is reenabled.

3. Checks to see if either of the DCPC completion interrupts needs to be reenabled, as follows:

   If the memory protect fence was initially OFF, the driver must not reenable 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 reenable 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 reenable the interrupt.

4. Restores all saved registers to their original values.

5. 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).

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

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

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

   i. 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 ***Pxnn***. (Performed only in systems with the Dynamic Mapping feature.)

ii. 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 7, execution of the JRS instruction (in step 8) 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 reenabled before executing the JRS instruction. The explanation of map switching given in the "Subroutines for Special Mapping Functions" subsection of this manual illustrates this procedure.

4-5. COMPLETION SECTION

When the time-out 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.

4-6. 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:

a. 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 reentrant.

b. 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.

c. If a privileged driver wishes to use the EQT entry for temporary storage, 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 since 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.)

d. Since privileged drivers are required to keep information relating to the I/O request internally (see "c" 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***.

4-7. 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 will have to 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.

4-8. 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; 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 power fail interrupts nor does it process any time-outs, except for the time-out 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 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 **I*xnn***.

Since 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 since these addresses will not be available in base page on subsequent interrupts.

The modification of the trap cell is also performed just once (as part of the configuring routine) and is not modified again on any later entries into the initiation section. The trap cell is altered so that the device controller interrupts are channelled to the privileged section of the driver (**P*xnn***) instead of to CIC. The JSB $JPNN,I instruction (where $JPNN contains the address of **P*xnn***) is established by coding a JSB instruction on the base page (see listing).

The request code is checked for validity. All write requests 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 time-out" bit in EQT entry word 4 is cleared so that any unexpected time-outs are handled by the system. This bit is later reset to 1 by the privileged section when it sets up a time-out 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 (**P*xnn***) is entered directly as a result of the controller's trap cell modification.

Because entry is made directly into **P*xnn*, P*xnn*** must do the housekeeping that is normally done by CIC when a standard interrupt occurs. Thus, before **P*xnn*** can turn the interrupt system back on to allow higher priority privileged interrupts to be recognized, **P*xnn*** 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, **P*xnn*** must also save the Dynamic Mapping System status at the time of interrupt.

P*xnn* then loads and stores the data in the next unfilled buffer word. If there is yet another data point to be taken, **P*xnn*** 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, **P*xnn*** must also restore the Dynamic Mapping System status to its original value. All of this basically resets the system to its state before **P*xnn*** was entered.

When the last reading is taken, **P*xnn*** disables the interrupt system, turns off the device controller, and sets up the privileged controller's EQT so that a time-out will occur at the next tick of the real-time clock. **P*xnn*** 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 time-out 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 **C*xnn***. 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.

4-9. TIME-OUT VALUES FOR PRIVILEGED DRIVERS

If the user wishes to specify a time-out value for the privileged controller (to prevent indefinite suspension in the event that the controller malfunctions), the time-out value must be long enough to cover the entire period from I/O initiation to completion. This is different from the time-out 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 time-out clock (EQT entry word 15) to the value specified at generation. However, since privileged drivers are not entered from CIC on interrupt, the time-out value is inserted into the privileged controller's time-out clock only at initiation. If this value is not long enough to cover the entire I/O transfer period, a time-out 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 time-out value, or by specifying a time-out value of zero (which disables the time-out feature entirely).

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

4-10. SUBROUTINES FOR SPECIAL MAPPING FUNCTIONS (DMS SYSTEMS ONLY)

DMS privileged drivers normally communicate with user programs via SYSTEM COMMON or SSGA. If it is necessary to access a buffer within the user program, the privileged driver will have to reload the User Map to describe the desired program before making the access. Subroutine $PVMP can be called by privileged drivers to perform this type of map switching. The driver can then access the second buffer. After all accesses have been made, the driver restores the User Map to its original state and continues with its normal operation under the System Map. The recommended procedure for using $PVMP is somewhat different in RTE-III and RTE-IV, and each of these procedures is discussed 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 Section III 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:

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

b. 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, since the privileged section is entered directly from the interrupt trap cell. The I/O request buffer should therefore be located in SYSTEM COMMON so that the privileged section of the driver can access the buffer directly without switching maps. In the general case, the buffer may be located either in the User Map (program not swappable) or may be in the System Map. A technique is presented below to identify which map is needed.

The procedure for using $PVMP is as follows: The driver first saves the current state of the User Map registers and Dynamic Mapping System. After the driver has determined that the User Map is being used and not the System Map, it calls $PVMP to reload the user Map registers to describe the calling program. (Note that the ID segment address of this program must have been saved previously by the initiation section, since this information is not available on base page on subsequent interrupts.) The driver can then switch to operation under this map and access the memory described by it. After all accesses have been made, the driver reenables 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 XEQT           Access address of program making request
         STA IDADR          Save for use of Privileged Section later

PROCD    NOP
              .
              .
              .
```

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

         JRS DMSST NXT   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
```

When called, $PVMP checks to see if the referenced program is resident in memory. If the program is not resident, the User Map registers are not reloaded and the A-register is zero on return. If the program is resident in memory, the User Map registers are reloaded to describe the program and the A-register is non-zero on return.

Remember that any driver using this routine must save the original contents of the User Map registers and the DMS status before calling $PVMP. It must also restore the original User

Map and DMS status after all accesses to the buffer have been made. 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 is in fact 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

Privileged drivers in RTE-IV must have the "M" (driver does its own mapping) option specified at generation time (during the EQT entry definition phase). This causes the driver to be placed in the System Driver Area (SDA) of memory and causes the "M" bit to be set in the driver's DMT (Driver Mapping Table) entry. It also implies that an RTE-IV privileged driver is always entered with the System Map enabled, because:

a. When an interrupt occurs, the System Map is automatically enabled, and the privileged section of the driver is entered directly from the trap cell.

b. The initiation and completion sections of drivers for which the "M" option was selected are always entered from IOC with the System Map enabled, regardless of the location of the I/O buffer of the request.

The I/O request buffer should therefore be located in SYSTEM COMMON, so that the privileged section of the driver can access the buffer directly without switching maps. If the design of the driver is such that the privileged section must access a buffer in the calling program, subroutine $PVMP must be called to reload the User Map registers to describe the desired program.

The procedure for using $PVMP is as follows: The driver first saves the current contents of the User Map registers and then calls $PVMP to reload the User Map registers to describe the calling 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 since drivers in the SDA are not necessarily included in all users' 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.

The following code illustrates the use of subroutine $PVMP in an RTE-IV system:

```
         EXT $PVMP

IXNN     NOP                 Initiation Section entry point (System Map Enabled)
          .
          .
          .
```

```
         LDA XEQT        Access address of program making request
         STA IDADR       Save for use of Privileged Section later
              .
              .
              .
         JMP IXNN,I      Return to IOC

PXNN     NOP             Privileged Section entry point (System Map Enabled)

              .
              .          Normal privileged 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 address of program that contains buffer
         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 since 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
```

When called, $PVMP checks to see if the referenced program is resident in memory. If the program is not resident, the User Map registers are not reloaded and the A-register is zero on return. If the program is resident in memory, the User Map registers are reloaded to describe the program and the A-register is non-zero on return.

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.

4-11. SAMPLE DMS PRIVILEGED DRIVER

The sample driver illustrated in Figure 4-1 demonstrates some of the principles involved in writing a privileged I/O driver for use in an RTE system with Dynamic Mapping. Note that this driver is for tutorial purposes only and is not one of the drivers supplied with the system.

4-12. SAMPLE NON-DMS PRIVILEGED DRIVER

The sample driver illustrated in Figure 4-2 demonstrates some of the principles involved in writing a privileged I/O driver for use in an RTE system without Dynamic Mapping. Note that this driver is for tutorial purposes only and is not one of the drivers supplied with the system.

```
 PAGE 0002 #01  ** RTE DMS PRIVILEGED DRIVER EXAMPLE **

                    ASMB,L
*
       00000               NAM DVYNN   ** RTE 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.
*     THIS CAUSES THE DRIVER TO BE 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
*
* 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
*
```

Figure 4-1. DMS Privileged RTE Driver Example

```
PAGE 0003 #01  ** DMS PRIVILEGED DRIVER - INITIATION SECTION **

*
*                        *********************
*                        * INITIATION SECTION *
*                        *********************
*
      00000 000000  IXNN  NOP             INITIATION SECTION ENTRY POINT
      00001 072200R       STA SCODE       SAVE SELECT CODE OF CONTROLLER
*
      00002 066203R       LDB FIRST       ACCESS FIRST TIME THROUGH FLAG
      00003 006002        SZB             IS THIS THE FIRST TIME THRU?
      00004 026020R       JMP INIT        NO, SO SKIP CONFIGURATION CODE
*
* CONFIGURE I/O INSTRUCTIONS
*
      00005 032217R       IOR LIA         CREATE LIA INSTRUCTION
*                          .                  .
*                          .                  .
*                          .                  .
*
* MODIFY TRAP CELL
*
      00006 060000B       LDA $JSB        SET TRAP CELL TO
      00007 172200R       STA SCODE,I     JSB $JPNN,I ($JPNN = ADDR OF PXNN)
*
* SAVE EQT ADDRESSES
*
      00010 061774        LDA EQT15       SAVE EQT15
      00011 072215R       STA EQ15
      00012 061663        LDA EQT4        EQT 4
      00013 072214R       STA EQ4
      00014 061660        LDA EQT1        AND EQT1
      00015 072213R       STA EQ1         ADDRESSES
*
      00016 002404        CLA,INA         SET FLAG TO PREVENT CONFIGURING ON
      00017 072203R       STA FIRST       SUBSEQUENT INITIATIONS
*
* CLEAR THE "DRIVER PROCESSES TIME-OUT" BIT TO ALLOW
* NORMAL TIME-OUT OPERATION
*
      00020 161663  INIT  LDA EQT4,I      ACCESS EQT WORD 4
      00021 012221R       AND =B167777    CLEAR BIT 12
      00022 171663        STA EQT4,I      AND RESET EQT WORD 4
*
* CHECK THE REQUEST CODE
*
      00023 161665        LDA EQT6,I      ACCESS REQUEST CODE
      00024 012222R       AND =B3         ISOLATE REQUEST TYPE
      00025 052223R       CPA =B1         READ REQUEST?
      00026 026041R       JMP PROC        YES, GO PROCESS READ REQUEST
*
      00027 052222R       CPA =B3         CONTROL REQUEST?
      00030 026033R       JMP CNTRL       YES, GO PROCESS CONTROL REQUEST
*
      00031 002404        CLA,INA         NO, SO REJECT AS ILLEGAL WRITE REQUEST
      00032 126000R       JMP IXNN,I
*
```

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

```
PAGE 0004 #01  ** DMS PRIVILEGED DRIVER - INITIATION SECTION **

*  CONTROL REQUEST. CHECK IF IT IS A "CLEAR" CONTROL REQUEST
*  IF SO, ASSUME IT WAS ISSUED BY SYSTEM, CLEAR DEVICE, AND RETURN
*
   00033 161665  CNTRL LDA EQT6,I     ACCESS CONTROL WORD
   00034 012224R       AND =B3700     ISOLATE SUBFUNCTION
   00035 002002        SZA            "CLEAR" REQUEST?
   00036 026037R       JMP REJCT      NO, SO REJECT AS ILLEGAL CONTROL REQUEST
*
*                          .
*                          .          EXECUTE CODE TO CLEAR CONTROLLER
*                          .
*
   00037 062225R REJCT LDA =B2        REJECT AS ILLEGAL CONTROL REQUEST
   00040 126000R       JMP IXNN,I
*
*  SET UP FOR THE DATA TRANSFER
*
   00041 161667  PROC  LDA EQT8,I     ACCESS # OF CONVERSIONS REQUIRED
   00042 003004        CMA,INA        NEGATE FOR CONVERSION COUNTER
   00043 072201R       STA CVCTR      AND SAVE
   00044 002021        SSA,RSS        REJECT IF
   00045 026037R       JMP REJCT      NUMBER <0
   00046 161666        LDA EQT7,I     SAVE DATA BUFFER ADDRESS
   00047 072202R       STA DAPTR      FOR PXNN
*
*  INITIATE A READ AND RETURN
*
   00050 016053R       JSB READ       START A READ
   00051 103700  I.1   STC SC,C       ENCODE DEVICE
   00052 126000R       JMP IXNN,I     RETURN TO IOC
*
*  SUBROUTINE TO INITIATE A READ
*
   00053 000000  READ  NOP            ROUTINE CONTAINING
*                          .          CONFIGURED I/O
*                          .          INSTRUCTIONS TO
*                          .          SET UP THE DEVICE
*                          .          TO INITIATE ONE READING
   00054 126053R       JMP READ,I
```

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

```
PAGE 0005 #01  ** DMS PRIVILEGED DRIVER - PRIVILEGED SECTION **

*
*                          *********************
*                          * PRIVILEGED SECTION *
*                          *********************
*
* SAVE STATE OF COMPUTER AT INTERRUPT
*
 00055 000000  PXNN  NOP            PRIVILEGED SECTION ENTRY POINT
*
 00056 103100        CLF 0          TURN OFF INTERRUPT SYSTEM
*
 00057 106706        CLC 6          TURN OFF DCPC COMPLETION INTERRUPTS
 00060 106707        CLC 7
*
 00061 072204R       STA ASV        SAVE REGISTERS
 00062 076205R       STB BSV
 00063 001520        ERA,ALS
 00064 102201        SOC
 00065 002004        INA
 00066 072206R       STA EOSV
 00067 105743        STX XSV        SAVE X REGISTER
 00071 105753        STY YSV        SAVY Y REGISTER
 00073 105714        SSM DMSTS      SAVE DYNAMIC MAPPING SYSTEM STATUS
*
 00075 061770        LDA MPTFL      SAVE OLD MEMORY PROTECT FLAG
 00076 072212R       STA MPFSV
 00077 002404        CLA,INA        SET MEMORY PROTECT FLAG TO OFF
 00100 071770        STA MPTFL      SINCE MEMORY PROTECT IS NOW OFF
*
 00101 102100        STF 0          TURN INTERRUPT SYSTEM BACK ON
*
* CHECK FOR SPURIOUS INTERRUPT
*
 00102 162213R       LDA EQ1,I      ACCESS REQUEST LIST POINTER WORD
 00103 012226R       AND =B77777    ISOLATE REQUEST LIST POINTER
 00104 002002        SZA            IS A REQUEST IN PROGRESS?
 00105 026111R       JMP PREAD      YES, GO PROCESS INTERRUPT
*
 00106 103100        CLF 0          NO, TURN OFF INTERRUPT SYSTEM
 00107 107700  I.2   CLC SC,C       RESET CONTROLLER, AND
 00110 026121R       JMP EXIT       IGNORE SPURIOUS INTERRUPT BY RETURNING
*
* PROCESS READ REQUEST
*
 00111         PREAD EQU *
*                        .          LOAD IN DATA FROM DEVICE
*                        .          VIA CONFIGURED I/O INSTRUCTIONS
*                        .
*
 00111 172202R       STA DAPTR,I    STORE WORD IN DATA BUFFER
 00112 036201R       ISZ CVCTR      IS THIS THE LAST CONVERSION?
 00113 002001        RSS            NO
 00114 026164R       JMP DONE       YES, GO SET UP TO TERMINATE CALL
*
 00115 036202R       ISZ DAPTR      NO, SET UP FOR NEXT CONVERSION
 00116 016053R       JSB READ       INITIATE IT
```

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

```
 PAGE 0006 #01   ** DMS PRIVILEGED DRIVER - PRIVILEGED SECTION **

*
* RESTORE MACHINE TO ORIGINAL STATE ON INTERRUPT
*
      00117 103100         CLF 0          TURN OFF INTERRUPT SYSTEM TEMPORARILY
*
      00120 103700  I.3    STC SC,C       ENCODE DEVICE
*
      00121 062212R EXIT   LDA MPFSV      ACCESS PREVIOUS STATE OF MEMORY PROTECT
      00122 002002         SZA            WAS MEMORY PROTECT ON?
      00123 026134R        JMP EXIT1      NO, SO DO NOT TURN ON DCPC INTERRUPTS
*
      00124 065654         LDB INTBA      YES, TURN DCPC COMPLETION INTERRUPTS
      00125 160001         LDA B,I        BACK ON IF THEY WERE ON INITIALLY.
      00126 002020         SSA            ON/OFF STATUS IS INDICATED BY BIT 15
      00127 102706         STC 6          OF EACH DCPC ASSIGNMENT WORD IN THE
      00130 006004         INB            INTERRUPT TABLE
      00131 160001         LDA B,I
      00132 002020         SSA
      00133 102707         STC 7
*
      00134 062206R EXIT1  LDA EOSV       RESTORE E AND O REGISTERS
      00135 103101         CLO
      00136 000036         SLA,ELA
      00137 102101         STF 1
      00140 066205R        LDB BSV        RESTORE B-REGISTER
      00141 105745         LDX XSV        RESTORE X REGISTER
      00143 105755         LDY YSV        RESTORE Y REGISTER
*
      00145 062212R        LDA MPFSV      RESTORE MEMORY PROTECT FLAG
      00146 071770         STA MPTFL      IN BASE PAGE
      00147 002002         SZA            WAS MEMORY PROTECT ON AT INTERRUPT?
      00150 026157R        JMP EXIT2      NO
*
      00151 062204R        LDA ASV        YES, RESTORE A-REGISTER
      00152 102100         STF 0          TURN ON INTERRUPT SYSTEM
      00153 102705         STC 5          SET MEMORY PROTECT ON
      00154 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 THE SYSTEM MAP
*                                                IS CURRENTLY ENABLED.  THIS
*                                                DRIVER HAS BEEN DESIGNED SUCH
*                                                THAT THIS IS ALWAYS THE CASE.
*
      00157 062204R EXIT2  LDA ASV        NO,RESTORE A-REGISTER
      00160 102100         STF 0          TURN ON INTERRUPTS
      00161 105715         JRS DMSTS PXNN,I      RESTORE DMS STATUS AND RETURN
*
* THIS CODE SETS UP THE TIME OUT TO COMPLETE THE CALL
*
      00164 103100  DONE   CLF 0          TURN OFF THE INTERRUPT SYSTEM
      00165 106700  I.4    CLC SC         TURN OFF PRIVILEGED DEVICE
      00166 003400         CCA            SET TIME OUT FOR
      00167 172215R        STA EQ15,I     ONE TICK AND SET
      00170 162214R        LDA EQ4,I      BIT12 IN EQT4 SO
```

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

```
PAGE 0007 #01   ** DMS PRIVILEGED DRIVER - PRIVILEGED SECTION **

0262  00171 032216R        IOR BIT12      RTIOC WILL
0263  00172 172214R        STA EQ4,I      CALL CXNN ON TIME-OUT
0264  00173 026121R        JMP EXIT       GO TO EXIT ROUTINE
```

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

```
0266*
0267*                        *********************
0268*                        * COMPLETION SECTION *
0269*                        *********************
0270*
0271  00174 000000  CXNN  NOP           COMPLETION SECTION ENTRY POINT
0272*
0273  00175 002400        CLA           SET A = 0 = NORMAL RETURN
0274  00176 165667        LDB EQT8,I    SET B = TRANSMISSION LOG
0275  00177 126174R       JMP CXNN,I    RETURN TO IOC
0276*
```

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

```
PAGE 0009 #01   ** DMS PRIVILEGED DRIVER - DATA AREA **

0278*
0279* CONSTANT AND STORAGE AREA
0280*
0281  00000          A     EQU 0
0282  00001          B     EQU 1
0283  00000          SC    EQU 0          DUMMY I/O SELECT CODE NUMBER
0284*
0285  00200 000000   SCODE BSS 1
0286  00201 000000   CYCTR BSS 1
0287  00202 000000   DAPTR BSS 1
0288  00203 000000   FIRST BSS 1
0289  00204 000000   ASV   BSS 1
0290  00205 000000   BSV   BSS 1
0291  00206 000000   EOSV  BSS 1
0292  00207 000000   XSV   BSS 1
0293  00210 000000   YSV   BSS 1
0294  00211 000000   DMSTS BSS 1
0295  00212 000000   MPFSV BSS 1
0296  00213 000000   EQ1   BSS 1
0297  00214 000000   EQ4   BSS 1
0298  00215 000000   EQ15  BSS 1
0299  00216 010000   BIT12 OCT 10000
0300  00217 102500   LIA   LIA 0
0301*
0302* BASE PAGE COMMUNICATIONS AREA DEFINITION
0303*
0304  01650                EQU 1650B
0305  01654          INTBA EQU .+4
0306  01660          EQT1  EQU .+8
0307  01663          EQT4  EQU .+11
0308  01665          EQT6  EQU .+13
0309  01666          EQT7  EQU .+14
0310  01667          EQT8  EQU .+15
0311  01774          EQT15 EQU .+84
0312  01770          MPTFL EQU .+80
0313*
0314* CODE TO SET UP JSB $JPNN,I INSTRUCTION ON BASE PAGE
0315*
0316  00220 000055R $JPNN DEF PXNN       PRIV. SECTION ENTRY POINT ADDR
0317*
0318  00000                ORB           RESET LOCATION COUNTER TO BASE PAGE
0319  00000 116220R $JSB  JSB $JPNN,I   JSB INSTR. TO PRIV. SECTION, INDIRECT
0320*
0321                       END
**  NO ERRORS *TOTAL **RTE ASMB 760924**
```

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

```
 PAGE 0002 #01  ** RTE NON-DMS PRIVILEGED DRIVER EXAMPLE **

                      ASMB,L
*
      00000                  NAM DVYNN   ** RTE 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
*
```

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

```
PAGE 0003 #01  ** NON-DMS PRIVILEGED DRIVER - INITIATION SECTION **

*
                              **********************
                              * INITIATION SECTION *
                              **********************
*
      00000 000000  IXNN   NOP            INITIATION SECTION ENTRY POINT
      00001 072162R        STA SCODE      SAVE SELECT CODE OF CONTROLLER
*
      00002 066165R        LDB FIRST      ACCESS FIRST TIME THROUGH FLAG
      00003 006002         SZB            IS THIS THE FIRST TIME THRU?
      00004 026020R        JMP INIT       NO, SO SKIP CONFIGURATION CODE
*
* CONFIGURE I/O INSTRUCTIONS
*
      00005 032176R        IOR LIA        CREATE LIA INSTRUCTION
*                               .                    .
*                               .                    .
*                               .                    .
*
* MODIFY TRAP CELL
*
      00006 060000B        LDA $JSB       SET TRAP CELL TO
      00007 172162R        STA SCODE,I    JSB $JPNN,I ($JPNN = ADDR OF PXNN)
*
* SAVE EQT ADDRESSES
*
      00010 061774         LDA EQT15      SAVE EQT15
      00011 072174R        STA EQ15
      00012 061663         LDA EQT4       EQT4
      00013 072173R        STA EQ4
      00014 061660         LDA EQT1       AND EQT1
      00015 072172R        STA EQ1        ADDRESSES
*
      00016 002404         CLA,INA        SET FLAG TO PREVENT CONFIGURING ON
      00017 072165R        STA FIRST      SUBSEQUENT INITIATIONS
*
* CLEAR THE "DRIVER PROCESSES TIME-OUT" BIT TO ALLOW
* NORMAL TIME-OUT OPERATION
*
      00020 161663  INIT   LDA EQT4,I     ACCESS EQT WORD 4
      00021 012200R        AND =B167777   CLEAR BIT 12
      00022 171663         STA EQT4,I     AND RESET EQT WORD 4
*
* CHECK THE REQUEST CODE
*
      00023 161665         LDA EQT6,I     ACCESS REQUEST CODE
      00024 012201R        AND =B3        ISOLATE REQUEST TYPE
      00025 052202R        CPA =B1        READ REQUEST?
      00026 026041R        JMP PROC       YES, GO PROCESS READ REQUEST
*
      00027 052201R        CPA =B3        CONTROL REQUEST?
      00030 026033R        JMP CNTRL      YES, GO PROCESS CONTROL REQUEST
*
      00031 002404         CLA,INA        NO, SO REJECT AS ILLEGAL WRITE REQUEST
      00032 126000R        JMP IXNN,I
*
```

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

```
PAGE 0004 #01  ** NON-DMS PRIVILEGED DRIVER - INITIATION SECTION **

* CONTROL REQUEST. CHECK IF IT IS A "CLEAR" CONTROL REQUEST
* IF SO, ASSUME IT WAS ISSUED BY SYSTEM, CLEAR DEVICE, AND RETURN
*
  00033 161665  CNTRL LDA EQT6,I     ACCESS CONTROL WORD
  00034 012203R       AND =B3700     ISOLATE SUBFUNCTION
  00035 002002        SZA            "CLEAR" REQUEST?
  00036 026037R       JMP REJCT      NO, SO REJECT AS ILLEGAL CONTROL REQUEST
*
*                         .
*                         .          EXECUTE CODE TO CLEAR CONTROLLER
*                         .
*
  00037 062204R REJCT LDA =B2        REJECT AS ILLEGAL CONTROL REQUEST
  00040 126000R       JMP IXNN,I
*
* SET UP FOR THE DATA TRANSFER
*
  00041 161667  PROC  LDA EQT8,I     ACCESS # OF CONVERSIONS REQUIRED
  00042 003004        CMA,INA        NEGATE FOR CONVERSION COUNTER
  00043 072163R       STA CVCTR      AND SAVE
  00044 002021        SSA,RSS        REJECT IF
  00045 026037R       JMP REJCT      NUMBER <0
  00046 161666        LDA EQT7,I     SAVE DATA BUFFER ADDRESS
  00047 072164R       STA DAPTR      FOR PXNN
*
* INITIATE A READ AND RETURN
*
  00050 016053R       JSB READ       START A READ
  00051 103700  I.1   STC SC,C       ENCODE DEVICE
  00052 126000R       JMP IXNN,I     RETURN TO IOC
*
* SUBROUTINE TO INITIATE A READ
*
  00053 000000  READ  NOP            ROUTINE CONTAINING
*                         .          CONFIGURED I/O
*                         .          INSTRUCTIONS TO
*                         .          SET UP THE DEVICE
*                         .          TO INITIATE ONE READING
  00054 126053R       JMP READ,I
```

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

```
PAGE 0005 #01  ** NON-DMS PRIVILEGED DRIVER - PRIVILEGED SECTION **

*
*                        **********************
*                        * PRIVILEGED SECTION *
*                        **********************
*
* SAVE STATE OF COMPUTER AT INTERRUPT
*
 00055 000000  PXNN  NOP          PRIVILEGED SECTION ENTRY POINT
*
 00056 103100        CLF 0        TURN OFF INTERRUPT SYSTEM
*
 00057 106706        CLC 6        TURN OFF DCPC COMPLETION INTERRUPTS
 00060 106707        CLC 7
*
 00061 072166R       STA ASV      SAVE REGISTERS
 00062 076167R       STB BSV
 00063 001520        ERA,ALS
 00064 102201        SOC
 00065 002004        INA
 00066 072170R       STA EOSV
*
 00067 061770        LDA MPTFL    SAVE OLD MEMORY PROTECT FLAG
 00070 072171R       STA MPFSV
 00071 002404        CLA,INA      SET MEMORY PROTECT FLAG TO OFF,
 00072 071770        STA MPTFL    SINCE MEMORY PROTECT IS NOW OFF
*
 00073 102100        STF 0        TURN INTERRUPT SYSTEM BACK ON
*
* CHECK FOR SPURIOUS INTERRUPT
*
 00074 162172R       LDA EQ1,I    ACCESS REQUEST LIST POINTER WORD
 00075 012205R       AND =B77777  ISOLATE REQUEST LIST POINTER
 00076 002002        SZA          IS A REQUEST IN PROGRESS?
 00077 026103R       JMP PREAD    YES, GO PROCESS INTERRUPT
*
 00100 103100        CLF 0        NO, TURN OFF INTERRUPT SYSTEM
 00101 107700  I.2   CLC SC,C     RESET CONTROLLER, AND
 00102 026113R       JMP EXIT     IGNORE SPURIOUS INTERRUPT BY RETURNING
*
* PROCESS READ REQUEST
*
 00103         PREAD EQU *
*                      .          LOAD IN DATA FROM DEVICE
*                      .          VIA CONFIGURED I/O INSTRUCTIONS
*                      .
*
 00103 172164R       STA DAPTR,I  STORE WORD IN DATA BUFFER
 00104 036163R       ISZ CVCTR    IS THIS THE LAST CONVERSION?
 00105 002001        RSS          NO
 00106 026146R       JMP DONE     YES, GO SET UP TO TERMINATE CALL
*
 00107 036164R       ISZ DAPTR    NO, SET UP FOR NEXT CONVERSION
 00110 016053R       JSB READ     INITIATE IT
*
* RESTORE MACHINE TO ORIGINAL STATE ON INTERRUPT
*
```

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

```
PAGE 0006 #01  ** NON-DMS PRIVILEGED DRIVER - PRIVILEGED SECTION **

 00111 103100         CLF 0        TURN OFF INTERRUPT SYSTEM TEMPORARILY
*
 00112 103700  I.3    STC SC,C     ENCODE DEVICE
*
 00113 062171R EXIT   LDA MPFSV    ACCESS PREVIOUS STATE OF MEMORY PROTECT
 00114 002002         SZA          WAS MEMORY PROTECT ON?
 00115 026126R        JMP EXIT1    NO, SO DO NOT TURN ON DCPC INTERRUPTS
*
 00116 065654         LDB INTBA    YES, TURN DCPC COMPLETION INTERRUPTS
 00117 160001         LDA B,I      BACK ON IF THEY WERE ON INITIALLY.
 00120 002020         SSA          ON/OFF STATUS IS INDICATED BY BIT 15
 00121 102706         STC 6        OF EACH DCPC ASSIGNMENT WORD IN THE
 00122 006004         INB          INTERRUPT TABLE
 00123 160001         LDA B,I
 00124 002020         SSA
 00125 102707         STC 7
*
 00126 062170R EXIT1  LDA EOSV     RESTORE E AND O REGISTERS
 00127 103101         CLO
 00130 000036         SLA,ELA
 00131 102101         STF 1
 00132 066167R        LDB BSV      RESTORE B-REGISTER
*
 00133 062171R        LDA MPFSV    RESTORE MEMORY PROTECT FLAG
 00134 071770         STA MPTFL    IN BASE PAGE
 00135 002002         SZA          WAS MEMORY PROTECT ON AT INTERRUPT?
 00136 026143R        JMP EXIT2    NO
*
 00137 062166R        LDA ASV      YES, RESTORE A-REGISTER
 00140 102100         STF 0        TURN ON INTERRUPT SYSTEM
 00141 102705         STC 5        SET MEMORY PROTECT ON
 00142 126055R        JMP PXNN,I   RETURN TO POINT OF INTERRUPTION
*
 00143 062166R EXIT2  LDA ASV      NO,RESTORE A-REGISTER
 00144 102100         STF 0        TURN ON INTERRUPT SYSTEM
 00145 126055R        JMP PXNN,I   RETURN TO POINT OF INTERRUPTION
*
* THIS CODE SETS UP THE TIME OUT TO COMPLETE THE CALL
*
 00146 103100  DONE   CLF 0        TURN OFF THE INTERRUPT SYSTEM
 00147 106700  I.4    CLC SC       TURN OFF PRIVILEGED DEVICE
 00150 003400         CCA          SET TIME OUT FOR
 00151 172174R        STA EQ15,I   ONE TICK AND SET
 00152 162173R        LDA EQ4,I    BIT12 IN EQT4 SO
 00153 032175R        IOR BIT12    RTIOC WILL
 00154 172173R        STA EQ4,I    CALL CXNN ON TIME-OUT
 00155 026113R        JMP EXIT     GO TO EXIT ROUTINE
```

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

```
PAGE 0007 #01   ** NON-DMS PRIVILEGED DRIVER - COMPLETION SECTION **

*
*                      *********************
*                      * COMPLETION SECTION *
*                      *********************
*
00156 000000  CXNN  NOP             COMPLETION SECTION ENTRY POINT
*
00157 002400        CLA             NO, SET A = 0 = NORMAL RETURN
00160 165667        LDB EQT8,I      SET B = TRANSMISSION LOG
00161 126156R       JMP CXNN,I      MAKE COMPLETION RETURN (P+1) TO IOC
```

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

```
PAGE 0008 #01   ** NON-DMS PRIVILEGED DRIVER - DATA AREA **


0254*
0255* CONSTANT AND STORAGE AREA
0256*
0257  00000          A     EQU 0
0258  00001          B     EQU 1
0259  00000          SC    EQU 0         DUMMY I/O SELECT CODE NUMBER
0260*
0261  00162 000000   SCODE BSS 1
0262  00163 000000   CVCTR BSS 1
0263  00164 000000   DAPTR BSS 1
0264  00165 000000   FIRST BSS 1
0265  00166 000000   ASV   BSS 1
0266  00167 000000   BSV   BSS 1
0267  00170 000000   EOSV  BSS 1
0268  00171 000000   MPFSV BSS 1
0269  00172 000000   EQ1   BSS 1
0270  00173 000000   EQ4   BSS 1
0271  00174 000000   EQ15  BSS 1
0272  00175 010000   BIT12 OCT 10000
0273  00176 102500   LIA   LIA 0
0274*
0275* BASE PAGE COMMUNICATIONS AREA DEFINITION
0276*
0277  01650                EQU 1650B
0278  01654          INTBA EQU .+4
0279  01660          EQT1  EQU .+8
0280  01663          EQT4  EQU .+11
0281  01665          EQT6  EQU .+13
0282  01666          EQT7  EQU .+14
0283  01667          EQT8  EQU .+15
0284  01774          EQT15 EQU .+84
0285  01770          MPTFL EQU .+80
0286*
0287* CODE TO SET UP JSB $JPNN,I INSTRUCTION ON BASE PAGE
0288*
0289  00177 000055R $JPNN DEF PXNN      PRIV. SECTION ENTRY POINT ADDR
0290*
0291  00000                ORB          RESET LOCATION COUNTER TO BASE PAGE
0292  00000 116177R $JSB  JSB $JPNN,I   JSB INSTR. TO PRIV SECTION, INDIRECT
0293*
      00200 167777
      00201 000003
      00202 000001
      00203 003700
      00204 000002
      00205 077777
0294                       END
**  NO ERRORS *TOTAL **RTE ASMB 760924**
```

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

READER COMMENT SHEET

RTE Operating System
Driver Writing Manual

92200-93005 **May 1978**

We welcome your evaluation of this manual. Your comments and suggestions help us improve our publications. Please use additional pages if necessary.

Is this manual technically accurate?

Is this manual complete?

Is this manual easy to read and use?

Other comments?

FROM:

Name ______________________________

Company ______________________________

Address ______________________________

____

____

FOLD FOLD

FIRST CLASS
PERMIT NO. 141
CUPERTINO
CALIFORNIA

BUSINESS REPLY MAIL

No Postage Necessary if Mailed in the United States   Postage will be paid by

Hewlett-Packard Company
Data Systems Division
11000 Wolfe Road
Cupertino, California 95014
ATTN: Technical Marketing Dept.

FOLD FOLD

PART NO. 92200-93005
Rev. Code 1805
Printed in U.S.A. 5/78

HEWLETT hp PACKAR
Sales and service from 172 offices in 65 count
11000 Wolfe Road, Cupertino, California 95014