RTE Operating System Drivers and Device Subroutines

Programming and Operating Manual

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

PART NO. 92200-93005

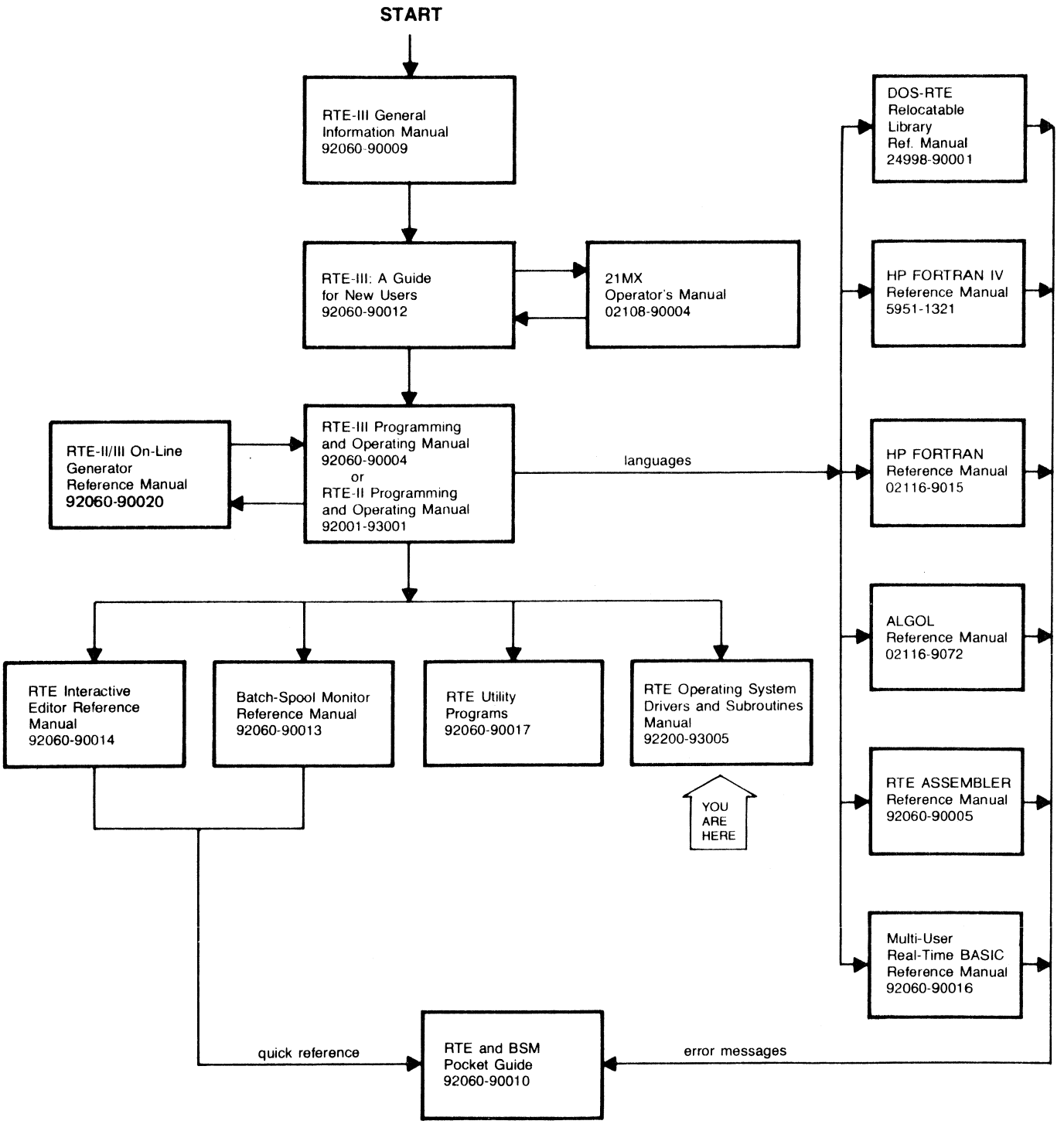
Printed in U.S.A. 7/76

LIST OF EFFECTIVE PAGES

Changed pages are identified by a change number adjacent to the page number. Changed information is indicated by a vertical line in the margin of the page. Original pages (Change 0) do not include a change number. Insert latest changed pages and destroy superseded pages.

Change 0 (Original) ..... JUL 1976

DOCUMENTATION MAP



CONTENTS

ILLUSTRATIONS

TABLES

SECTION I
INTRODUCTION

1-1. PURPOSE

The purpose of the Hewlett-Packard Programming and Operating Manual for Real-Time Executive System Drivers is to enable the user to write relocatable drivers for HP Assembly language and FORTRAN programs. By these programs, full compatibility and integration can be accomplished for use on HP 2100 Series Computers and associated equipment operating in a Real-Time Executive (RTE) environment.

1-2. SCOPE

The manual provides the user with a detailed description of Real-Time Executive System input/output (I/O) concepts, followed by information on all Hewlett-Packard supplied RTE drivers and RTE device subroutines contained in the RTE Operating System Driver and Device Subroutine Package (92062A), as listed in Table 1-1. Included in the manual are full details on writing driver calls, their sequence, the parameters needed, the required system operating environment, and techniques for utilizing the drivers.

1-3. SUPPORTING DOCUMENTATION

For complete information regarding Real-Time Executive Software Systems, the reader should refer to the Software Programming and Operating Manual provided with his system. Documentation available includes:

a. RTE Operating System Programming and Operating Manuals, as listed in Table 1-1.

b. *Real-Time Executive Core-Based Software System Programming and Operating Manual,* HP Part No. 29101-93001.

c. *Real-Time Executive BASIC Software Programming and Operating Manual,* HP Part No. 29102-93001.

d. *Real-Time Executive III Software System Programming and Operating Manual,* HP Part No. 92060-90004.

e. *Real-Time Executive II Software System Programming and Operating Manual,* HP Part No. 92001-93001.

Table 1-1. Real-Time Executive System Drivers (Standard Products)

DRIVER NUMBER	DESCRIPTION	MANUAL PART NUMBER
00	Multiple-Device System Control	29029-95001
05	HP 2640/2644 Terminal	92001-90015
10	HP 7210 Plotter	— — —
10	CalComp Plotter	12560-90023
11	HP 2892A Card Reader	09600-93010
12	HP 2607A Line Printer	92200-93001
12	HP 2767A Line Printer	— — —
13	HP 91200B TV Monitor	91200-90005
15	HP 7261A Mark Sense Card Reader	09601-93014
23	HP 7970 Series 9-Track Magnetic Tape Units	92202-93001
24	HP 7970 Series 7-Track Magnetic Tape Units	25117-93003
30	Fixed Head Disc	— — —
31	HP 7900/7901 Moving Head Disc	— — —
32	HP 7905 Moving Head Disc	— — —
37	HP 59310B, HP-IB	59310-90063

REAL-TIME INPUT/OUTPUT

SECTION II

2-1. INTRODUCTION

In the Real-Time Executive System, centralized control and logical referencing of I/O operations effect simple, device-independent programming. Two terms used in this manual are:

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 channel.

I/O DEVICE — A physical unit defined by an EQT entry (I/O controller) and subchannel.

Each I/O device is interfaced to the computer through an I/O controller associated with one or more I/O channels which are linked by hardware to corresponding memory locations for interrupt processing. By means of several user-defined I/O tables, self-contained multi-device drivers, and program EXEC calls, RTE relieves the programmer of most I/O problems. For further details on the hardware input/output organization, consult the appropriate computer manuals.

2-2. SOFTWARE I/O STRUCTURE

An Equipment Table records each controller's I/O channels, driver, DMA buffering and time-out specifications. A Device Reference Table assigns one or more logical unit numbers to each device and points each device to the appropriate Equipment Table entry, allowing the programmer to reference changeable logical units instead of fixed physical units.

An Interrupt Table directs the system's action when an interrupt occurs on any channel; RTE can call a driver (which is responsible for initiating and continuing operations on all devices' controllers of an equivalent type), schedule a specified program, or handle the interrupt itself.

The programmer requests I/O by means of an EXEC Call in which he specifies the logical unit, control information, buffer location, buffer length, and type of operation. Other subsystems may require additional parameters.

2-3. EQUIPMENT TABLE

The Equipment Table (EQT) has an entry for each I/O controller recognized by RTE. (These entries are established by the user when the RTE System is generated.) The 15-word EQT entries reside in the system, and have the format shown in Table 2-1. The driver returns the device status to EQT Word 5, bits 0 through 7. Table 2-2 is provided as an aid in determining the meaning of the bits. Also returned to the driver is EQT Word 6 which is the current I/O request word as specified in the call. Refer to Figure 2-1 for an explanation of the bits in EQT Word 6.

When RTE initiates or continues an I/O operation, it places the addresses of the EQT entry for the device's controller into the base page communication area (see Table 2-2) before calling the driver routine.

Table 2-1. Equipment Table

WORD	CONTENTS															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	I/O Request List Pointer															
2	Driver "Initiation" Section Address															
3	Driver "Completion" Section Address															
4	D	B	P	S	T	Unit #					Channel #					
5	AV		EQUIP TYPE CODE								STATUS					
6	CONWD (Current I/O Request Word)															
7	Request Buffer Address															
8	Request Buffer Length															
9	Temporary Storage for Optional Parameter*															
10	Temporary Storage for Optional Parameter*															
11	Temporary Storage for Driver															
12*	Temporary Storage for Driver (EQT extension size)															
13	Temporary Storage for Driver (EQT extension starting address)															
14	Device Time-Out Reset Value															
15	Device Time-Out Clock															

*Modified by RTE at each I/O initialization

Where:

D = 1 if DMA 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)

Unit # = Last sub-channel addressed

Channel # = 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 DMA channel

Table 2-1. Equipment Table (Continued)

Where: (Continued)

EQUIP TYPE CODE = Type of devices on this controller. When this octal number is linked with "DVR", it identifies the device's software driver routine as follows

00 to 07	=	paper tape devices (or system control devices)
00	=	teleprinter (or system keyboard control device)
01	=	photoreader
02	=	paper tape punch
05 sub 0	=	console (or system keyboard control device)
05 sub 1 05 sub 2	=	mini cartridge 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
30	=	fixed head disc or drum
31	=	7900 moving head disc
32	=	7905 moving head disc
40 to 77	=	instruments

STATUS = Actual physical status or simulated status at the end of each operation. For paper tape devices, two status conditions are simulated: Bit 5 = 1 means end-of-tape on input, or tape supply low on output.

CONWD = User control word supplied in the I/O EXEC call.

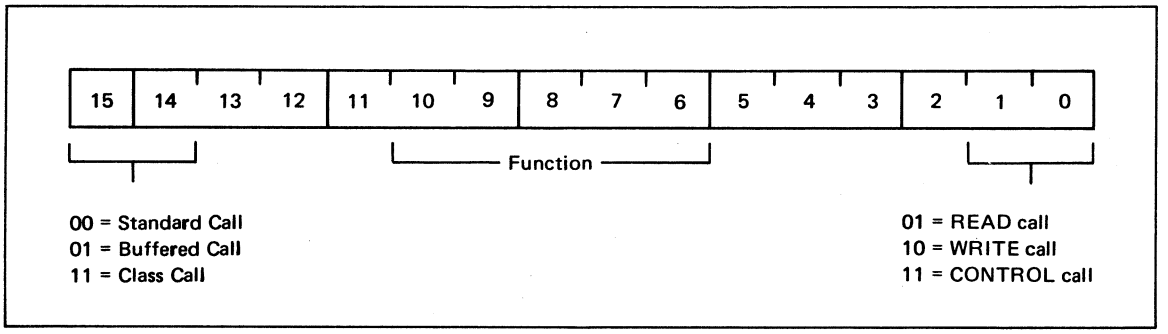


Figure 2-1. Expansion of CONWD Word (EQT Word 6)

Table 2-2. EQT Word 5, STATUS Table

DEVICE \ STATUS	7	6	5	4	3	2	1	0
Teleprinter(s) Photoreader(s) Punch(es) DVR00	X	—	End of I/O Tape	—	—	STL	TEN	—
2640 Terminal 2644 Terminal Cartridge Tape Unit DVR05	BF EOF	— TLP	CD EOT	— RE	— LCA	— CWP	TEN EOD	— CNE/DB
7210 Plotter DVR10	—	—	—	—	—	—	—	PD
2892 Card Reader DVR11	HE	—	—	—	—	—	—	RNR
2767 Line Printer DVR12	—	—	—	—	—	"<" LCF	"*" LCF	NE
2607 Line Printer DVR12	TOF	DM	ON	RY	X	X	Auto page eject	X
7261 Card Reader 2761 Mark Sense Reader DVR15	EOF —	— —	HE/SF HE/SF	PF PF	— —	— —	DE DE	RNR RNR
3030 Mag Tape 7970 DVR22 DVR23	EOF	ST	EOT	TE	I/OR	NW	PE	DB/OL
2766 Fixed Head 2773 Disc/Drum DVR30	DR(1) NR(0)	—	SAC	—	AF	WE	PE	DB
7900 Moving Head Disc DVR31	—	NR	EOT	AE	FC	SC	DE	EE
7905 Moving Head Disc DVR32	PS	FS	HF	FC	SC	NR	—	EE

Where:

PE = Parity Error
HE = Hopper Empty
SF = Stacker Full
RNR = Reader Not Ready
PF = Pick Fail
DE = Data Error
OL = Off Line
ON = On Line
CE = Compare Error
BT = Broken Tape
DB = Device Busy
EOF = End of File
ST = Start of Tape
TE = Timing Error
I/OR = I/O Reject
NW = No Write (write enable ring missing or tape unit is rewinding)
SC = Seek Check
FC = Flagged Track (protected)
AE = Address Error
EOT = End of Track
NR = Not Ready
RY = Ready (0 = power on)
LCF = Last Character Flag
NE = No Error
DR = Disc Ready
SAC = Sector Address Coincidence (troubleshooting only)
PS = Protect Switch Set
FS = Driver Format Switch is Set
HF = Hardware Fault
AF = Abort Flag [NR (Bit = 7=0) has occurred during/since last data transfer]
WE = Currently addressed track is write enabled
EE = Error exists
TEN = Terminal Enabled
TOF = Top of Form
DM = Demand (1 = Idle)
X = Driver Internal Use
STL = Stall required/In program
PD = Pen Down
CD = Control-D Entered
BF = Buffer Flushed
CNI = Cartridge Not Inserted
EOD = End of Data
CWP = Cartridge Write Protected
LCA = Last Command Aborted
RE = Read Error
TLP = Tape at Load Point

2-4. BASE PAGE COMMUNICATION AREA

A block of storage in base page, starting at 1647_8, contains the system communication area and is used by RTE to define request parameters, I/O tables, scheduling lists, operating parameters, memory bounds, etc. The Real-Time Assembler allows absolute references into this area (i.e., less than 2000_8) within relocatable programs, so that user programs can read information from this area, but cannot alter it because of the memory protect feature. The information pertaining to I/O operation contained in the base page communication area is listed in Table 2-3. (For a complete description of the system communication area, refer to the appropriate RTE System Software Operating and Service Manual.)

2-5. LOGICAL UNIT NUMBERS

Logical unit numbers from decimal 1 to 63 provide logical addressing of the physical devices defined by the EQT (I/O controller) and the subchannels (if applicable) and also define the physical devices' (LU) status. These numbers are maintained in a two word Device Reference Table (DRT), which is created at generation, and can be modified by the LU operator request (see figure 2-2).

DRT word one contains the EQT entry number of the device assigned to the logical unit, and the subchannel number within the EQT entry. The second DRT word contains the logical unit's status (up or down) and a pointer to any downed I/O requests. If the pointer is less than 64, it is the LU number off of which the downed I/O requests are queued. If several LU's point to the same device, the requests are queued off the lowest LU number (the major LU). If the pointer is greater than 64, it points to the device's downed I/O request list. There are separate tables for words one and two, with the word two table located in memory immediately following the word one table. The starting address and length of the word one table are recorded in the base page. The functions of logical units 0 through 6 are predefined in the RTE System as:

0 – bit bucket (null device)
1 – system console
2 – reserved for system
3 – reserved for system
4 – standard output unit
5 – standard input unit
6 – standard list unit

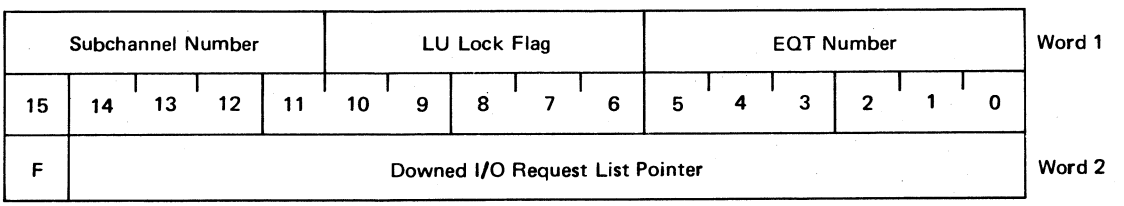


Figure 2-2. Device Reference Table

Logical units 7 through 63 may be assigned for any functions desired, although logical unit 8 is recommended to be the magnetic tape device. The operator can assign EQT numbers and subchannel numbers within the EQT entries to the logical unit numbers when the RTE System is generated, or after the system is running. The user determines the number of logical units when the system is generated.

Table 2-3. Base Page Communications Area - I/O Operations

OCTAL LOCATION	CONTENTS	DESCRIPTION
01650	EQTA	FWA of equipment table
01651	EQT#	No. of EQT entries
01652	DRT	FWA of device reference word 1 table
01653	LUMAX	No. of logical units (in DRT)
01654	INTBA	FWA of interrupt table
01655	INTLG	No. of interrupt table entries
01656	TAT	FWA of track assignment
01657	KEYWD	FWA of keyword block
01660	EQT1	Addresses of first 11 words of current EQT entry
01661	EQT2	
01662	EQT3	
01663	EQT4	
01664	EQT5	
01665	EQT6	
01666	EQT7	
01667	EQT8	
01670	EQT9	
01671	EQT10	
01672	EQT11	
01673	CHAN	Current DMA channel number
01674	TBG	I/O address of time-base card
01675	SYSTY	EQT entry address of system console
. . . .		
01771	EQT12	Addresses of last 4 words of current EQT entry
01772	EQT13	
01773	EQT14	
01774	EQT15	

Logical unit numbers are used by executing programs to specify on which device I/O transfers are to be carried out. In an I/O EXEC Call, the program simply specifies a logical unit number and does not need to know which actual device or which I/O controller handles the transfer.

2-6. INTERRUPT TABLE

The Interrupt Table contains an entry, established at system generation time, for each I/O channel in the computer. If the entry is equal to 0, the channel is undefined in the system. If an interrupt occurs on one of these channels, RTE prints this message:

```
ILL INT xx
```

where xx is the octal I/O channel number. RTE then clears the interrupt flag on the channel and returns to the point of interruption.

If the contents of the entry are positive, the entry contains the address of the EQT entry for the controller on the channel. If the contents are negative, the entry contains the negative of the address for the ID segment of a program to be scheduled whenever an interrupt occurs on the channel.

The interrupt locations in memory contain a JSB $CIC; CIC is the Central Interrupt Control routine which examines the Interrupt Table to decide what action to take. On a power failure interrupt RTE halts unless the power fail routine is used. If privileged interrupt processing is included in the system, the privileged channels bypass $CIC and the interrupt table entirely.

2-7. GENERAL OPERATION OF I/O PROCESSOR

2-8. STANDARD I/O CALLS

A user program makes an EXEC Call to initiate I/O transfers. If the device's controller is not buffered, or in the case of all input transfers, the calling user program is suspended until the transmission is completed. The next lower priority program is allocated execution time during the suspension of a higher priority program.

An I/O request (i.e., Read, Write, Control) is channeled to IOC by the executive request processor. After the necessary legality checks are made, the request is linked into the request list corresponding to the referenced I/O controller. The parameters from the request are set in the temporary storage area of the Equipment Table.

If the device's controller is available (i.e., no prior requests were stacked), the "initiation" section of the associated driver is called. The initiation section initializes the device's controller and starts the data transfer or control function. On return from the initiation section, or if the device's controller is busy, or a required DMA channel is not available, IOC returns to the scheduling module to execute the next lower priority program.

If the device's controller (EQT) or the device (LU) is down, the calling program is automatically suspended in the general wait list (status = 3). While in this list the program is swappable, and if any LU or EQT is set up the program is automatically rescheduled.

Interrupts from the device's controller cause the Central Interrupt Control (CIC) module to call the "completion" section of the driver. At the end of the operation, the driver returns to CIC and consequently to IOC. IOC causes the requesting program to be placed back into the schedule list and checks for an I/O stacked request. If there are no stacked requests, IOC exits to the dispatching module (DISP); otherwise, the initiation section is called to begin the next operation before returning.

2-9. CLASS I/O CALLS

Class I/O consists of unique scheme of programming within the RTE-system to effectively handle several programs addressing either other programs or I/O devices. The following description of Class I/O relies upon a Glossary of Terms directly related to Class I/O (see Table 2-4).

Table 2-4. Glossary of Terms for Class Input/Output

TERM	DESCRIPTION
Class	An account which is owned by a program which may be used by a group of programs.
Class Number	The account number referred to in number one.
Class Users	Programs that use the class number.
Class Request	An access to a logical unit number with a class number.
Class Members	Logical unit numbers that are currently being accessed in behalf of a class. Completion of access removes the association between class number and logical unit number (completion of access is defined as when the driver completes the request).
Class Queue (Pending)	The set of uncompleted class requests.
Class Queue (Completed)	The set of all completed class requests where the structure is defined as first in, first out.

Class numbers are established during system generation after the last system module is loaded. The generator requests how many class numbers are to be established and the operator responds with a number between 0 and 255. Once the numbers are established the system keeps track of them and assigns them (if available) to the calling program when a Class I/O call is made and the Class Number parameter is set to zero. Once the number has been allocated, the user can keep it as long as desired and use it to make multiple Class I/O Calls. When the user is finished with the number it can be returned to the system for use by some other Class user. One example of using Class I/O is program-to-program communication. The example program in Figure 2-3 and described in the following sequence of events shows how this is accomplished.

a. User Program 1 issues a Class I/O call with the Class Number parameter set to zero and the logical unit number portion of the control word parameter set to zero. This causes the system to allocate a Class Number (if available) and the request to complete immediately. (Logical unit zero specifies a system "bit bucket" which implies immediate completion).

b. User Program 1 now sets bit 14 of the Class Number parameter which specifies that the Class Number is to be saved, and issues a Class GET call to that Class Number.

c. The GET call completes (data buffer is returned) and the Class Number is saved in common.

d. A second GET call is issued to the same Class Number. This time user Program 1 suspends because there is no outstanding I/O call against that Class Number.

e. User Program 2 obtains Program 1's Class Number from common and issues a Class I/O call to that number (specifying logical unit zero for immediate completion). This causes Program 1 to be rescheduled and it accepts the data passed to it by Program 2.

f. If the same process is repeated for Program 2 and part of the data it passes Program 1 is Program 2's Class Number, then complete program-to-program communication is possible with both programs passing each other's data.

The system handles a Class I/O call in the following manner. When the class user issues a Class I/O call and the call is received, the system allocates a buffer from available memory and puts the call in the header (first 8 words) of this buffer. The call is placed in the pending class queue and the system returns control to the class user. If this is the only call in the pending class queue, the driver is called immediately, otherwise the system returns control to the class user and calls the driver according to program priority. If buffer space is not available, the class user is memory suspended unless bit 15 ("no wait") is set. If the "no wait" bit is set, control is returned to the class user with the A-Register containing a −2 indicating no memory available. If the class number is not available or the I/O device is down, the class user is placed in the general wait list (status = 3). If the call is successful, the A-Register will contain the class number on return to the program.

The buffer area furnished by the system is filled with the caller's data if the request is either a WRITE or a WRITE/READ call. The buffer is then queued (pending) on the specified logical unit number. Since the system forms a direct relationship between logical unit numbers and EQT entries, the buffer can also be thought of as being queued on the EQT entry.

After the driver receives the Class I/O call (in the form of a standard I/O call) and completes, the system will:

a. Release the buffer portion of the request if a WRITE. The header is retained for the GET call.

b. Queue the header portion of the buffer in the Completed Class Queue.

c. If a GET call is pending on the Class Number, reschedule the calling program. (This means that if the user issues a Class GET call or examines the Completed Class Queue before the driver completes, the user has effectively beat the system to the Completed Class Queue.) Note that the program that issued the Class I/O call and the program that issued the Class GET call do not have to be the same program.

d. If there is no GET call outstanding, the system continues and the driver is free for other calls.

```
FTN,L
      PROGRAM PROGA
      DIMENSION IBFR(32),INAME(3)
            .
            .
            .
C
C     DO CLASS WRITE/READ TO LU=0.
C
      ICLAS=0
      CALL EXEC(20,0,IBFR,-64,IDUMY,JDUMY,ICLAS)
C
C     SCHEDULE RECEIVING PROGRAM AND PASS IT CLASS#.
C
      INAME(1)=501228
      INAME(2)=47507B
      INAME(3)=41000B
      CALL EXEC(10,INAME,ICLAS)
            .
            .
            .
      END



FTN,L
      PROGRAM PROGB
      DIMENSION IBFR(32),IPRAM(5)
C
C     SAVE CLASS #, IPRAM(1)
C
      CALL RMPAR(IPRAM)
C
C     ACCEPT DATA FROM PROGA USING CLASS GET CALL
C     AND RELEASE THE CLASS NUMBER.
C
      CALL EXEC(21,IPRAM(1),IBFR,32)
            .
            .
            .
```

Figure 2-3. Example of Class I/O Mailbox Communication

Only after the driver completes can the user effectively issue a Class GET call to check the Completed Class Queue for a completion. When the user issues the GET call, the Completed Class Queue is checked and one of the following paths is taken.

a. If the driver has completed, the header of the buffer is returned (plus the data). The user (calling program) has the option of leaving the I/O request in the Completed Class Queue so as not to lose the data. In this case a subsequent GET call will obtain the same data. Or the user can dequeue the request and release the Class number.

b. If the driver has not yet completed (GET call beat system to the Completed Class Queue), the calling program is suspended in the general wait list (status = 3) and a marker so stating is entered in the Completed Class Queue header. If desired, the program can set the "no wait" bit to avoid suspension. In any case, when the driver completes, programs waiting in the general wait list for this class are automatically rescheduled.

2-10. DRIVER STRUCTURE AND OPERATION

An I/O driver, operating under control of the Input/Output Control (RTIOC) and Central Interrupt Control (CIC) modules of RTE, is responsible for all data transfer between an I/O device and the computer. The device EQT entry contains the parameters of the transfer and the base page communication area contains the number of the allocated DCPC channel, if required. It should be noted that RTE operation makes it mandatory that a synchronous device driver must use a DCPC or privileged interrupt channel for data transfer.

An I/O driver always has an initiation section and usually a completion section. If nn is the octal equipment type code of the device, Ixnn and Cxnn are the entry point names of the two sections respectively, the DVynn is the driver name.

As shown, the driver name is five characters long, starting with the characters "DV" and ending with a two-digit octal number (e.g., DVR00). This name is usually obtained from the software distribution package. The entry point names are four characters in length and start with either "I" or "C" and usually end with the same two-digit octal number used in the driver name.

However, since the system generator does not examine the driver's NAM record, the driver may in fact be renamed to support more than one device. The rules for the choice of "x" and "y" above are as follows:

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

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

Using the above rules, a driver named DVR16 has entry points named I.16 and C.16. A driver named DVP16 has entry points IP16 and CP16. This allows one driver to support more than one device type.

Privileged drivers are in a special class. Refer to the end of this section for a discussion of privileged drivers.

2-11. INITIATION SECTION

The RTIOC module of RTE calls the initiation section directly when an I/O transfer is initiated. Locations EQT1 through EQT15 of the base page communication area contain the addresses of the appropriate EQT entry. CHAN in base page contains the number of the DMA channel assigned to the device's controller, if needed. This section is entered by a jump subroutine to the entry point, I.nn. The A-Register contains the select code (channel number) of the channel (bits 0 through 5 of EQT entry word 4). The driver returns to IOC by an indirect jump through I.nn.

Before transferring to I.nn RTE places the request parameters from the user program's EXEC Call into words 6 through 10 of the EQT entry. The subchannel number is placed into bits 6 through 10 of word 4. Word 6, CONWD, is modified to contain the request code in bits 0 and 1 in place of the logical unit.

Once initiated, the driver can use words 6 through 13 of the EQT entry in any way, but words 1 through 4 must not be altered. The driver updates the status field in word 5, if appropriate, but the rest of word 5 must not be altered.

2-12. FUNCTIONS OF THE INITIATION SECTION. The initiation section of the driver operates with the interrupt system disabled (or as if it were disabled, in the case of privileged interrupt processing; see discussion of special conditions under "Privileged Interrupt Processing").

The initiation section of the driver is responsible for these functions (as flow-charted in Figure 2-4).

a. Checks for power fail entry by examining bit 15 (= 1) of EQT word 5. This bit is set only on powerfail entry (see "b" in Power Fail).

b. Rejects the request and proceeds to "g" if:

   1. The device or controller is inoperable,
   2. The request code, or other of the parameters, is illegal.

c. Configures all I/O instructions in the driver to include the select code (and DMA channel) of the device's controller.

d. Initializes DMA, if appropriate.

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

f. Optionally sets the device's controller time-out clock (EQT 15).

g. Returns to RTIOC with the A-Register set to indicate initiation or rejection and the cause of the reject:

   If A = 0, then operation was initiated.

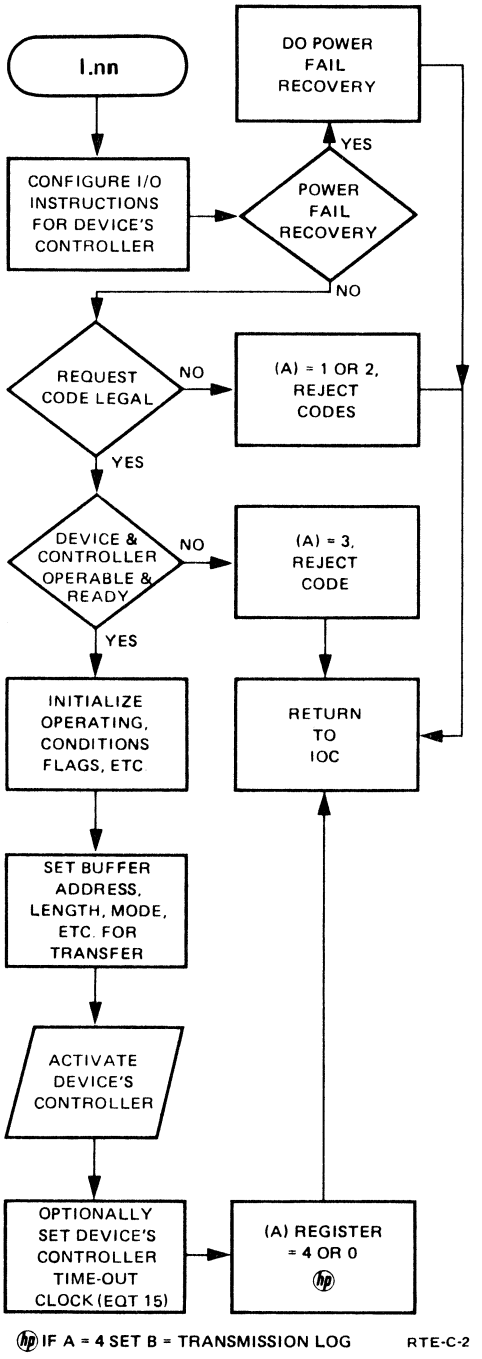


Figure 2-4. I/O Driver Initiation Section

If A = 1,2,3, then operation rejected because:

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

If A = 4, immediate completion. (Transmission log should be returned in the B-Register in this case.)

If A = 5, DMA channel required.

2-13. DMA INITIALIZATION. A driver can obtain a DMA channel in two ways:

a. The channel can be assigned during generation by entering a "D" in the driver's Equipment Table Entry.

b. The driver can dynamically assign a DMA channel as required.

If a driver requires DMA but does not require or use the DMA interrupt, the DMA control should be cleared after DMA initialization. Further special processing is required in this case.

If a driver requires DMA, and the DMA interrupt, special processing must be included in the driver. After disabling the interrupt system, initiating DMA and clearing control, the driver sets a software flag to indicate that a DMA channel is active.

The software flag is either the first or second word of the interrupt table, depending on which DCPC channel is used. The flag is set by making bit 15 equal to 1.

INTBL (1) – channel 1 (location 6)
INTBL (2) – channel 2 (location 7)

The address of INTBL is contained in the word INTBA in the base page communication area. When bit 15 is set, the rest of the word must not be altered. The operation can be performed only if DUMMY is non-zero (meaning the system includes privileged interrupt processing).

The following code demonstrates these principles:

```
CLF 0                 Disable interrupts.

STC DMA, C            Initiate DCPC channel

CLA                   ┐
CPA DUMMY             │ Bypass this section if
JMP X                 ┘ DUMMY = 0 and special
                        processing is not needed.

CLC DMA               ┐
LDB INTBA             │
LDA CHAN              │ Clear DMA control. Set B =
CPA = D7              │ address of the appropriate
INB                   ┘ entry in the interrupt table.
```

```
LDA B,I
IOR = B100000
STA B,I
STF 0
```

Set bit 15 of the entry equal to 1 and return to the interrupt table. Enable interrupt system.

```
X EQU *
```

Continue processing.

There may be times when a driver will only occasionally need DMA, and thus not want to always tie up a DCPC channel while it is operating. This may be done in either of two ways: (Note that in example No. 1, the DCPC channel is always assigned before the driver is entered. In example No. 2, the DCPC channel is assigned only if the driver requests it.)

Example 1—The DMA flag is set at generation time by entering a "D" in the driver's equipment table entry. The driver may return the DCPC channel (before completion if desired) by clearing the appropriate INTBL word (first or second word of interrupt table). This may be done as follows:

```
LDA   DMACH     Get current channel
LDB   INTBA     and INTBL address
SLA             If Channel 7
INB             step address
CLA             Clear the
STA   B,I       channel word
```

Example 2—The DMA flag is not set at generation time as above. In this case the driver is entered by RTIOC without a channel being assigned. The driver must analyze the request and determine if a channel is required, and if so, request a channel from RTIOC by returning via I.XX,I with A=5. RTIOC will assign a channel and recall the driver. The recall completely resets EQT words 6 through 10. Since it is possible for the calling program to be aborted between the request for DMA and the resulting recall of the driver, the driver must determine, independently of its past history, if it has DMA. The following code illustrates these principles:

```
.
.
.
DLD   INTBA,I   Come here if DMA required
CPA   EQT1      Is channel 6 assigned?
JMP   CH6       Yes; go configure
CPB   EQT1      Is channel 7 assigned?
JMP   CH7       Yes; go configure
LDA   =B5       No channel so
JMP   I.XX,I    Request one from RTIOC
.
.
.
```

In this case the driver must also tell RTIOC that it has a DCPC channel at completion of request. This is done by setting the sign bit in the A-Register on the completion return to RTIOC. This bit may be set at all times — even when the driver does not own a DCPC channel. However, if

set when not required, some extra overhead in RTIOC is incurred. The sign bit is set in addition to the normal completion code. The following code illustrates this principle:

```
.
.
.
LDA   COMCD      Get completion code
IOR   =B100000   Set the sign bit
JMP   C.XX,I     Return to RTIOC
```

NOTE

If your driver wishes to do a series of non-DMA operations, but still retain the DCPC channel assignment, you must clear bit 15 in the first or second word of the INTBL entry to prevent the system from restoring DMA. The correct word must be determined by the driver and is the word described in the above paragraphs. That is:

INTBL (1) – channel 1 (location 6)
INTBL (2) – channel 2 (location 7)

Programming Hint—A driver may use the following code to determine which DCPC channel it is using at any time:

```
DLD INTBA,I      Get DMA words
RAL,CLE,ERA      Clear sign
RBL,CLE,ERB      bits (needed only if driver set the sign bit)
CPA EQT1         Channel 6?
JMP CH6          Yes
CPB EQT1         Channel 7?
JMP CH7          Yes
JMP NODMA        No-no DMA assigned
```

2-14. COMPLETION SECTION

RTE calls the completion section of the driver whenever an interrupt is recognized on an I/O controller associated with the driver. Before calling the driver, CIC sets the EQT entry addresses in base page, sets the interrupt source code (select code) in the A-Register, and clears the I/O interface or DMA flag. The interrupt system is disabled (or appears to be disabled if privileged interrupt processing is present). The calling sequence for the completion section is:

Location	Action
	Set A-Register equal to interrupt source code
(P)	JSB C.nn
(P+1)	Completion return from C.nn
(P+2)	Continuation or error retry return from C.nn

The return points from C.nn to CIC indicate whether the transfer is continuing or has been completed (in which case, end-of-operation status is returned also).

The completion section of the driver is flowcharted in Figure 2-5 and performs the following functions in the order indicated.

a. Checks whether word 1 (controller I/O request list pointer) of the EQT entry equals zero. If it does, a spurious interrupt has occurred (i.e., no I/O operation was in process on the controller). The driver ignores the interrupt, sets EQT 15 (time-out clock) to zero to prevent time-out, and makes a continuation return. If not zero, the driver configures all I/O instructions in the completion section to reference the interrupting controller, and then proceeds to "b".

b. If both DMA and the device controller completion interrupts are expected and the device controller interrupt is significant, the DMA interrupt is ignored by returning to CIC in a continuation return.

c. Performs the input or output of the next data item if the device is driven under program control. If the transfer is not completed, the driver proceeds to "f".

d. If the driver detects a transmission error, it can re-initiate the transfer and attempt a retransmission. A counter for the number of retry attempts can be kept in the EQT. The return to CIC must be (P+2) as in "f".

e. At the end of a successful transfer or after completing the retry procedure, the following information must be set before returning to CIC at (P+1):

   1. Set the actual or simulated device controller status into bits 0 through 7 of EQT word 5.

   2. Set the number of words or characters (depending on which the user requested) transmitted into the B-Register.

   3. Set the A-Register to indicate successful or unsuccessful completion and the reason:

      A equals 0 for successful operation,
      A does not equal 0 for unsuccessful:

      1 – device or controller malfunction or not ready,
      2 – end-of-tape (information),
      3 – transmission parity error,
      4 – device time-out.

f. Clears the device controller and DMA control, if end-of-operation, or sets the device controller and DMA for the next transfer or retry. If not end-of-operation (i.e., a continuation exit is to be made), the driver can again optionally set the device controller time-out clock. Returns to CIC at:

   (P+1) – completion with the A- and B-Registers set as in "e".
   (P+2) – continuation; the registers are not significant.

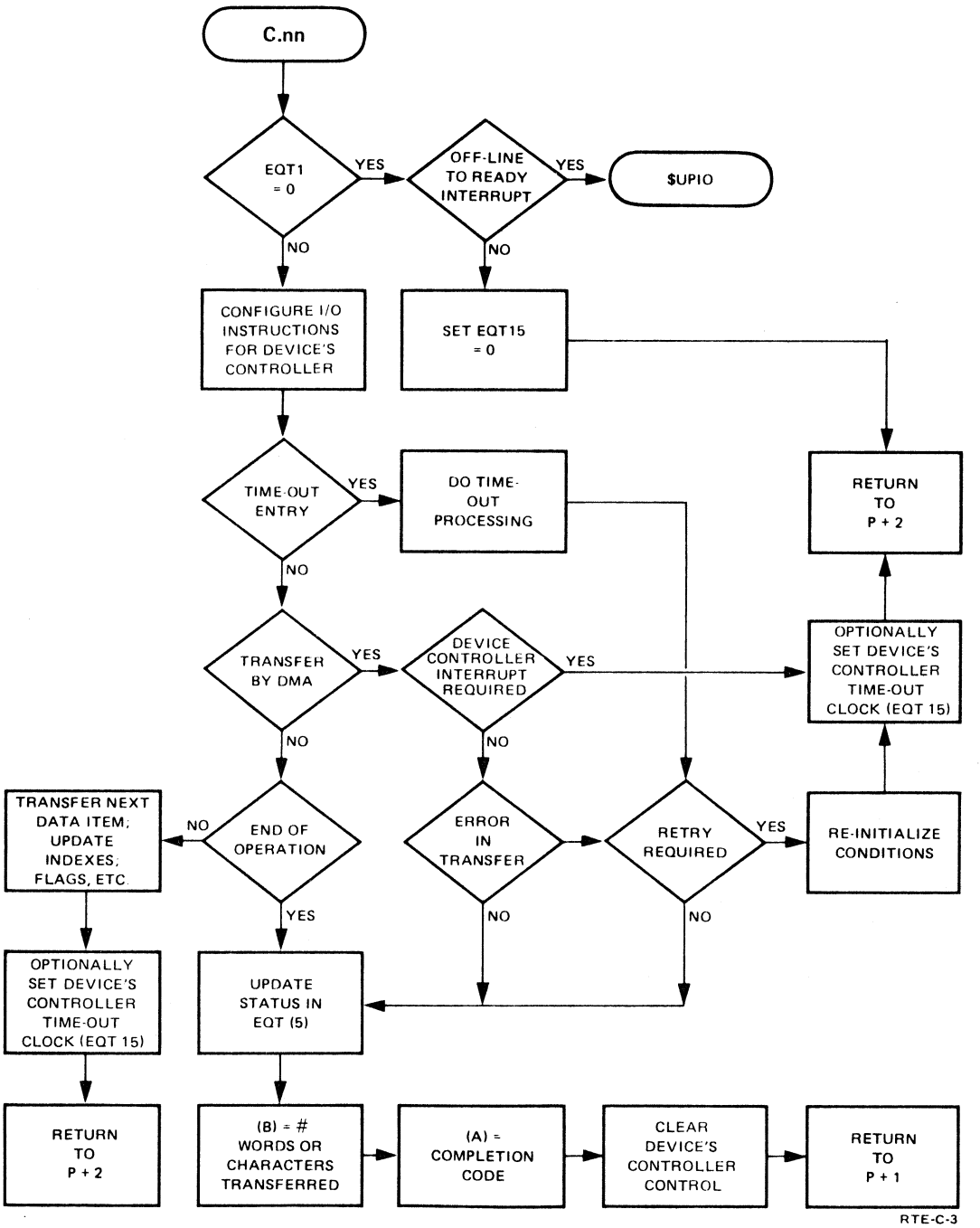


Figure 2-5. I/O Driver Completion Section

2-15. I/O CONTROLLER TIME-OUT

Each I/O controller can have a time-out clock that will prevent indefinite I/O suspension. Indefinite I/O suspension can occur when a program initiates I/O, and the device's controller fails to return a flag (possible hardware malfunction or improper program encoding). Without the controller time-out, the program which made the I/O call would remain in I/O suspension indefinitely awaiting the operation done indication from the device's controller. With respect to privileged drivers, the time-out parameter must be long enough to cover the period from I/O initiation to transfer completion.

Two words, EQT 14 and EQT 15, of the EQT entry for each I/O controller function as a controller time-out clock. EQT 15 is the actual working clock, and before each I/O transfer is initiated, is set to a value m, where m is a negative number of 10 ms time intervals. If the controller does not interrupt within the required time interval, it is to be considered as having "timed out". The EQT 15 clock word for each controller can be individually set by two methods.

a. The system inserts the contents of EQT 14 into EQT 15 before a driver (initiation or completion section) is entered. EQT 14 can be preset to m by entering (T=) m 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 can set the value m into EQT 15 just before it exits. This value m can be coded permanently into the driver or else passed to the driver as an I/O call parameter.

NOTE

The system always inserts the contents of EQT 14 into EQT 15 before entering a driver except on initialization if EQT 15 is not zero, it is not reset. However, a time-out value inserted directly into EQT 15 by the driver overrides any value previously set by the system (from EQT 14).

2-16. DRIVER PROCESSING OF TIME-OUT

A driver indicates to the system that it wants to process time-out by setting bit 12 in EQT word 4. The system never clears this bit so it need be set only once. In this case when a controller times out, the following action takes place.

a. Bit 11 in EQT word 4 is set.

b. The driver is entered at C.nn with the A-Register set to the select code (from EQT word 4).

c. The driver must recognize that the entry is for time-out by examining bit 11 of EQT word 4 and do whatever is necessary. The driver should then clear bit 11 in the event it is entered again prior to completion of the operation so that it knows why it is being entered on the next call. (RTIOC will clear this bit prior to entering the driver at I.nn.)

d. The driver may continue or complete the operation. If it completes the operation it may set the A-Register to 4 to indicate time-out.

e. If the A-Register is set to 4, RTIOC will issue the message

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

where

x is the EQT number, *y* is the LU number, and *z* is the subchannel. The LU is set down.

2-17. SYSTEM PROCESSING OF TIME-OUT

In the case where the driver does not set bit 12 of EQT word 4, the following actions take place on time-out.

a. The calling program is rescheduled, and a zero transmission log is returned to it.

b. The LU is set to the down status, and bit 15 in the second word of the device's LU entry is set to one. An error message is printed; e.g.,

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

c. The system issues a CLC to the controller's select code(s) through the EQT number located in the interrupt table.

Due to the system issuing a CLC to the device's select code, each controller interface card requires an entry in the interrupt table during generation. If an I/O card did not normally interrupt, and therefore did not have an entry in the interrupt table, and the controller had timed out, the system would not be able to issue a CLC to the I/O card.

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

2-18. DEVICE CLEAR

If an I/O suspended program is aborted while waiting for a controller, the system clears the controller by sending a clear control request (00) to the driver. If the controller can be cleared without interrupt (i.e., immediately), the driver should return with A-Register = 2 or 4. If an interrupt is required, the driver should return with A-Register = 0. In this latter case the system will force a 1-second time-out for the controller. If the interrupt is not serviced before this time-out, the system will process it as in step "c" above. Note that the driver is not allowed to process time time-out.

2-19. DRIVER AUTO UP

A driver has the capability of automatically "uping" itself through a JMP instruction. For example, if a driver makes a not ready, parity error, EOT, or time-out return to the system, and subsequently detects an interrupt (or time-out) entry which signals that the controller is now ready, it may "up" itself as follows:

```
JMP $UPIO
```

The device's EQT and any of the EQT's downed LU's will be upped. If any request is pending the system will call the driver at I.xx.

2-20. MAPPING SUBROUTINES FOR DRIVERS (RTE-III ONLY)

There are two routines supplied with the RTE-III System that may be called by a driver to load the User Map to describe a certain program. $PVMP may be called by privileged drivers and $XDMP may be called by non-privileged drivers.

2-21. $PVMP SUBROUTINE (PRIVILEGED)

The $PVMP subroutine may be called by a privileged driver to load the User Map to describe a certain program. This routine is necessary only if the privileged driver needs to access a buffer in a user program. This routine is a type 8 module which means it is not re-entrant. A Type 8 module is loaded with each driver that calls it even though only one copy of the relocatable is input to RTGEN. Note that this routine should be used only by privileged drivers.

The calling sequence is as follows:

```
        EXT     $PVMP
        .                       Normal privileged entry Code (save regis-
        .                       ters, etc)
        .
        SSM     DMSST           Save DMS status at interrupt
        LDA     MAPAD           Address of map storage area
        IOR     SIGN            Set sign bit indicating store in memory
        USA                     Store user map in memory
        LDA     IDADR           Get ID segment address of program
        JSB     $PVMP           Go set user map
        SZA,RSS                 Check for error return
        JMP     ERROR
        UJP     CONT            Enable user map and continue
CONT    .                       Processing for privileged interrupt
        .
        .
EXIT    LDA     MAPAD           Address of map storage area
        USA                     Restore original map
        JRS     DMSST RTN       Restore DMS status at interrupt and
        .                       continue
        .
RTN     .                       Restore registers and memory protect status
        .                       then return to point of interrupt
        MAPAD
MAPAD   DEF     MAP
MAP     BSS     32              Map save area
SIGN    OCT     100000
IDADR   BSS     1               Storage for ID segment address
DMSST   BSS     1               Storage for DMS status at interrupt
```

$PVMP will check to see if the program is resident in memory. If it is not, the User Map will not be reloaded and the A-Register will be zero on return. If the program is resident, the User

Map will be loaded and the A-Register will be non-zero on return. Note that any privileged driver using this routine must save the contents of the User Map before calling this routine and must restore the contents before returning to the point of interrupt.

2-22. $XDMP SUBROUTINE (NON-PRIVILEGED)

The $XDMP subroutine may be called by a non-privileged driver to load the User Map to describe a certain program. Since the system will enter the driver with the map enabled which describes the buffer of the current I/O call, this routine is necessary only for those drivers which must access another buffer which is contained in a user program. Note that this routine is intended for the use of non-privileged drivers only and is included as part of the system. The calling sequence is as follows:

```
        EXT     $XDMP
        .
        .
        .                       Normal driver processing
        RSA                     Get DMS status
        RAL,RAL                 Position current status in upper bits
        STA     DMSST           Save status for later
        LDA     MAPAD           Address of map storage area
        IOR     SIGN            Set sign bit indicating store in memory
        USA                     Store user map in memory
        LDA     IDADR           Get ID segment address of program
        JSB     $PVMP           Go set user map
        SZA,RSS                 Check for error return
        JMP     ERROR
        UJP     CONT            Enable user map and continue
CONT
        .
        .
        .                       Process buffer under User Map
EXIT    LDA     MAPAD           Address of map storage area
        USA                     Restore original map
        JRS     DMSST           NXT Restore earlier DMS status and
                                continue
NXT     .                       Proceed with normal processing
        .
        .
        .
MAPAD   DEF     MAP
MAP     BSS     32              Map storage area
SIGN    OCT     100000
IDADR   BSS     1               Storage for ID segment address
DMSST   BSS     1               Storage for DMS status
```

The routine will check to see it the program is resident in memory. If it is not, the User Map will not be reloaded, and the A-Register will be zero on return. If the program is resident, the User Map will be reloaded and the A-Register will be non-zero on return.

Note that any driver using this routine must save the contents of the User Map before calling this routine and must restore the contents before exiting.

2-23. POWER FAIL

The system power fail routine, if loaded at generation, will perform the following steps:

a. When power comes on, it will restart the real-time clock, set up a time-out entry (TO) back to its EQT, and then return to the power fail interrupt location.

b. When the EQT entry times out, the system will check EQT word 5 bit 14 and 15 of each I/O controller. The status of bits 14 and 15 will indicate whether the I/O controller is "down" or "busy". The system will also check bit 13 of EQT word 4 (set by driver) which indicates if the driver is to process the power fail.

c. If the I/O controller was busy when the power failed and the power fail bit is set when power resumes, the driver is entered at I.nn and the EQT is not reset. If the power fail bit is not set, the controller is set "down". The system then sets the controller "up", resets the EQT, and enters the driver at I.nn.

In other words, if the controller was reading or writing data when the power failure occurred and the driver is designed to handle power fail, when power resumes the controller driver has the responsibility to recover from the power fail in the best possible manner. If the controller was busy when power failure occurred and the controller driver is not written to handle power failure, the routine attempts to restart the I/O operation.

If the controller or device was down when the power failed and the power fail bit is set or not set, when power resumes the system "ups" the device, resets the EQT and enters the driver at I.nn. In other words, if the controller or device was down when power failed, when power resumes the system "ups" the controller and device and attempts to start the operation, if any, in the I/O request list. An HP-supplied program called AUTOR will be scheduled. AUTOR will send the time of power failure to all teletypes on the system (which re-enables all terminals). AUTOR is written in FORTRAN, with the source tape supplied so the user can easily modify the program to suit his individual needs.

2-24. SAMPLE I/O DRIVER

The sample driver shown in Figure 2-6 demonstrates the principles involved in writing an I/O driver for the RTE System. Note that this driver is for tutorial purposes only and not one of the drivers supplied with the system.

2-25. PRIVILEGED INTERRUPT PROCESSING

When a special I/O interface card, HP 12610, is included in the system, RTE allows a class of privileged interrupts to be processed independently of regular RTE operation, with a minimal delay in responding to interrupts. The presence and location of the special I/O card is selected at system generation time. Its actual hardware location is stored in the word DUMMY in base page (or if not available, zero).

The special I/O card physically separates the privileged interrupts from the regular system-controlled interrupts. When an interrupt occurs the card has its flag set which enables the card

to hold off non-privileged, lower priority interrupts. This means that the system does not operate with the interrupt system disabled, but in a hold-off state. Furthermore, the privileged interrupts are always enabled when RTE is running, and can interrupt any process taking place.

2-26. PRIVILEGED INTERRUPTS

The privileged interrupts are processed in two ways:

a. Through a privileged driver which has in general the structure of a standard I/O driver plus a special privileged interrupt processor routine.

b. Through a special routine embedded in the system area.

Note that the routines which handle privileged interrupts must be completely independent of RTE (i.e., they cannot use any system functions).

If the first method is used, the calling program can make a standard I/O call to the privileged device. The calling program will be suspended for the time it takes to do the transfer, after which it will be rescheduled. To the calling program, there is no difference between a privileged type device I/O call and a non-privileged (standard) type device I/O call. If the privileged driver is assigned a time-out parameter, the parameter must be long enough to cover the period from I/O initiation to transfer completion.

If the second method is used, a "JSB--,I" in the interrupt location (set by using "ENT,name" when configuring the interrupt table) channels the special interrupt directly to the entry point of its associated special routine. CIC and the rest of RTE are not aware of these interrupts. CIC sets a software flag in base page (MPTFL) indicating the status of the memory protect fence.

If MPTFL equals zero, the memory protect was "ON" at the time of the interrupt. Any special interrupt routine must restore the status of memory protect and (for RTE-III) the dynamic mapping system immediately before returning to the point of interruption by a "JMP---,I".

If MPTFL equals one, RTE itself was executing when the privileged interrupt occurred, and memory protect was "OFF". The special routine must not restore memory protect in this case.

2-27. MEMORY ACCESS BY PRIVILEGED INTERRUPT (RTE-III ONLY)

A privileged interrupt routine, whether embedded directly within the system or within a privileged driver, must save and restore all registers which are used (including index registers), restore memory protect to its original state (word MPTFL contains this status), and restore the status of the Dynamic Mapping System. If the privileged driver wants to access a buffer within a disc resident program, the User Map will have to be saved and restored. In addition, it will have to be loaded to describe the correct partition. However, if the driver limits its access to a buffer in either common or the subsystem global area, the driver may access that buffer while in the System Map if the generation option to include common and SSGA in the System Map was exercised. That is, if the program the driver is servicing puts a buffer in the common area, then the driver can access that data through the System Map which is enabled by the interrupt. Note that no standard HP software uses common; it is reserved for the user.

2-28. SPECIAL PROCESSING BY CIC

The Central Interrupt Control (CIC) module is entered on all normal (non-privileged) interrupts. CIC disables the entire interrupt system (including privileged), saves registers, issues a clear flag instruction (CLF) to the interrupt location, sets the memory protect "OFF" (MPTFL= 1), and checks the DUMMY word. If the DUMMY word is zero, the hardware interrupt system is left disabled and normal processing continues. If non-zero, a set control instruction (STC) is issued to the I/O location specified (this assumes that the flag on the special I/O card is set). The STC holds off lower priority interrupts until the control flip-flop is cleared on the special card.

If DUMMY is non-zero, CIC also clears the control flip-flop on each DCPC channel to defer DMA completion interrupts, and enables the interrupt system (a NOP in the interrupt location for the special card causes its interrupts to be ignored). The DMA transfer itself is not affected, only the completion interrupt.

Interrupt processing continues and control is passed to a driver, timer routine, scheduler, or other appropriate executive module. Privileged interrupts can occur during this processing. RTE returns to user program processing through the interrupt return module, $IRT.

$IRT briefly disables the entire interrupt system. Each active DCPC channel is reset (STC) to allow it to interrupt. User register values are restored, and the memory protect system is inactivated. The User Map (for RTE-III) is enabled and control is transferred to the user program with the interrupt system on.

2-29. PRIVILEGED INTERRUPT ROUTINES

A privileged interrupt routine, whether embedded directly within the system or within a privileged driver, must save and restore all registers, and restore memory protect to its original state (word MPTFL contains this status) and (for RTE-III) restore the status of the Dynamic Mapping System. This is because any interrupt automatically disables memory protect. The privileged interrupt routine must not use any features or requests of RTE, or either DCPC channel. It can communicate with normal user programs by use of the appropriate COMMON region. Flags, parameters, control words, etc., can be set and monitored by either routine in the pre-defined locations in COMMON. The starting address of the COMMON region is available in base page. A normal user program can be scheduled to run at periodic time intervals to scan and set indicators in COMMON.

2-30. SAMPLE PRIVILEGED DRIVER

The following discussion describes Figure 2-7, an example of a privileged driver, generalized to DVRXX, which is controlling a device operating in the privileged mode. Figure 2-8 shows a similar driver written specifically for RTE-III.

The device transfers one word of data each time it interrupts, and the data is stored into the buffer passed to the driver via the call parameters. Also passed to the driver is the number of data words to be input from the privileged device, this being the length of the data buffer.

The concepts behind such a driver are as follows:

a. It is called by standard EXEC I/O call.

b. The calling program is placed into I/O suspension.

c. This device controller's trap cell is changed from "JSB CIC" to "JSB P.XX" where P.XX is the entry point to the privileged routine within the driver.

d. Each time the device's controller interrupts, the RTIOC overhead is circumvented because the privileged routine P.XX is entered directly.

e. After each interrupt, if another data point is still required to satisfy the buffer length, the device's controller is again encoded to subsequently interrupt, and the privileged routine is exited.

f. When the entire data buffer has been filled, the driver needs a way to communicate to the Executive that the transfer is complete. This is accomplished by allowing the driver to time-out. The time-out causes RTIOC to re-enter the driver at C.XX.

g. C.XX returns the transmission log, via the B-Register, and a successful completion indication, via the A-Register, to RTIOC.

h. RTIOC then reschedules the program which called the driver through its normal I/O completion machinery.

A standard RTE driver uses the EQT for all its temporary storage so that the same driver can be driving more than one device controller simultaneously. A privileged driver, however, cannot do this because it can never know the state of pointers to the EQT while it is running since it is running independently of the Executive. The privileged driver keeps its temporary storage internally, and therefore can control only one device controller. For each device controller the driver will control, the driver must be reassembled with all names DVRXX and $JPXX (for this example) changed to another number. Then one driver per device controller must be loaded into the system at generation time.

2-31. INITIATION SECTION, I.XX. Refer to the partial listing of the sample privileged driver (Figure 2-6). A standard I/O call to input from the device controller causes the calling program to be I/O suspended and driver to be entered at I.XX. The request code is checked for validity.

Because this driver can control just one device controller (unlike standard drivers) there is no need to configure it more than once. Therefore, the first time the driver is entered, it is configured and the switch at "FIRST" is cleared to that on all subsequent entries the configuration code is not executed.

The modification of the device controller trap cell is performed just once, after the configuring routine, and is not modified again on all later entries into the initiator. The trap cell is altered so that the device controller interrupts will be channeled to the P.XX subroutine instead of to RTIOC. The "JSB P.XX" instruction and its associated base page link are established via the small program "$JPXX" (see listing).

A counter, which is incremented in routine P.XX, is established for the number of readings to be taken; the buffer address for the storage of the data is saved, and the device controller is set up to initiate a reading and is encoded. The initiator then exits.

2-32. PRIVILEGED SECTION, P.XX. When the device controller interrupts, P.XX is entered as a result of the controller's trap cell modification.

Because entry is made directly into P.XX the routine must do the housekeeping which RTIOC does when entered from an interrupt. Before P.XX can turn the interrupt system back on for higher priority interrupts, it must ensure that the DMA channels cannot interrupt, save the old memory protect status, and set its new status. For RTE-III, it must also save the dynamic mapping system status at time of interrupt.

P.XX then loads and stores the data in the next unfilled buffer word. If there is yet another data point to be taken, P.XX sets up the device controller for the next reading, disables the interrupt system, encodes the device controller, restores memory protect status and its flag, turns the interrupt system back on, and exits. This basically resets the system to its state before P.XX was entered.

When the last reading is taken, P.XX disables the interrupt system, turns off the device controller, and sets up the driver for an immediate time-out. Before P.XX exits, it restores memory protect status and its flags, and turns the interrupt system back on, and (RTE-III only) restores the dynamic mapping system status.

2-33. COMPLETION SECTION, C.XX. The status of the device controller and the driver is now unchanged until the TBG interrupts. The TBG interrupt will cause a time-out (this is because −1 is set in EQT word 15), which will cause RTIOC to pass control to C.XX which returns a transmission log and a normal completion indication to RTIOC.

RTIOC then goes to its I/O completion section which reschedules the calling program and processes the controller request list as if it were a standard (non-privileged) controller.

2-34. ACCESS TO BUFFER IN USER AREA (RTE-III ONLY)

If the sample privileged driver described above were written to access a buffer within the user program area instead of common, the following changes would be necessary: (The caution note on the sample also applies here, since information is stored in the driver).

1. I.XX — The ID segment address of the calling program would have to be saved. This would be done as follows:

```
I.XX    STA   SCODE    Save select code
        LDA   1717B    Get ID Segment address
        STA   IDADR    Save it
```

2. P.XX — In addition to saving and restoring the DMS status, the User Map would have to be saved, reloaded to describe the user, and restored on exit. This would be done as follows:

```
P.XX
          .
          .
          .
P.X1    SSM   DMSTS    Save DMS status
        LDA   MAPAD    Get address of map storage area
        IOR   SIGN     Set sign bit indicating store to memory
(37μ SEC) USA          Save user map
        LDA   IDADR    Get ID segment address
(100μ SEC) JSB  $PVMP  Call routine to set map
        UJP   CONT     Enable user map and jump to CONT
```

```
CONT    LDA   MPTFL
        .
        .
        .
EXIT1   LDA   MAPAD   Get address of map storage area
        USA           Restore user map
        LDA   EOSV
        .
        .
        .
MPADR   DEF   MAP     Address of save Map area
IDADR   BSS   1       Save ID segment address
MAP     BSS   32      Save map area
DMSST   BSS   1       Save DMS status
SIGN    OCT   100000  Sign Bit
```

The subroutine $PVMP is a Type 8 subroutine and is loaded with each driver that accesses it. It is not re-entrant.

The times given in the example are approximate.

It is recommended that privileged drivers be designed for user communication through common or subsystem global. If this is done, the driver does not have the overhead of map switching; it simply saves and restores the state of the machine (including DMS status).

```
JIMB1  T=00003 IS ON CR00002 USING 00024 BLKS R=0000

0001   ASMB,R,L
0002         HED ** RT EXEC DRIVER <70> G.P.R. (OUTPUT) **
0003                         NAM DVR70
0004   *
0005   *
0006                         ENT I.70,C.70
0007   *
0008   *
0009   *   DRIVER 70 OPERATES UNDER THE CONTROL OF THE
0010   * I/O CONTROL MODULE OF THE REAL-TIME EXECUTIVE.
0011   * THIS DRIVER IS RESPONSIBLE FOR CONTROLLING
0012   * OUTPUT TRANSMISSION TO A 16 BIT EXTERNAL
0013   * DEVICE. <70> IS THE EQUIPMENT TYPE CODE ASSIGNED
0014   * GENERALLY TO THESE DEVICES.  I.70 IS THE
0015   * ENTRY POINT FOR THE *INITIATION* SECTION AND C.70
0016   * IS THE *COMPLETION* SECTION ENTRY.
0017   *
0018   *
0019   * - THE INITIATION SECTION IS CALLED FROM I/O
0020   *   CONTROL TO INITIALIZE A DEVICE AND INITIATE
0021   *   AN OUTPUT OPERATION.
0022   *
0023   *   I/O CONTROL SETS THE ADDRESS OF EACH WORD OF THE
0024   *   15 WORD EQT ENTRY (FOR THE DEVICE) IN THE SYSTEM
0025   *   COMMUNICATIONS AREA FOR BOTH INITIATOR AND CONTINUATOR
0026   *   SECTIONS.  THE DRIVER REFERENCES TO THE EQT ARE:
0027   *   -EQT1 THRU EQT15-
0028   *
0029   *   CALLING SEQUENCE:
0030   *
0031   *        (A) = SELECT CODE OF THE I/O DEVICE.
0032   *        P   JSB I.70
0033   *        P+1 -RETURN-
0034   *
0035   *
0036   *           (A) = 0, OPERATION INITIATED
0037   *           (A) = REJECT CODE
0038   *
0039   *               1. ILLEGAL REQUEST
0040   *               2. ILLEGAL MODE
0041   *
0042   * -THE COMPLETION SECTION IS CALLED BY CENTRAL
0043   *   INTERRUPT CONTROL TO CONTINUE OR COMPLETE
0044   *   AN OPERATION.
0045   *
0046   *     CALLING SEQUENCE:
0047   *
0048   *        (A) = SELECT CODE OF THE I/O DEVICE.
0049   *
0050   *        P   JSB C.70
0051   *        P+1 -COMPLETION RETURN-
0052   *        P+2   CONTINUATION RETURN-
0053   *
0054   *          (A) = 0, SUCCESSFUL COMPLETION WITH
0055   *                (B) = # OF WORDS TRANSMITTED.
0056   *          (A) = 2, TRANSMISSION ERROR DETECTED
```

Figure 2-6. RTE Driver Example (Sheet 1 of 4)

```
0057  *                  (B), SAME AS FOR (A)=0
0058  *
0059  *            - CONTINUATION RETURN: REGISTERS
0060  *                MEANINGLESS
0061  *
0062  * -RECORD FORMAT-
0063  *
0064  *     THIS DRIVER PROVIDES A 16 BIT BINARY
0065  *      WORD TRANSFER ONLY.
0066          HED < DRIVER 70 *INITIATION* SECTION >
0067  *      *** INITIATION SECTION ***
0068  *
0069  I.70  NOP             ENTRY FROM IOC
0070  *
0071          JSB SETIO      SET I/O INSTRUCTIONS FOR DEVICE
0072  *
0073          LDA EQT6,I     GET CONTROL WORD OF REQUEST,
0074          AND =B3         ISOLATE.
0075  *
0076          CPA =B1        IF REQUEST IS FOR INPUT
0077          JMP I.70,I      THEN REJECT.
0078          CPA =B2        PROCESS FOR WRITE REQUEST
0079          JMP D.X1        GO TO WRITE REQUEST
0080  *
0081  * REQUEST ERROR- CAUSE REJECT RETURN TO I/O CONTROL.
0082  *
0083          LDA =B2        SET A=2 FOR ILLEGAL CONTRL REQ.
0084          JMP I.70,I     -EXIT-
0085  *
0086  * WRITE REQUEST PROCESSING
0087  *
0088  D.X1  LDA EQT7,I     GET REQUEST BUFFER ADDRESS
0089          STA EQT9,I      AND SET AS CURRENT ADDRESS
0090          LDA EQT8,I     GET BUFFER LENGTH
0091          CMA,INA         SET NEGATIVE AND SAVE
0092          STA EQT10,I     AS CURRENT BUFFER LENGTH.
0093          SZA            CHECK LENGTH
0094          JMP D.X3       NON-ZERO
0095          LDA =B4        IMMEDIATE COMPLETION
0096          CLB            SET TLOG IN B-REG
0097          JMP I.70,I     IF ZERO
0098  *
0099  * CALL COMPLETION SECTION TO WRITE FIRST WORD.
0100  *
0101  D.X3  LDA P2          ADJUST RETURN
0102          STA C.70        TO INITIATOR SECTION.
0103          JMP D.X2       GO TO COMPLETION SECTION
0104  *
0105  IEXIT CLA             RETURN TO I/O CONTROL WITH
0106          JMP I.70,I      OPERATION INITIATED.
0107  *
0108          HED < DRIVER 70 *COMPLETION SECTION* >
0109  *
0110  *       *** COMPLETION SECTION ***
0111  *
0112  C.70  NOP             ENTRY
0113          LDB EQT1,I     SPURIOUS
0114          SZB,RSS         INTERRUPT?
```

Figure 2-6. RTE Driver Example (Sheet 2 of 4)

```
0115          JMP SPURI      YES = IGNORE
0116          JSB SETIO      SET I/O INSTRUCTIONS FOR DEVICE
0117  *
0118  D.X2    CLA            IF CURRENT BUFFER LENGTH = 0,
0119          CPA EQT10,I     THEN,GO TO
0120          JMP I.3         STATUS SECTION.
0121  *
0122          LDB EQT9,I     GET CURRENT BUFFER ADDRESS

0123          ISZ EQT9,I     ADD 1 FOR NEXT WORD
0124          LDA B,I        GET WORD
0125          ISZ EQT10,I    AND INDEX WORD COUNT
0126          NOP            IGNORE P+1 IF LAST WORD.
0127  *
0128  *
0129  I.1     OTA 0          OUTPUT WORD TO INTERFACE
0130  I.2     STC 0,C        TURN DEVICE ON
0131          RSS
0132  SPURI   STB EQT15,I    ZERO TIME-OUT CLOCK WORD
0133  *
0134          ISZ C.70       ADJUST RETURN TO P+2
0135          JMP C.70,I     =EXIT=
0136  *
0137          HED < DRIVER 70 *COMPLETION SECTION* >
0138  *
0139  * STATUS AND COMPLETION SECTION.
0140  *
0141  I.3     LIA 0          GET STATUS WORD
0142          AND =B77       STRIP OFF BITS
0143          STA B           AND SAVE IN B
0144          LDA EQT5,I     REMOVE PREVIOUS
0145          AND =B177400    STATUS BITS
0146          IOR B          SET NEW
0147          STA EQT5,I      STATUS BITS
0148  *
0149          CLA            SET NORMAL RETURN COND
0150          CPB =B4        ERROR STATUS BIT ON?
0151          LDA =B2        YES, SET ERROR RETURN
0152  *
0153          LDB EQT8,I     SET (B) = TRANSMISSION LOG
0154  *
0155  I.4     CLC 0          CLEAR DEVICE
0156  *
0157          JMP C.70,I     =EXIT FOR COMPLETION
0158  *
0159  *
0160  * SUBROUTINE <SETIO> CONFIGURES I/O INSTRUCTIONS.
0161  *
0162  SETIO   NOP
0163          IOR LIA        COMBINE LIA WITH I/O
0164          STA I.3        SELECT CODE AND SET.
0165  *
0166          ADA =B100      CONSTRUCT OTA INSTRUCTION
0167          STA I.1
0168  *
0169          ADA =B1100     CONSTRUCT STC,C INSTRUCTION
0170          STA I.2
0171  *
```

Figure 2-6. RTE Driver Example (Sheet 3 of 4)

```
0172          IOR =B4000     CONSTRUCT CLC INSTRUCTION
0173          STA I,4
0174  *
0175          JMP SETIO,I    =RETURN=
0176  *
0177          HED < DRIVER 70 *COMPLETION SECTION* >
0178  *
0179  * CONSTANT AND STORAGE AREA
0180  *
0181  A       EQU 0          A=REGISTER
0182  B       EQU 1          B=REGISTER
0183  *
0184  LIA     LIA 0
0185  P2      DEF IEXIT-1
0186  *
0187  *** SYSTEM AND BASE PAGE COMMUNICATIONS AREA ***
0188  *
0189  .       EQU 16500B
0190  *
0191  *    I/O MODULE/DRIVER COMMUNICATION
0192  *
0193  EQT1    EQU .+8
0194  EQT2    EQU .+9
0195  EQT3    EQU .+10
0196  EQT4    EQU .+11
0197  EQT5    EQU .+12
0198  EQT6    EQU .+13
0199  EQT7    EQU .+14
0200  EQT8    EQU .+15
0201  EQT9    EQU .+16
0202  EQT10   EQU .+17
0203  EQT11   EQU .+18
0204  EQT12   EQU .+81
0205  EQT13   EQU .+82
0206  EQT14   EQU .+83
0207  EQT15   EQU .+84
0208  *
0209  *
0210          END
```

Figure 2-6. RTE Driver Example (Sheet 4 of 4)

```
JIMB2  T=00003 IS ON CR00002 USING 00021 BLKS R=0000

0001   ASMB,R,L,T,B
0002   *
0003   *
0004   *  DRIVER WITH PRIVILEGED INTERRUPT
0005   *
0006   *
0007           NAM DVRXX
0008           ENT I.XX,P.XX,C.XX
0009           EXT $JPXX
0010   *
0011   *
0012   *
0013   *  CALLING SEQUENCE:
0014   *      JSB EXEC        CALL EXEC
0015   *      DEF *+5         RETURN POINT
0016   *      DEF RCODE       REQUEST CODE
0017   *      DEF CONWD       CONTROL WORD
0018   *      DEF BUFFR       ADDRESS OF BUFFER
0019   *      DEF LENTH       LENGTH OF BUFFER
0020   *
0021   *
0022   *
0023   *
0024   *  CAUTION:  THIS DRIVER WILL NOT WORK WITH MORE THAN
0025   *      ONE SUBSYSTEM.  IF MORE THAN ONE SUBSYSTEM
0026   *      EXISTS IN A SYSTEM, BOTH DVRXX AND $JPXX MUST
0027   *      BE RE-ASSEMBLED WITH ALL THE NAMES CONTAINING
0028   *      'XX' CHANGED TO SOME OTHER NUMBER.  THEN ONE
0029   *      DRIVER PER SUBSYSTEM MUST BE PUT INTO THE SYSTEM
0030   *      AT GENERATION TIME.
0031   *
0032   *  INITIATION SECTION
0033   *
0034   I.XX   NOP
0035           STA SCODE       SAVE SELECT CODE
0036           LDA EQT6,I      REQUEST CODE TO A
0037           AND M77
0038           CPA =B1         READ REQUEST?
0039           JMP *+3          YES
0040   REJCT   CLA,INA         NO - ERROR
0041           JMP I.XX,I      REJECT RETURN
0042   FIRST  RSS             CONFIGURE FIRST
0043           JMP INIT         TIME ONLY
0044   *
0045           LDA SCODE
0046           IOR LIA         CONFIGURE
0047           STA IOO          IO INSTRUCTIONS
0048           .
0049           .
0050           .
0051           LDA $JPXX       SET TRAP CELL TO
0052           STA SCODE,I      JSB P.XX
0053           LDA EQT4,I      CLEAR EQT4 BIT12
0054           IOR BIT12        TO ALLOW NORMAL
0055           XOR BIT12        TIME OUT.
```

Figure 2-7. RTE Privileged Driver Example (Sheet 1 of 4)

```
0056          STA EQT4,I
0057          LDA EQT15       SAVE EQT15
0058          STA EQ15         AND EQT4 ADDRESSES
0059          LDA EQT4         FOR LATER.
0060          STA EQ4
0061          CLA             SET SO AS NOT TO
0062          STA FIRST        CONFIGURE AGAIN
0063  *
0064  INIT    LDA EQT8,I      NUMBER OF CONVERSIONS TO A
0065          CMA,INA         NEGATE FOR
0066          STA CVCTR        CONVERSION COUNTER
0067          SSA,RSS         REJECT IF
0068          JMP REJCT        NUMBER <=0
0069          LDA EQT7,I      SAVE DATA BUFFER
0070          STA DAPTR        ADDRESS FOR P.XX
0071          JSB READ        START A READING
0072  IO1     STC IO,C        ENCODE DEVICE
0073          JMP I.XX,I      RETURN
0074  *
0075  READ    NOP             ROUTINE CONTAINING
0076          .                CONFIGURED IO
0077          .                INSTRUCTIONS TO
0078          .                SET UP THE DEVICE
0079          JMP READ,I       TO INITIATE ONE READING
0080  *
0081  *  PRIVILEGED INTERRUPT ROUTINE
0082  *
0083  P.XX    NOP
0084          CLF 0           TURN OFF INTERRUPTS
0085          CLC 6           TURN OFF
0086          CLC 7            DMA INTERRUPTS
0087          STA ASV         S
0088          STB BSV          A
0089          ERA,ALS           V
0090          SOC                E
0091          INA
0092          STA EOSV         REGISTERS
0093          LDA MPTFL       SAVE MEMORY
0094          STA MPFSV        PROTECT FLAG
0095          CLA,INA         TURN OFF MEMORY
0096          STA MPTFL        PROTECT FLAG
0097          STF 0           TURN ON INTERRUPTS
0098  *
0099          .               LOAD IN DATA
0100          .                FROM DEVICE
0101          .                VIA IO INSTRUCTIONS
0102  *
0103          STA DAPTR,I     STORE IN DATA BUFFER
0104          ISZ CVCTR       LAST CONVERSION
0105          RSS              NO
0106          JMP DONE         YES
0107          ISZ DAPTR       SET UP FOR
0108          JSB READ         NEXT CONVERSION
0109          CLF 0           TURN OFF INTERRUPTS
0110  IO4     STC IO,C        ENCODE DEVICE
0111  *
0112  EXIT    LDA MPFSV       WAS MEMORY
0113          SZA              PROTECT ON?
```

Figure 2-7. RTE Privileged Driver Example (Sheet 2 of 4)

```
0114          JMP EXIT1       NO, FORGET DMA'S
0115          LDB INTBA      TURN
0116          LDA B,I         DMA'S
0117          SSA              BACK
0118          STC 6              ON
0119          INB                 IF
0120          LDA B,I              THEY
0121          SSA                    WERE
0122          STC 7                    ON
0123  EXIT1   LDA EUSV       RESTORE
0124          CLU              E AND
0125          SLA,ELA           O
0126          STF 1              FLAGS
0127          LDB BSV        RESTORE B
0128          LDA MPFSV      RESTORE MEMORY PRCTECT
0129          STA MPTFL       FLAG IN SYSTEM
0130          SZA            MEMORY PROTECT ON?
0131          JMP EXIT2       NO
0132          LDA ASV         YES, RESTORE A
0133          STF 0          TURN ON INTERRUPTS
0134          STC 5          SET MEMORY PRCTECT
0135          JMP P.XX,I     RETURN
0136  EXIT2   LDA ASV        RESTORE A
0137          STF 0          TURN ON INTERRUPTS
0138          JMP P.XX,I     RETURN
0139  *
0140  DONE    CLF 0          TURN OFF INTERRUPTS
0141  I07     CLC IO         TURN OFF DEVICE
0142          CCA            SET TIME OUT FOR
0143          STA EQ15,I      ONE TICK AND SET
0144          LDA EQ4,I       BIT12 IN EQT4 SO
0145          IOR BIT12       RTIOC WILL CALL
0146          STA EQ4,I       C.XX ON TIME OUT.
0147          JMP EXIT       GO TO EXIT
0148  *
0149  *
0150  *  COMPLETION SECTION
0151  *
0152  *
0153  C.XX    NOP
0154          CLA            SET UP FOR NORMAL RETURN
0155          LDB EQT8,I     TRANSMISSION LOG TO B
0156          JMP C.XX,I     RETURN
0157  *
0158  *
0159  *  CONSTANTS AND TEMPORARIES
0160  *
0161  *
0162  SCODE   OCT 0
0163  CVCTR   OCT 0
0164          .
0165          .
0166          .
0167  LIA     LIA 0
0168  M77     OCT 77
0169  DAPTR   DEF 0
0170  ASV     OCT 0
0171  BSV     OCT 0
```

Figure 2-7. RTE Privileged Driver Example (Sheet 3 of 4)

```
0172  EOSV   OCT 0
0173  MPFSV  OCT 0
0174  EQ4    NOP
0175  EQ15   NOP
0176  BIT12  OCT 10000
0177  *
0178  *
0179  *  SYSTEM COMMUNICATION AREA
0180  *
0181  .      EQU 1650B
0182  INTBA  EQU .+4
0183  EQT4   EQU .+11
0184  EQT6   EQU .+13
0185  EQT7   EQU .+14
0186  EQT8   EQU .+15
0187  EQT15  EQU 1774B
0188  XA     EQU .+49
0189  XB     EQU .+50
0190  XEU    EQU .+51
0191  MPTFL  EQU .+80
0192  A      EQU 0
0193  B      EQU 1
0194         END
```

Figure 2-7. RTE Privileged Driver Example (Sheet 4 of 4)

```
 T=00004 IS ON CR00002 USING 00023 BLKS R#0203

ASMB,R,L,T,8
*
*
*  DRIVER WITH PRIVILEGED INTERRUPT
*
*
       NAM DVRXX
       ENT I.XX,P.XX,C.XX
       EXT $JPXX
*
*
*
*  CALLING SEQUENCE:
*      JSB EXEC       CALL EXEC
*      DEF *+5        RETURN POINT
*      DEF RCODE      REQUEST CODE
*      DEF CONWD      CONTROL WORD
*      DEF BUFFR      ADDRESS OF BUFFER
*      DEF LENTH      LENGTH OF BUFFER
*
*
*
*
*  CAUTION:  THIS DRIVER WILL NOT WORK WITH MORE THAN
*      ONE SUBSYSTEM.  IF MORE THAN ONE SUBSYSTEM
*      EXISTS IN A SYSTEM, BOTH DVRXX AND $JPXX MUST
*      BE RE-ASSEMBLED WITH ALL THE NAMES CONTAINING
*      'XX' CHANGED TO SOME OTHER NUMBER.  THEN ONE
*      DRIVER PER SUBSYSTEM MUST BE PUT INTO THE SYSTEM
*      AT GENERATION TIME.
*
*  INITIATION SECTION
*
I.XX   NOP
       STA SCODE      SAVE SELECT CODE
       LDA EQT6,I     REQUEST CODE TO A
       AND M77
       CPA =B1        READ REQUEST?
       JMP *+3         YES
REJCT  CLA,INA         NO - ERROR
       JMP I.XX,I     REJECT RETURN
FIRST  RSS            CONFIGURE FIRST
       JMP INIT        TIME ONLY
*
       LDA SCODE
       IOR LIA        CONFIGURE
       STA IO0         IO INSTRUCTIONS
       .
       .
       .
       LDA $JPXX      SET TRAP CELL TO
       STA SCODE,I     JSB P.XX
       LDA EQT4,I     CLEAR EQT4 BIT12
       IOR BIT12       TO ALLOW NORMAL
       XOR BIT12       TIME OUT.
       STA EQT4,I
       LDA EQT15      SAVE EQT15
       STA EQ15        AND EQT4 ADDRESSES
```

Figure 2-8. RTE-III Privileged Driver Example (Sheet 1 of 4)

```
        LDA EQT4        FOR LATER.
        STA EQ4
        CLA             SET SO AS NOT TO
        STA FIRST        CONFIGURE AGAIN
*
INIT    LDA EQT8,I      NUMBER OF CONVERSIONS TO A
        CMA,INA         NEGATE FOR
        STA CVCTR        CONVERSION COUNTER
        SSA,RSS         REJECT IF
        JMP REJCT        NUMBER <=0
        LDA EQT7,I      SAVE DATA BUFFER
        STA DAPTR        ADDRESS FOR P.XX
        JSB READ        START A READING
IO1     STC IO,C        ENCODE DEVICE
        JMP I.XX,I      RETURN
*
READ    NOP             ROUTINE CONTAINING
        .                CONFIGURED IO
        .                INSTRUCTIONS TO
        .                SET UP THE DEVICE
        JMP READ,I       TO INITIATE ONE READING
*
*  PRIVILEGED INTERRUPT ROUTINE
*
P.XX    NOP
        CLF 0           TURN OFF INTERRUPTS
        CLC 6           TURN OFF
        CLC 7            DMA INTERRUPTS
        STA ASV         S
        STB BSV          A
        ERA,ALS           V
        SOC                E
        INA
        STA EOSV         REGISTERS
        STX XSV         SAVE INDEX
        STY YSV          REGISTERS
P.X1    SSM DMSTS       SAVE DMS STATUS
CONT    LDA MPTFL       SAVE MEMORY
        STA MPFSV        PROTECT FLAG
        CLA,INA         TURN OFF MEMORY
        STA MPTFL        PROTECT FLAG
        STF 0           TURN ON INTERRUPTS
*
        .               LOAD IN DATA
        .                FROM DEVICE
        .                VIA IO INSTRUCTIONS
*
        STA DAPTR,I     STORE IN DATA BUFFER
        ISZ CVCTR       LAST CONVERSION
        RSS              NO
        JMP DONE         YES
        ISZ DAPTR       SET UP FOR
        JSB READ         NEXT CONVERSION
        CLF 0           TURN OFF INTERRUPTS
IO4     STC IO,C        ENCODE DEVICE
*
EXIT    LDA MPFSV       WAS MEMORY
        SZA              PROTECT ON?
        JMP EXIT1        NO, FORGET DMA'S
        LDB INTBA       TURN
```

Figure 2-8. RTE-III Privileged Driver Example (Sheet 2 of 4)

```
        LDA B,I         DMA'S
        SSA              BACK
        STC 6             ON
        INB                IF
        LDA B,I             THEY
        SSA                  WERE
        STC 7                 ON
EXIT1   LDA EOSV        RESTORE
        CLO              E AND
        SLA,ELA           O
        STF 1              FLAGS
        LDB BSV         RESTORE B
        LDX XSV         RESTORE INDEX
        LDY YSV          REGISTERS
        LDA MPFSV       RESTORE MEMORY PROTECT
        STA MPTFL        FLAG IN SYSTEM
        SZA             MEMORY PROTECT ON?
        JMP EXIT2        NO
        JRS DMSTS EX1
  EX1   LDA ASV          YES, RESTORE A
        STF 0           TURN ON INTERRUPTS
        STC 5           SET MEMORY PROTECT
        JMP P.XX,I           RESTORE DMS
*                             STATUS AND
*                             RETURN
EXIT2   LDA ASV         RESTORE A
        STF 0           TURN ON INTERRUPTS
        JRS DMSTS P.XX,I    RESTORE DMS
*                             STATUS AND
*                             RETURN
*
DONE    CLF 0           TURN OFF INTERRUPTS
IO7     CLC IO          TURN OFF DEVICE
        CCA             SET TIME OUT FOR
        STA EQ15,I       ONE TICK AND SET
        LDA EQ4,I        BIT12 IN EQT4 SO
        IOR BIT12        RTIOC WILL CALL
        STA EQ4,I        C.XX ON TIME OUT.
        JMP EXIT        GO TO EXIT
*
*
*  COMPLETION SECTION
*
*
C.XX    NOP
        CLA             SET UP FOR NORMAL RETURN
        LDB EQT8,I      TRANSMISSION LOG TO B
        JMP C.XX,I      RETURN
*
*
*  CONSTANTS AND TEMPORARIES
*
*
SCODE   OCT 0
CVCTR   OCT 0
        .
        .
        .
LIA     LIA 0
M77     OCT 77
DAPTR   DEF 0
```

Figure 2-8. RTE-III Privileged Driver Example (Sheet 3 of 4)

```
ASV    OCT 0
BSV    OCT 0
EOSV   OCT 0
MPFSV  OCT 0
EQ4    NOP
EQ15   NOP
BIT12  OCT 10000
*
*
*  SYSTEM COMMUNICATION AREA
*
.      EQU 1650B
INTBA  EQU .+4
EQT4   EQU .+11
EQT6   EQU .+13
EQT7   EQU .+14
EQT8   EQU .+15
EQT15  EQU 1774B
XA     EQU .+49
XB     EQU .+50
XEO    EQU .+51
MPTFL  EQU .+80
A      EQU 0
B      EQU 1
       END



JIMB3  T=00004 IS ON CH00105 USING 00001 BLKS R=0000

00001  ASMB,R,L,B
00002          NAM $JPXX
00003          ENT $JPXX
00004          EXT P.XX
00005  $JPXX   JSB P.XX
00006          END
```

Figure 2-8. RTE-III Privileged Driver Example (Sheet 4 of 4)

READER COMMENT SHEET

RTE Operating System

Drivers and Device Subroutines

92200-93005 Jul 1976

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

Manager, Technical Publications
Hewlett-Packard Company
Data Systems Division
11000 Wolfe Road
Cupertino, California 95014

FOLD FOLD