RTE Driver DVR61 For HP 6940A/6941A Bidirectional Multiprogrammer

Programming and Operating Manual

HEWLETT PACKARD

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

PART NO. 29100-93003

Printed in U.S.A. 3/76

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 outer margin of the page. Original pages do not include a change number and are indicated as change number 0 on this page. Insert latest changed pages and destroy superseded pages.

Change 0 (original) . . . . . MAR 1976

SPECIAL NOTICE

RTE Driver DVA72, part number 09611-16005, allows either local or remote computer control of the 6940A Multiprogrammer and is capable of operating up to eight 6940A's concurrently. The terms local and remote refer to the hardware interface between the computer and a 6940A, and not to the setting of the LOCAL/REMOTE switch on the 6940A: a local 6940A is interfaced to the computer by up to 50 feet of cable; a remote 6940A is interfaced by a remote interface kit and up to 10,000 feet of cable.

This manual is applicable to DVA72 when the 6940A is locally controlled. When the 6940A is remotely controlled, the same information is applicable, except as follows:

a. Status bit 7 of EQT5 can be used to check for transmission errors to or from a remote 6940A; this bit is set whenever such a transmission error occurs.

b. The Read Operator Data requests cannot be used. When entering the EQT number of the 6940A into the Device Reference Table during RTGEN, also enter a subchannel number: enter a 0 if the 6940 is local; enter a 1 if it is remote.

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

CONTENTS

ILLUSTRATIONS

TABLES

SECTION I
GENERAL INFORMATION

1-1. GENERAL DESCRIPTION

1-2. This manual contains information and procedures that allow the user to write application programs using FORTRAN or Assembly language programs and RTE Driver DVR61. Section III provides information required when configuring DVR61 into a Real-Time Executive Operating System.

1-3. RTE HP 6940A Multiprogrammer Driver DVR61 controls the transfer to data from user programs to and from one or more HP 6940A Multiprogrammers (and, if applicable, up to fifteen HP 6941A Multiprogrammer Extenders) attached to each HP 6940A. The DVR61 driver executes on any RTE system to exchange data with a master HP 6940A Multiprogrammer via Microcircuit Interface Card 12566B which is provided as part of Interface Kit 14550A.

1-4. Through its I/O cards, the HP 6940A and HP 6941A Multiprogrammers provide for bidirectional data exchange between user programs and external (user supplied) devices. Since each HP 6940A and HP 6941A Multiprogrammer unit can accommodate up to 15 I/O cards and since up to fifteen HP 6941A extenders in addition to the master HP 6940A Multiprogrammer can be utilized in a single system, the multiprogrammer system can convert one 16-bit computer I/O channel into a total of up to 240 12-bit I/O channels.

1-5. Output data distribution (via, of course, the Microcircuit Interface Card) to a device requires a multiprogrammer output card which is installed in the multiprogrammer's I/O slots and connected, in turn, to the applicable device. All multiprogrammer output cards store programmed data and, depending upon the type of card employed, provide some form of output signal or action to the external device. For example, the data to a multiprogrammer unit I/O slot containing a resistance output card can be used to vary the voltage of a dc power supply through its remote programming terminals.

1-6. Input data multiplexing form external devices to user programs requires associated multiprogrammer input cards installed in the multiprogrammer's I/O slots and connected to the device. Digital input cards are available to receive logic-level or relay contact closure data from external devices for input to user programs. Also the event sense card is available and allows the user to monitor input lines for a change of state. The driver can be operated in the sense mode in which event sense cards can be enabled whenever I/O data transfers are not in progress. Finally, one output card, the relay output/readback card, can be read like an input card. The data from the card, however, is not external data, but reflects, instead, the output data sent to the relays on the card and is used for status checking.

1-7. OPERATING ENVIRONMENT

1-8. The operating environment for the software must be an HP 2100 Series Computer, an RTE Operating System, the interface Kit hardware, and the multiprogrammer components. Refer to the HP 6940A Interface Kit Operating and Service Manual (HP Part No. 14543-90004) and to the HP 6940A/6941A Multiprogrammer Operating and Service Manual (HP Part No. 06940-90003) for hardware details.

1-9. COMPONENTS

1-10. The following components are included with DVR61:

a. This manual
b. Driver DVR 61 binary tape, HP Part No. 14907-60001.

CAUTION

RTE Driver DVR61 has been changed from Revision C to Revision D, described as follows:

DVR61 Revision D permits the user to assign a time-out value that is applicable during normal I/O transfers even if the HP 6940A is to be used in the sense mode. If a time-out occurs during a normal I/O transfer, the driver sets the status word (EQT5, bit 2) and makes a completion return to RTE. Thus, the user's calling program can check for time-out; the program does not hang up and the multiprogrammer I/O channel is not set to the down status. Note that the driver's processing of alarm-pending time outs during the sense mode has not been changed. Revision D applies to: Table 2-1 (Time Out Processing specification), Paragraphs 2-25(c), 2-43, 2-73, 2-74, and 3-4 (delete step 3), and Figure 2-14.

The handshake flag delay has been increased from 50 usec to approximately 500 usec, Thus, 500 usec is now allowed before a flag error is reported and a transfer aborted. Handshake flag references appear throughout the manual; change 50 usec to 500 usec wherever applicable.

An 8-usec delay has been inserted between the time data is output to the HP 6940A and the setting of the Gate. This allows the data and address lines to settle before the bits are strobed into the HP 6940A. This change applies to all I/O transfers requiring a gate (Paragraph 2-31).

A delay has been inserted between the addressing of an input card (in a poll request) and the reading of its data. This change also allows the address lines to settle before data is accepted. This delay applies to all poll requests (Paragraph 2-46).

SECTION II
APPLICATION INFORMATION

2-1. GENERAL

2-2. This section details the calls to the driver, provides programming examples and describes any results of the hardware/software marriage where the hardware may influence software techniques.

2-3. Before writing programs using DVR61, it is recommended that the user consult the operating instructions (Section III) in the HP 6940A Multiprogrammer Operating and Service Manual and the HP 6940A Interface Kit Operating and Service Manual. These provide detailed instructions on how to effect interface with the computer, including a description of operating modes, command structure and computer programming techniques.

2-4. DRIVER SPECIFICATIONS

2-5. Ten I/O Read/Write EXEC calls are processed by the driver, eight Read and two Write requests (see Table 2-1). In addition, the driver processes three I/O Control EXEC requests which are provided (along with two of the Read/Write EXEC calls) to control the driver's automatic sense mode of operation. Depending upon the I/O job requested, the driver is either an interrupt driver that utilizes the computer interrupt system and operates under control of Real-Time Input Output Control (RTIOC) or it completes the I/O job without using the interrupt system (or RTIOC). For non-interrupt transfers (the Read Direct requests, all four Poll requests, the Write with Handshake Flag request, and the three I/O control requests), the driver does not return control to RTE until the request is completed. For interrupt transfers (the normal Read and Write requests and the Read Operator Data request), the driver returns control to RTE after each word is transferred. However, if the multiprogrammer system is operated in the fast mode, it is possible that it will immediately interrupt when RTE has finished its "housekeeping" chores (and turned the interrupt system back on) so that the driver can appear to run uninterrupted even for those I/O functions that operate under control of the interrupt system. These possibilities, and suggested methods of avoiding them, are discussed in more detail later in this Section.

2-6. Request parameters defined in the call to the driver (see Paragraph 2-12) are passed (via applicable basepage locations EQT 1 through EQT 15) to the initiation section of the driver by the RTIOC module of RTE. The first time the driver is called for any request, it determines if the user has supplied an alarm processing program during system generation time. If a program was supplied, the driver stores its ID segment address (a positive value) in EQT 13 and then executes the requested I/O job. If an alarm processing program was not supplied, EQT 13 is set to -1 and the I/O request is executed. Thus, the driver subsequently uses EQT 13 to determine if the sense mode will be allowed on a 6940A channel (if EQT 13 <Ø, the sense mode bit is not set by the driver even if the user tries to set it).

Table 2-1. Driver DVR61 Specifications

VALID REQUESTS — Request Code (See Paragraph 2-12)

I/O READ/WRITE

I. READ	Request Type (Read/Write/Control)	Request Function
1. Normal Read	1	0
2. Read Direct (No Gate)	1	2
3. Read Operator Data	1	3
4. Poll to First Input	1	4
5. Poll All Inputs	1	5
6. Read Direct (With Gate)	1	6
7. Event Sense Poll to First Input	1	7
8. Event Sense Poll All Inputs	1	8
II. WRITE		
1. Normal Write	2	0
2. Write with Handshake Flag	2	1

I/O CONTROL

1. Set Sense Mode Enable Bit (EQT 5, Bit Ø)	3	20_8
2. Reset Sense Mode Enable Bit (EQT 5, Bit Ø)	3	21_8
3. Clear Alarm Pending Bit (EQT 5, Bit 1)	3	25_8

DATA FORMAT

One 16-bit word from the buffer list makes one 16-bit system or device control word to multiprogrammer.

One 16-bit buffer word location required for each multiprogrammer word.

Data formats not checked by driver. (See Figure 2-1.)

BUFFER FORMATS

Input and output buffer requirements depend on request type (see Figures 2-2 to 2-6).

MODE OF OPERATION

Interrupt/non-interrupt/sense mode.

ERROR PROCESSING

1. Request Validity - The request is rejected if not a valid READ or WRITE (request type code no 1 or 2) or CONTROL (request type code not 3).

Table 2-1. Driver DVR61 Specifications (Continued)

ERROR PROCESSING (Continued) 2. Multiprogrammer Handshake Flag Timing - Request terminated if multiprogrammer Handshake Mode Flag not returned in approximately 50 μsec. (Not applicable for normal read or write requests.) TIME OUT PROCESSING In sense mode, driver handles time outs. If sense mode interrupt occurs and user's alarm processing program not dormant, driver notifies RTE to call it every 50 msec until it can schedule user's program (i.e. it is dormant).

2-7. For data transfer requests, the driver next configures its I/O instructions for the appropriate channel number (also passed to the driver from RTIOC VIA THE A-Register) and then checks the validity of the request, rejecting it if it is not legal. If the request is one that does not use the interrupt system, the driver processes it to completion entirely in the initiation section. If the request is one that uses the interrupt system, the driver initiation section (after preliminary processing) transfers to the continuation/completion section where the first word is transferred. After transferring the word, the driver returns control to RTIOC. For the requests that use the computer interrupt system, then, subsequent data processing and word transfers are performed by the continuation/completion section when it is given service by RTE in response to a computer interrupt from the multiprogrammer. The multiprogrammer interrupt location contains a link to the CIC module of RTE which is given service when the multiprogrammer Flag notifies the Microcircuit Interface Card to generate an interrupt request. CIC, then, ultimately gives service to the driver continuation/completion section. After all buffer words (only one for the Read Operator Data request) have been transferred via the continuation/completion section, the driver completes the request. Note that whenever it completes any request (data transfer or control) and before it returns to RTE, the driver checks if the sense mode is required (sense mode bit is on and no previous sense mode interrupt is waiting for service from the alarm processing program) and, if it is, programs the channel to the interrupt enable mode (IEN, ISL, SYS, and TME are all on). One further note, since the interrupt enable mode may have been programmed on when the driver completed the last request, it is immediately programmed off (the Gate to the multiprogrammer is reset) at the beginning of all requests. Thus, the sense mode is only programmed when I/O is not in process. Of course, if the sense bit is not turned on by the user (or if a previous alarm interrupt is already pending), the driver does not program the interrupt enable mode when it completes an I/O request.

2-8. The three I/O control requests are all executed in the initiator and do not result in any data transfer except, of course, that the channel is programmed to the interrupt enable mode as necessary. Note, though, that even if the set Sense Mode Control request is issued, the driver will not set the sense mode bit (which defines whether or not the driver automatically programs the interrupt enable mode between I/O requests) if the user does not provide an alarm processing program at system generation time.

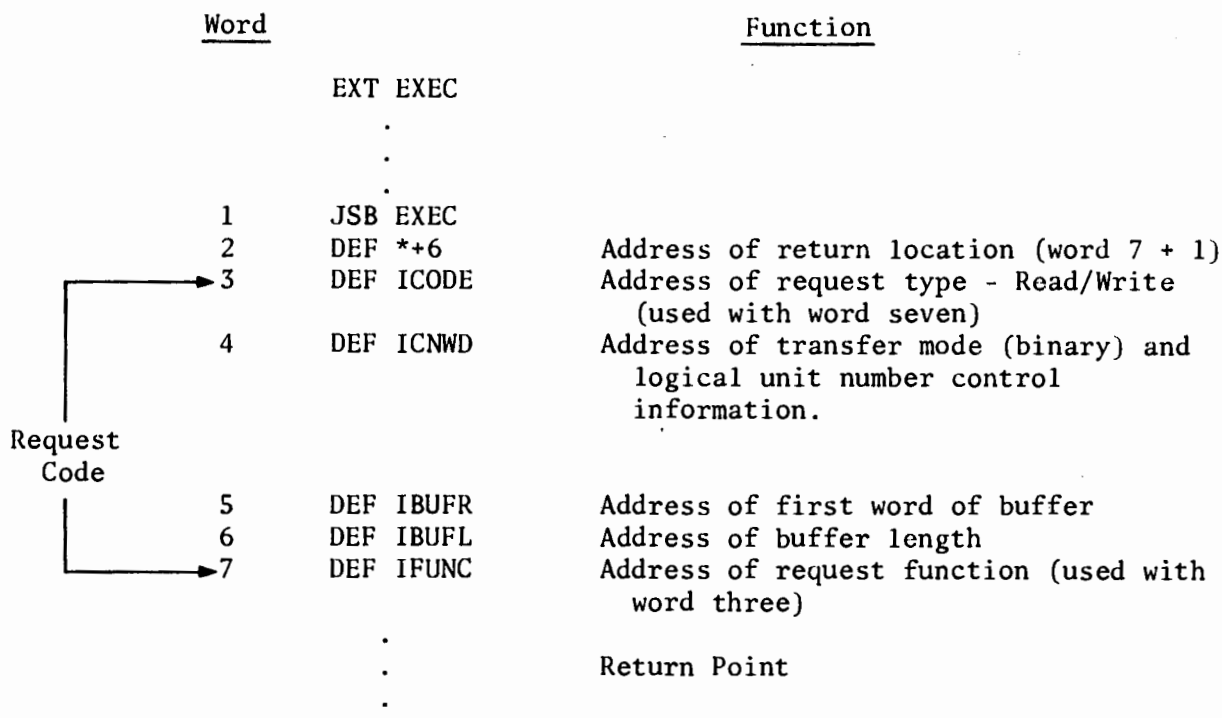
2-9. EXEC CALLS

2-10. DVR 61, like any other RTE driver, is accessed by the user's program via an EXEC call to the driver. The EXEC call can be given in HP Assembly language, FORTRAN, or ALGOL format, all of which are described below. No attempt should be made to access DVR 61 through FORTRAN or ALGOL READ or WRITE statements.

2-11. HP ASSEMBLY LANGUAGE READ/WRITE EXEC CALL

2-12. The general form of the DVR 61 Assembly language Read/Write EXEC call is given below. The format for the I/O control EXEC call is discussed in Paragraph 2-14. Notice that all of the request labels (ICODE, ICNWD, etc.) are examples only and may be any integer variable name desired. The names used do, however, conform to general RTE usage.

I/O Read/Write EXEC Call Format



2-13. The request parameters are defined as follows:

a. Word Two. Word two contains the address of the point in the user's program that will be executed when RTE returns control to the program. Since RTE handles all communication with the driver and assuming the requested function was successfully completed, when the program is again given service by RTE the word two location will be accessed and, as a result, the next executable statement in the program will be executed.

b. Word Three. Word three contains the address of ICODE which in conjunction with word seven (IFUNC) forms the request code. Word three

specifies the type of transfer, Read or Write, the driver is to execute as follows:

ICODE = 1 specifies Read request
ICODE = 2 specifies Write request

If IFUNC = 0, ICODE specifies a normal (interrupt system-controlled Read/Write request. If IFUNC ≠ 0, the read or write request type specified by ICODE is defined by the value of IFUNC. For Read/Write EXEC calls, word three should not be any value except 1 or 2.

c. Word Four. Word four contains the address of control word ICNWD which specifies the logical unit number of a 6940A for the I/O transfer as well as the mode of data transfer that is to take place. Since the transfer mode is always binary for the multiprogrammer, the M bit (bit 6) must be 1 and all other control bits in the word zero. Thus, ICNWD always equals the logical unit number plus 100_8 (i.e., if the 6940A is assigned logical unit number 17_8, ICNWD = 117_8).

d. Word Five. Word five contains the address of IBUFR, the first word of the I/O job buffer. The buffer format requirements for each of the I/O requests are discussed in subsequent paragraphs.

e. Word Six. Word six contains the address of IBUFL which specifies the length of the buffer (total number of words) involved in the transfer. The driver uses the word count in processing the I/O request.

f. Word Seven. Word seven contains the address of IFUNC which in conjunction with word three (ICODE) specifies the request to be executed as listed below. Notice that if IFUNC is 0, ICODE alone specifies either a normal (interrupt system-controlled) Read or normal Write request. If however, IFUNC ≠ 0, the type of request specified by ICODE is defined by IFUNC as shown. Further, if IFUNC is not in the range $0 \leq$ IFUNC ≤ 8; the driver rejects the request.

ICODE	IFUNC	REQUEST
1	0	Normal Read
2	0	Normal Write
2	1	Write with Handshake Flag (fast)
1	2	Read Direct (No Gate)
1	3	Read Operator Data
1	4	Poll to first Interrupt (Hit)
1	5	Poll All Inputs
1	6	Read Direct (With Gate)
1	7	Event Sense Poll to First Interrupt
1	8	Event Sense Poll All

2-14. HP ASSEMBLY LANGUAGE CONTROL EXEC CALL

2-15. The general form of the DVR 61 Assembly language I/O Control EXEC call is given on next page. Again, the labels can be any integer variable label desired, but they do conform to general RTE usage.

I/O Control EXEC Call Format

Word		Function
	EXT EXEC	
	.	
	.	
	.	
1	JSB EXEC	
2	DEF * +3	Address of return location (word 4 + 1)
3	DEF ICODE	Address of request type (must be 3)
4	DEF ICNWD	Address of control word
	Return Point	
	.	
	.	
	.	

2-16. The request parameters are defined as follows:

a. Word Three. Word three contains the address of ICODE which, for I/O Control requests, must always be 3.
b. Word Four. Word four contains the address of ICNWD, the control word that defines the desired request and the applicable logical unit number. ICNWD consists of two fields, each of which is 2-octal integers long as follows:

0 0 0 0	FUNCTION CODE	LOGICAL UNIT #
15 12	11 6	5 0

The function code defines the desired Control request as follows:

Function Code	Request
20_8	Set Sense Mode Bit (EQT5(Ø)=1)
21_8	Reset Sense Mode Bit (EQT5(Ø)=Ø)
25_8	Rest Alarm Pending Bit (EQT5(1)=Ø)

Of course, the logical unit number field is merely appended to the function field in constructing ICNWD. For example:

ICNWD OCT 2Ø12 would request that the sense mode bit be set for logical unit $10_{10}(12_8)$.

or

ICNWD OCT 2517 would request that the alarm pending bit be cleared for logical unit 15_{10} (17_8).

2-17. FORTRAN EXEC CALLS

2-18. RTE drivers (including the DVR 61) are called in FORTRAN using the standard RTE subroutine subprogram EXEC, whose general form for calling DVR 61 is:

a. Read/Write EXEC call

```
CALL EXEC (ICODE, ICNWD, IBUFR, IBUFL, IFUNC)
```

The parameters of the EXEC call can be integer variables as shown or, except for IBUFR, integer constants. In either case, the parameters define the type of request, logical unit number of the multiprogrammer, buffer address, and buffer length as previously described for the HP Assembly language EXEC call (Paragraph 2-11).

b. Control EXEC call

```
CALL EXEC (ICODE, ICNWD)
```

Again, the parameters of the EXEC call can be integer variables or constants. Of course ICODE must be equal to 3 to define the request as a control type. ICNWD defines the control function as described for the Assembly language EXEC call (Paragraph 2-14).

2-19. ALGOL EXEC CALLS

2-20. To access the DVR 61 driver in ALGOL, the user must declare EXEC references having five (Read/Write EXEC calls) and two (for Control EXEC calls) integer parameters. The required EXEC reference is declared using a CODE procedure. The sequence for declaring the EXEC reference is:

a. First, the EXEC reference having the required number of integer parameters is defined with an externally compiled PROCEDURE having a five-character name other than EXEC, i.e., xxxxx.
b. Next, xxxxx is declared in the main program as a CODE PROCEDURE.
c. Finally, the xxxxx reference can be made in the main program by name. In the call, the proper (five or two) number and type of parameters must be included. An example of the external definition of a five parameter Read/Write EXEC call is:

```
HPAL, P, "EXECM"
PROCEDURE EXECM (A, B, C, D, E); INTEGER A, B, C, D, E;
  PROCEDURE EXEC (V, W, X, Y, Z); INTEGER V, W, X, Y, Z;
  CODE: EXEC (A, B, C, D, E);
END
```

The main program PROCEDURE declaration for EXECM, then, could be written as:

```
(main program)
        :
PROCEDURE EXECM (TRANSFER TYPE,
LOGICAL UNIT NO, FIRST WORD OF BUFFER,
BUFFER LENGTH, FUNCTION);
```

```
INTEGER TRANSFER TYPE, LOGICAL UNIT NO,
FIRST WORD OF BUFFER, BUFFER LENGTH,
FUNCTION; CODE;
        .
        .
        .
```

Having compiled the necessary five-integer parameter EXEC and declared it in the main program, EXECM can now be called by the main program in order to transfer data to/from the Multiprogrammer System via the DVR 61 driver. A typical call, then, using the name of the EXEC reference in this PROCEDURE declaration is:

```
(main program)
        .
        .
        .
EXECM (ICODE, ICNWD, IBUFR [L], IBUFL, IFUNC);
        .
        .
        .
```

The parameters of the EXECM statement are integer variables and define the type of request, logical unit number, buffer address, and buffer length as previously described for the HP Assembly language Read/Write EXEC call (Paragraph 2-11). Note that the parameter must be defined before the EXECM procedure statement (and hence, call to DVR 61) is issued. For instance, if ICODE ← 1 and IFUNC ← 0 before the above PROCEDURE statement was given, the EXECM would call the driver to process a Normal Read request. Two parameter I/O Control EXEC calls can be externally defined and called in a similar manner.

2-21. DATA FORMAT

2-22 The Multiprogrammer System accepts 16-bit control, data, and address words (via the computer interface card) and returns a 12-bit (data bits 0-11, and control bit 15) input data word (see Figure 2-1). Control words are defined by the 1111 (17_8) address configuration in bits 15-12 and contain system mode assignment specifications for the Multiprogrammer System as well as multiprogrammer unit address information (bits 3-0). If bits 15-12, then, are not 17_8, the computer word to the multiprogrammer will be interpreted either as a data word (for output cards) or an address word (for input cards) depending upon the mode of operation established by the previous control word (and, particularly, the last programmed state of the ISL bit).

2-23. CONTROL WORD

2-24. The control word (address bits 15-12 all ones) specifies the mode of operation that the multiprogrammer is to assume including System Enable (SYE), Data Transfer Enable (DTE), Timing Mode Enable (TME), Input Select (ISL) and Interrupt Enable (IEN). In addition, the control word specifies which of the up to 16 multiprogrammer units is to subsequently receive output data words of input address words from the computer. Note that control word specifications (i.e., the mode commands as well as unit address) are in effect until changed by another control word. Further, while the unit address enables 1-of-16 multiprogrammers in the system, the mode control commands apply to all multiprogrammer units (and their

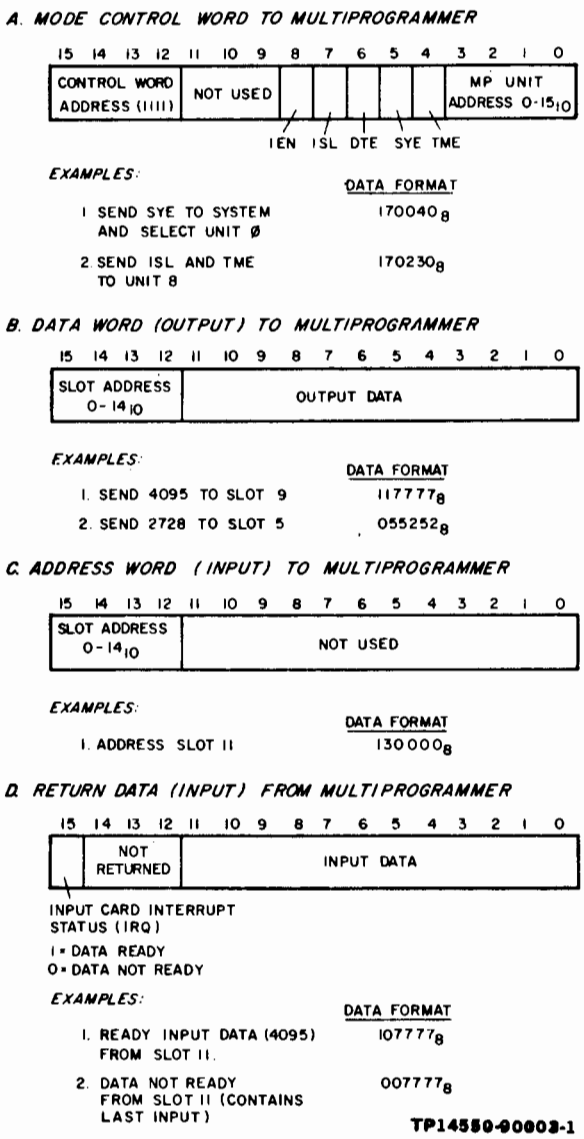


Figure 2-1. Multiprogrammer Data Formats

I/O cards) in the system regardless of which unit is currently addressed. The words are described as follows:

2-25. Output Modes

a. System Enable-SYE (40_8) - Enables or disables the outputs of all output cards in the entire Multiprogrammer System, regardless of the data on each card. SYE is off at power turn-on.

b. Timing Mode Enable-TME (20_8) - Disables (TME=0) or enables (TME=1) the multiprogrammer timed mode of operation. In the timed (waiting) mode, the Multiprogrammer System waits for all output card(s) to signal ready before it flags the computer that its operations are complete. When TME is not specified, the multiprogrammer automatically responds within 50 μsec with its Handshake Flag to notify the computer that it has accepted data input.

c. Data Transfer Enable-DTE(100_8) - Provides two functions for output cards: (1) enables output cards having dual-rank storage (high-speed cards) to immediately transfer data from the first register to the second and (2) enables the Gate output of cards that employ Gate/Flag circuits (these cards, i.e., the relay output card, have single rank storage). For the dual-rank storage cards, if DTE is not specified, data transferred to the card does not appear at the output (not transferred to the second storage register). When DTE is enabled, these cards appear to have single-level storage such that data appears at the output immediately upon receipt by the card. For output cards that generate a Gate to their external device, if DTE is not specified, the Gate output of the card is disabled. When DTE is enabled, the Gate circuit of the card is enabled.

NOTE

For output cards that produce a Gate to their external device, if DTE is off, the card's Gate circuit is disabled. Consequently, if these cards are addressed with TME on, DTE should also be on. If DTE is off, the card will not return a Flag to the multiprogrammer and the multiprogrammer will not, in turn, return a Flag to the computer. Thus, the computer could be in a "hang up" state waiting in the TME mode for the multiprogrammer to return a Timing Flag. Since the program that called the multiprogrammer is in I/O suspension until the request is completed, it too is "hung up" waiting for a TME Flag that will not be generated. Avoid this situation.

2-26. Input Modes

a. Timing Mode Enable-TME(20_8) - As for output modes, TME disables or enables the multiprogrammer timed mode of operation. When TME is disabled (0), the multiprogrammer returns its Handshake Flag after approximately 50 μsec to signal the computer that it can input new data or process an address word. When TME is enabled (1), the Multiprogrammer System (depending on the input mode established) either (a) waits for a currently-addressed (and previously-activated) input card to signal it has

data ready before generating the multiprogrammer Flag to the computer; or (b) waits for an input card (the first) from a group of previously addressed (and activated) cards to signal it is ready before it Flags the computer that new data is available. The first input mode of operation is the "dedicated" input mode and requires that the computer interface card continually address a particular multiprogrammer input card from which data is to be received (the read normal and both read direct requests use the dedicated input mode). The second input mode of operation is the "search" mode and allows a group of input cards to interface with the computer without requiring the interface card to maintain the address of any particular input card (both poll requests use the search input mode).

b. Input Select-ISL(200_8) - When programmed on (ISL=1), ISL accomplishes two basic functions. One, ISL turns the multiprogrammer System into an imput data multiplexer allowing data (bits 0-11, and bit 15) from input cards to be transferred to the computer. Input data is transferred to the computer when a card is addressed with or without a multiprogrammer Gate. Secondly, ISL also allows the activation of input cards. Input card activation requires that the card be addressed and amultiprogrammer Gate be issued after the ISL mode has been programmed. Upon activation, the input card begins a data transfer cycle with its associated device. Further, a card must be activated to signal it has data ready when the multiprogrammer is in the search input mode. When ISL is off (0), then, the multiprogrammer's input data bus is disabled and no input card data can be transferred to the computer. In addition, ISL must be off to de-activate input cards (i.e., by addressing an input card with a multiprogrammer Gate when ISL=0, the card will be prevented from signalling when it has data ready). Notice that if an optional jumper (W6) is installed on the applicable input card(s), the card(s) can be activated in a group by programming the interrupt enable (IEN) mode.

c. Interrupt Enable-IEN (400_8) - When programmed on (IEN=1), IEN allows an input card to signal that data is ready without regard to whether or not its address is currently stored in the computer interface card. Of course, since an input card is expected to signal data ready, the TME mode should also be programmed on to allow the input cards to control the multiprogrammer's Flag to the computer. Notice, too, that any number of input cards could be active and ready to input data when IEN is programmed, On the other hand, they could all be busy (timing Out) when IEN is programmed so that the first one that becomes ready will signal via the multiprogrammer Flag that data is available. The program must, of course, determine which card(s) became ready (by examining bit 15 of the returned card data). As mentioned previously, IEN can also be used (if optional jumper W6 is installed) to activate input cards providing they are not already active or timing out when IEN is programmed. When IEN is programmed off, the only way an input card can signal it has data ready is if its address is stored in the computer interface card (i.e., during the dedicated input mode).

2-27. DATA WORDS

2-28. Data words transmitted to the Multiprogrammer System control user devices through associated multiprogrammer output cards installed in the multiprogrammer units' I/O slots. The data words contain the address (Bit 15-12) of the I/O slot that is to receive the data portion (bits 0-11) of the data word. The I/O slot

addressed, of course, must be a slot in the multiprogrammer unit previously-addressed in a control word. Further, the data output operation must be preceded by the appropriate output mode control commands in the last programmed control word.

2-29. ADDRESS WORDS

2-30. Address words transmitted to the Multiprogrammer System control user devices through assoicated input cards installed in the multiprogrammer units' I/O slots. The address words contain only the address (bits 15-12) of the I/O slot that is to transmit its data to the computer (bits 0-11) are not applicable. As for output operations, the I/O slot addressed must be a slot in the multiprogrammer unit addressed in the last control word that also contained the appropriate input mode control commands. Address words are used not only to read input card data but, also, to start the card's timing cycle with its associated device (i.e., activate the card). Also, address words are used to deactivate input cards (turn-off their interrupt circuits). The input data returned to the computer interface card as a result of the address word (assuming the Multiprogrammer System is in the input mode) consists of 11 data bits (bits 0-11) and an input card interrupt identification status bit (bit 15) which can be examined by the user's program to determine if the addressed input card had made an interrupt request to the computer; i.e., if the card has data ready for computer input. Notice that bits 14-12 are not returned by the multiprogrammer to the computer interface card.

2-31. I/O READ/WRITE REQUESTS

2-32. READ REQUESTS

2-33. Eight READ requests are accepted and processed by the driver all of which require certain buffer formats. The requests allow a variety of functions to be executed including read-after-write from multiprogrammer I/O slots with or without the computer interrupt system; input polling; and interaction with the multiprogrammer operator. The particular read request selected will depend on the function to be executed and on system timing considerations since read requests are provided to execute with the multiprogrammer normally either in the timing mode or in the fast mode (hance the difference between using or not using the computer interrupt system). Two of the read requests, the Event Sense Poll to First Input and the Event Sense Poll All, are provided to facilitate the user of the driver's sense mode. The following paragraphs describe each of the read requests including buffer format and other requirements (i.e., hardware, timing) to be evaluated in considering the different read requests.

2-34. NORMAL READ

2-35. The normal read request writes a specified number of output words to the multiprogrammer system after which it reads a specified number of input words from a multiprogrammer input slot. All words are transferred using the computer interrupt system. Figure 2-2 illustrates the required buffer structure for the normal read request which is identical for the read direct requests. The first buffer word (the address of which is given in the EXEC call) specifies the total number of output words that are to be transmitted to the multiprogrammer system and must be a positive non-zero integer. The output words can be any combination of multiprogrammer control, data (for output card slots), or address (for input

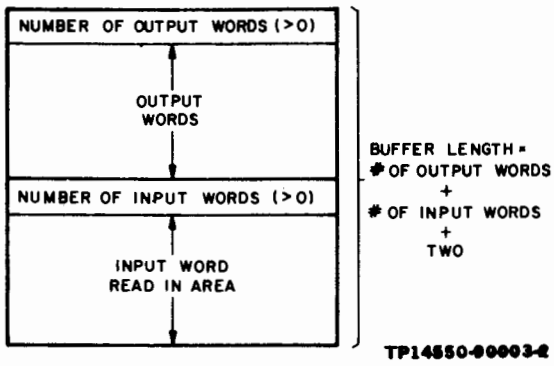


Figure 2-2. Normal Read and Read Direct Buffer Format

card slots) words. However, since the driver reads input words immediately following the transmission of the output list, the last words in the output list should enable input data reception from the desired multiprogrammer input slot. That is, the last control word in the output list should set the multiprogrammer to the Input Select Mode (ISL = 1) and enable the multiprogrammer unit that contains the input card from which data is to be read. The last output word, further, should be an address word that selects (and activates) the desired input card slot. The input card address is stored by the interface card so that the associated input card is continually enabled throughout the READ portion of the request.

2-36. The number of input words to be read (which also must be a positve non-zero integer) is specified following the output word list. Following this word, then, the buffer must include a read-in area that is as large as the specified input word total. The entire buffer length (specified in the EXEC call), therefore, must equal the output word list plus the input word read-in area plus two (the two words that specify the number of output and input words involved in the request). The driver checks the output and input word count specifications (they must be > 0) as well as the buffer length and rejects the request if any of these specifications are not correct.

2-37. Assuming the buffer is correctly structured, the driver transmits the output list in sequence after which it reads the input words and stores them in sequential locations in the buffer read-in area. Since all input words except the last are read in response to a preceding multiprogrammer Gate, the input card is activated each time a word is read. Further, after each data transfer (output or input), the driver returns control to RTE, with the next data transfer executed (by the driver completion section) when RTE gives service to the driver in response to the multiprogrammer interrupt via the computer interrupt system. Thus, if the multiprogrammer is operated in the fast mode (TME = 0), its Flag will request service in approximately 50 µsec after the last data transfer. Since RTE performs certain control processing when a driver returns control to it, in the fast mode the multiprogrammer will request service almost (if not) immediately after RTE turns the interrupt system back on when finished with its conrol processing. Thus, the driver may tie up all lower priority programs until all words have been transferred. To avoid this, the multiprogrammer can be operated in the

Timing Mode (TME = 1) during normal read and write requests, or the multiprogrammer channel and the user programs that call the DVR 61 driver be given sufficiently low priority to give other devices an opportunity of gaining computer service before the multiprogrammer. After it completes the request (i.e., the continuator is given service in response to the multiprogrammer's Flag and determines that all specified words have been read), the continuator checks if the sense mode is required (EQT5(0)=1) and if it is and no alarms are pending (EQT5 (1)=0), programs the interrupt enable mode (IEN, ISL, SYE, TME) to allow activated event sense cards to interrupt the computer if an input line changes.

2-38. Read Direct Requests

2-39. The read direct requests are similar to the normal read request in that both of these requests also WRITE (output) a specified number of words after which they both READ (input) a specified number of words. Further, like the normal read request, the output word list can be any combination of multiprogrammer control, data, or address words except that the last control word must select the appropriate multiprogrammer unit and input mode while the last output word must address the input slot from which data is to be read. In addition, as discussed for the normal read request, the driver rejects the request if the word count (output and input) specifications are not >0 or the buffer length is not corrected.

2-40. Unlike the normal read request, however, the read direct requests are executed entirely by the driver initiation section and require that the multiprogrammer be programmed to the fast mode of operation (TME = 0). The driver does not complete the request until the last input word is read. After each word is output to the multiprogrammer, the driver checks that the multiprogrammer Handshake (fast) Flag is returned within approximately 50 μsec and terminates the request if it is not.

2-41. The two read direct requests differ only in that for the read direct with Gate on input, the driver sets the multiprogrammer Gate(via the interface card) before receiving each input word. The multiprogrammer Gate activates the input card and causes it to generate its Gate to the external device. Since the multiprogrammer Gate is set, the driver checks the timing of the multiprogrammer Handshake Flag during read cycles as it does during output cycles. The input card Gate to the device can be used to transfer data into the card for those devices that require a transfer signal. For the read direct with not Gate, on the other hand, the driver inputs each without setting the multiprogrammer Gate. Thus, the input card is not activated so that the external device completely controls the transfer of data to the input card.

2-42. Some final considerations. In the Handshake mode, input data is read without regard to the ready status of the input card or external device. Further, as shipped from the factory, the digital input card requires an external device Flag to store input data. The device Flag, then, must meet the Flag timing requirements of the input card within the multiprogrammer Handshake Flag timing period. If the timing requirement is too fast, the device Flag storage requirement of the input card can be disabled in which case the card's storage circuits continually reflect the external input data lines. Thus, to avoid the possibility of using redundant data (either because the card is read twice before the external Flag strobes new data into the card or if the Flag strobe is disabled, the card is read twice before the input lines change with new data), the user's pro-

gram can be given the task of determining if new input data was received from the input card in each location of the read-in area. Of course, if the external device can meet the read cycle requirements (with or without the device Flag storage storbe), no program processing is required. Finally, as for the normal Read (and, in fact, all requests), at the completion of the Read Direct requests, the driver checks if the sense mode is required (EQT5(0)= 1 and EQT5(1)= 0) and, if it is, programs the channel to the interrupt enable mode. The driver then returns to RTE.

2-43. Read Operator Data

2-44. The read operator data request allows the user's program to interact with the multiprogrammer operator. The request requires only a two-word buffer (see Figure 2-3). When the driver receives this request, it transmits the output word to the multiprogrammer where it is displayed in the switch register to alert the operator that attention is required. The output word should be such that it alerts the operator without adversely affecting system operation (the word is processed by the multiprogrammer). After transmitting the output word, the driver checks that the multiprogrammer Handshake Flag is returned within approximately 50μsec thus implying that the output word is a control word (or the multiprogrammer was previously set to the fast mode) since the Handshake Flag is automatically generated when the multiprogrammer receives a control word (with IEN = 0). Next, the driver completion section is set to expect one input word and the driver returns control to RTE. When the operator responds to the programmed alert, he inputs a data word by first switching the multiprogrammer to local, entering his response code into the switch register, and depressing the RETURN DATA switch to actually send the data. The operator next places the multiprogrammer back to the REMOTE mode of operation.

2-45. When the RETURN DATA switch is depressed, it generates the multiprogrammer Flag to the computer interface card and, thus, generates a computer interrupt request. When the interrupt request is processed by RTE (by the CIC module) and the driver is given service, the completion secion reads the input word into the read-in area of the buffer and then terminates the request, returning control back to RTE. The user's program can now respond to the operator input. Note that since manual operations are involved in the request, the multiprogrammer channel is unavailable for other operations until the operator has input his response word and returned the multiprogrammer to the REMOTE mode of operation.

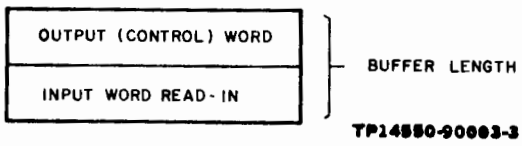


Figure 2-3. Read Operator Data Buffer Format

2-46. Poll Requests

2-47. There are four poll requests all of which are processed completely in the driver initiation section. The requests are used to read data from one or more previously-activated input card(s). The poll to first input requests sequentially search through a specified list of previously-activated input cards and determine which one, if any, has data ready for input to the program. The search is terminated when the first input card having data ready is detected, or if none are ready, after all specified cards are polled. The poll all requests sequentially read all input cards in the specified listand do not consider whether or not any card in the list has data ready. The difference between the Event Sense and normal Poll requests is the fact that the driver, for the event sense poll requests, automatically updates and rearms input cards that it finds had interrupted. That is, for the Event Sense Poll to First Input, if the driver detects an input card with data ready, it turns ISL off and outputs the same data that it read back to the card, programs the ISL mode on, and then addresses the card with a computer Gate to rearm it. Thus, if the card is an event sense card it will be automatically updated with a new reference word and rearmed. The same update and rearm procedure is implemented for the Event Sense Poll All in which case all ready input cards are updated and rearmed. Note that while the Event Sense Polls have been provided primarily for event sense cards and sense mode of operation, they can be used to rearm digital input cards also (sending a digital input card a reference word will have no affect on the card). The buffer format requirements for the poll requests are similar as shown in Figures 2-4 and 2-5. Note that for all of the poll requests, after the requests are completed, the driver checks if the sense mode bit is on and the alarm pending bit is off and, if they are, programs the multiprogrammer to the interrupt enable mode before returning to RTE.

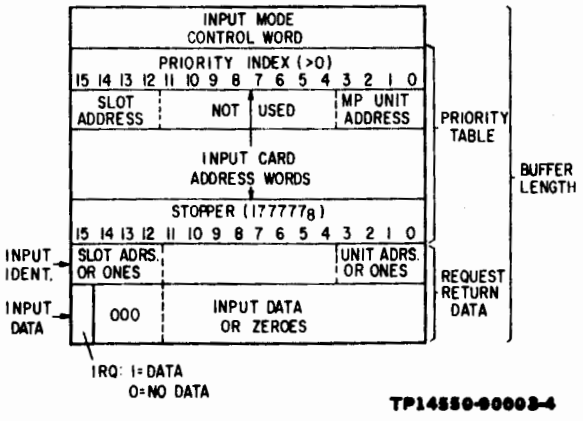


Figure 2-4. Poll to First Input or Event Sense Poll to First Input Buffer Format

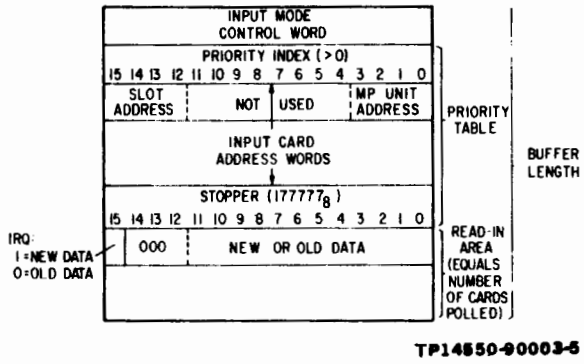


Figure 2-5. Poll All or Event Sense Poll All Buffer Format

2-48. As shown, the first buffer word for all poll requests must be an input mode control word that is transmitted by the driver to the multiprogrammer to set it to the input select mode. In addition, the driver adds the appropriate unit number (contained in each input card address word) to the control word and transmits it to the multiprogrammer before it transmits it to the multiprogrammer before it transmits the first address word and, thereafter, everytime it determines that the next address word to be transmitted contains a unit number that is different from the previously-transmitted unit number, the driver uses this first word. The next buffer word, the index word, indentifies the location in the priority schedule table at which the input card read sequence is to begin. Thus, a single priority schedule table can be used to control different poll sequences by changing the point at which a sequence begins. Of course, regardless of where the poll sequence is started, the order in which input cards care polled (read) is defined in the priority table in that all input cards from the indexed card to the end of the table (which is identified by the "stopper" word) are read in the sequence listed in the table.

2-49. The next area of the buffer is the list of input cards that are to be polled. Each location from the index to the stopper contains an input card address word that identifies the desired input card slot address (bits 15-12) and the multiprogrammer unit address (bits 3-0) in which the card is located. The unit address is examined by the driver to determine if the control word must be transmitted first in order to enable the associated multiprogrammer unit (and, of course, the input mode). Assuming the multiprogrammer unit is enabled, the driver then outputs the address word after which it reads in the input card's data (from, of course, the interface card). The input address words are transmitted without setting the multiprogrammer Gate which means that the input cards are not reactivated as a result of the poll requests. In fact, for the normal Polls (14 and 15), the input cards are not reactivated at all. The input data received from the addressed input card is processed according to the type of poll request issued.

2-50. For the poll all request, input data is stored in the buffer read-in area in the order that input cards are addressed. That is, data from the first input

card addressed (defined by the priority index word) is stored in the first read-in location with subsequent input data words stored in succeeding locations. Note that the state of bit 15 (the IRQ bit) of each input word is used during the Event Sense Poll All to determine if the associated data bits (bits 0-11) represent new data (bit 15 = 1) or old data (bit 15 = 0) from the card. If IRQ= 1, the data bits are transmitted back to the card with ISL off in order to update event sense cards with new reference words. The card(s) is also reactivated (addressed with a gate in the ISL mode). The input cards are read in the order they are listed in the priority table until the driver detects the stopper word (177777_8) at which time it terminates the poll all request, programs the interrupt enable mode as required, and returns control to RTE.

2-51. For the poll to first input request, the input data received from an addressed input card is examined by the driver upon receipt. If IRQ bit 15 = 1, the input card had signalled it was ready to send new data so that the driver stores the card's priority table word (which includes its slot and unit address) in the input identification location and then stores the input data in the associated data location. For the Event Sense Poll to First Input, the ready card is next updated with a new reference word and then reactivated. At this point, the driver checks if the interrupt enable mode is needed and then returns control to RTE. Thus, input cards are read until the first card that has new data is detected. If no cards are found to have data ready, the driver, upon detecting the priority table stopper word, stores the stopper in the identification location and then clears (zeros) the input data location. Thus, the user program can determine if the requests successfully received new data by examining the state of bit 15 of the input data location. If bit 15 = 1, the associated data bits (0-11) represent valid input data received from an input card the address of which is stored in bits 15-12 (slot address) and 3-0 (unit address) of the identification location. Conversely, if bit 15 = 0, no cards were found with data ready.

2-52. Note that before executing the requests, the driver checks the structure of the buffer and rejects either request if the stopper word, which defines the end of the priority schedule table, is not found in the correct place in the buffer. Further, the poll all request is rejected if the number of cards addressed does not equal the number of locations in the reading area.

2-53. WRITE REQUESTS

2-54. Two Write requests, the normal write and write with Handshake Flag, are accepted and processed by the driver. Neither write request requires any special buffer format except at least one word must be in the output buffer (see Figure 2-6). The output buffer can be any combination of multiprogrammer control, data or address words which are transmitted by the driver to the multiprogrammer in the order listed.

2-55. Normal Write

2-56. The normal write request is processed essentially the same as the WRITE portion of the normal read request. The driver outputs a specified number of words through the completion section and the computer interrupt system. Thus, the multiprogrammer may be set to the Timing mode (TME = 1) for all data and address word outputs. Notice that if the multiprogrammer is set to the fast

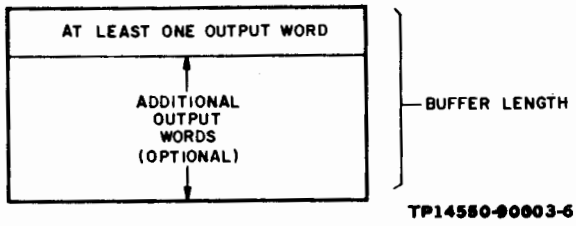


Figure 2-6. Normal Write and Write with Handshake Flag Buffer Format

(Handshake) mode (TME = 0), it is possible that the driver will tie-up the computer (run virtually uninterrupted) until all words have been transmitted as previously described for the normal read request. To avoid tying up the system in the fast mode, therefore, the multiprogrammer and the user calling programs should be assigned sufficiently low priorities in the system. After the continuator outputs all requested words, it programs the interrupt enable mode if required and then returns to RTE.

2-57. Write With Handshake Flag

2-58. The write with Handshake Flag is a faster version of the normal write. The driver transmits each word to the multiprogrammer expecting it to return the Handshake Flag in approximately 50 μsec. If it doesn't, the driver terminates the request. Since the Handshake Flag is expected, the multiprogrammer should be set to the Handshake mode (TME = 0) for all data and address word outputs. The request is processed entirely by the initiation section with the driver returning control to RTE only after all words have been written and the interrupt enable mode programmed if required.

2-59. I/O CONTROL REQUESTS

2-60. As mentioned previously, the driver processes three I/O control requests which are provided for the sense mode of operation. All of the requests control internal sense mode Flags kept by the driver for the multiprogrammer channel in its EQT (i.e., EQT5). Thus, no data is transferred for these requests and they are executed entirely in the initiator which performs the desired function, checks if the sense mode is required and, if it is, programs the multiprogrammer to the interrupt enable mode, and then returns to RTE. Each of the control requests are described in the following paragraphs.

2-61. SET SENSE MODE

2-62. For this I/O Control request, the driver first checks to see if an interrupt processing program has been supplied by the user. (As previously mentioned, if a program was supplied, the first time the driver is called, the initiator stores its I.D. segment address, a positive number, in EQT13; if a program was not supplied, the initiator sets EQT13 = -1.) That is, for this request, the driver first checks EQT13 and if it is > 0, sets the channel's internal "sense mode required" flag (EQT5, bit Ø = 1). In addition, the driver checks if the alarm pending bit is off and if it is, programs the 6940A to the interrupt enable mode by sending it a control word with IEN, ISL, TME, and SYE

(for active output cards) on. The driver next returns to RTE (A-Reg = 4). If EQT13 < Ø when the driver executes this request, it rejects the request (i.e., the user did not supply an interrupt processing program at RTGEN time. See Paragraph 2-74).

2-63. After the sense mode bit has been set, the driver will always leave the 6940A channel in the interrupt enable mode after it completes any of the ten I/O Read/Write EXEC calls. Thus, whenever a normal I/O transfer is not inprocess, the 6940A channel is in the interrupt enable mode and its event sense card(s) may be enabled. If one or more input lines to an enabled event card change state, an event sense interrupt will be processed by the driver's continuator as described in Paragraph 2-72.

2-64. CLEAR SENSE MODE

2-65. For this request, the driver clears the channel's sense mode flag [EQT5(Ø) = Ø] and then returns to RTE (A-Reg = 4). With the sense mode flag cleared, the driver will not program the 6940A to the interrupt enable mode when normal I/O requests are not in process. The 6940A channel, then, will not operate in the sense mode but will process any of the ten normal I/O requests.

2-66. CLEAR INTERRUPT PROCESSING REQUIRED FLAG

2-67. Whenever the continuator is called for a sense mode interrupt [i.e., an interrupt occurred, the sense mode flag is on, EQT5(Ø) = 1, and the driver was not called for normal I/O, EQT1 = 0, or a time out, EQT4 (11) = 0], the driver sets an interrupt-processing-required flag [EQT5 (1) = 1]. Also known as the "alarm pending bit" (APB), EQT5 (1) is the means by which a user can determine if the sense mode interrupt processing program is not dormant. That is, the driver holds the APB on whenever an interrupt has occurred and the driver found the Alarm Processing Program to be not dormant (i.e., couldn't schedule the program for the interrupt). Whenever the driver actually schedules the program (if the interrupt was due to another event or if RTE called the driver for a time out), it resets the APB. The user can determine the status of the APB by issuing an I/O Status EXEC call.

2-68. SENSE MODE OF OPERATION

2-69. The Sense Mode of HP 6940A operation is intended to allow continuous monitoring of, and rapid service for, "critical" alarm-type event inputs. In the Sense Mode, multiprogrammer HP 6943A card(s) can continually monitor input lines whenever a normal I/O request is not in process. If the Sense Mode has been programmed on and any of the input lines change state, the driver's continuation section is entered through the interrupt system and it:

1. Reads (polls) the input lines in unit ØØ twice.
2. Schedules a user-supplied interrupt processing program.
3. Passes the interrupt data it collected to the interrupt processing program's ID segment. The processing program can access the data with a call to RMPAR.
4. Continually initializes the channel's time out value (EQT14) and clock (EQT15) whenever a sense mode interrupt occurs and the interrupt processing program is not dormant (has been previously scheduled by the driver). This assures that the interrupt processing program will be eventually scheduled again even if it is not found dormant by the driver at the time of a subsequent alarm (or at 50 msec intervals thereafter).

The driver also sets an alarm pending bit [EQT5 (1)] that can be read with an I/O Status EXEC call. Note that if the driver did not process time outs, RTE would set the 6940A down when a time out occurred.

2-70. EQT1 is the mechanism that the continuation section uses to determine if interrupts are for normal I/O or for the sense mode. If EQT1 is Ø when the continuator is entered, an event change must have caused the interrupt since RTE sets EQT1 to Ø if the driver is not busy. If normal I/O is in process, EQT1 ≠ Ø in which case the driver continuator handles interrupts as it did before this revision.

2-71. To summarize, the Sense Mode:

a. Requires a user-supplied event sense interrupt processing program that is incorporated into the RTE system along with DVR61 at system generation time. The processing program should have sufficiently high priority so that evwnt changes are expeditiously processed. See Section III for information on system generation requirements.

b. Requires that the driver process time outs. The first time the driver is called for any request, it notifies RTE that it will process time outs (sets EQT4, bit 12). Since the driver is processing time outs, it will continually use its time out clock (EQT15) such that RTE will call the continuator every 50 msec (the time out period) until the interrupt processing program finally is found dormant whereupon the driver schedules it. If RTE handles time out, it sets the channel down if it detects a time out.

c. Requires that the user program the sense mode on via the Set Sense Mode I/O Control EXEC call (Paragraph 2-61).

d. If the sense mode has been programmed on for a 6940A channel, the driver will automatically program the 6940A to the interrupt enable mode whenever normal I/O is not in process. If a normal I/O request is directed to the 6940A, the driver executes it to completion after which it again leaves the 6940A in the interrupt enable mode before returning to RTE. Of course, a user can turn the mode off at any time with the Clear Sense Mode I/O Control EXEC call (Paragraph 2-64).

2-72. DVR61 CONTINUATOR OPERATION

2-73. The continuator section has the responsibility of determining if an interrupt is due to a normal I/O transfer or a sense mode interrupt or a system time out. After determining the reason for the interrupt (if it can), the continuator performs the required processing depending upon the type of interrupt. In general, the continuator can be entered for three valid reasons as follows:

1. A normal I/O read or write transfer is in process and the multiprogrammer is ready to transfer another word, or
2. an event change occurs and the multiprogrammer is in the sense mode, or
3. RTE times out a 6940A channel after an event interrupt occurred when the interrupt processing program was already scheduled by the continuator (in response to a previous event sense interrupt).

2-74. It is possible that the three valid continuator entry situations may overlap (i.e., RTE may time out a channel during a normal I/O transfer) in which case the continuator will ignore the interrupt and let the system recall

it with the conflict resolved. It is also possible, by the way, that the continuator will be called during circumstances that are not valid (i.e., the system times out the channel but the alarm pending bit is off). These entries are rejected by the continuator which also notifies the user (via EQT5, bit 2 which can be accessed by a Status call) of the error.

2-75. Figure 2-14 is an overview of the continuator's processing scheme. Note that all of the processing done by the driver is not illustrated. For instance, when it finishes a normal read/write, the driver reports the transmission log (number of words input and/or output) to RTE in EQT10. Figure 2-14 does, however, illustrate several important aspects of the driver's operation. For example, it can be seen from the flowchart that once the sense mode bit has been set [EQT5 (Ø) = 1], the driver will always program the 6940A to the interrupt enable mode when normal I/O is completed. Then, if an interrupt occurs and assuming it is not due to a time out, the continuator first does a Poll to First Input on unit ØØ [the driver assumes the event sense card(s) is/are in the master 6940A] in order to "catch" the event before it has a chance to reset. Next, assuming normal I/O is not in process, the driver sets the APB bit and then schedules the interrupt processing program. After scheduling the program, the continuator does another poll on unit ØØ (to get any other event changes that may have come in) and then puts the results of its two polls in the temporary area of the ID segment of the processing program as follows:

ID Segment Word	
2	Slot number of interrupting card (first poll). Set to 17_8 if no cards in unit ØØ interrupted.
3	Data from that card (bits 11-ØØ only).
4	Slot number of interrupting card (Second Poll). Set to 17_8 if no cards in unit ØØ interrupted.
5	Data from that card (bits 11-ØØ only)
6	Channel number of 6940A that interrupted.

2-76. The continuator next sets the address of the ID Segment temporary area in the ID Segment B-Register, clears the APB [EQT5 (1) = Ø] and exits. Note that the interrupt enable mode is not turned back on. Now it is up to the user's interrupt processing program to process the event data. First, it must retrieve the data passed from the driver by calling RMPAR. Next, it can do virtually anything it wants in response to the interrupt. For example, the program can do an Event Sense Poll All request to the driver in which case the event sense card(s) that detected the change(s) will be updated [with new reference words that reset the event(s)] and rearmed and the interrupt enable mode turned back on.

NOTE

If an event interrupt occurs and the driver finds the alarm processing program not dormant, it stores the results of its polls in available EQT locations. The driver passes the results of the polls to the program's ID Segment when it is able to schedule the program (i.e., it is dormant).

2-77. Notice another important processing path that the continuator takes. If an event interrupt occurs but the processing program is busy (the program has already been scheduled by a preceding event interrupt), the continuator will initialize its time out value and clock (EQT14 and 15) for a 50 msec time out period and return to RTE with the APB [EQT5 (1)] on. RTE will time out the driver and call it after 50 msec. The continuator will again attempt to schedule the processing program and, if it is still not dormant, again initialize the time out value and clock. The time out loop will be repeated until the driver can schedule the program (in which case it resets the time out value and clock). Thus, the continuator assures that the processing program will be scheduled for an event interrupt. Of course, whether or not the user's alarm processing program runs to completion depends on its priority (it should be assigned a high priority).

2-78. The user can look at the status of the APB with an I/O Status EXEC call. If the interrupt processing program has been scheduled but not run to completion for too long a time (i.e., the APB is on), the user can take an alternate path; i.e., he can clear the APB with the appropriate I/O control EXEC call and then notify the operator that a critical event has not been processed. Note that the time out value and clock are also cleared by the Clear APB EXEC call so that the continuator will not be called by RTE again due to a time out.

2-79. One final word about the continuator. If the continuator is called and cannot distinguish the exact reason, it will ignore the interrupt. For example, if a time out loop has occurred (the time out clocks were initialized and the APB set) and then a normal I/O transfer was requested, it is possible that the continuator will be called for a time out during the normal I/O transfer. Does the entry represent a Flag from the multiprogrammer for another word (normal I/O transfer) or the system calling the driver to try to schedule the interrupt processing program? The continuator assumes it was called for a time out but does not process it. Instead, it ignores the request and returns to RTE. If the interrupt was for normal I/O, the multiprogrammer's Flag will remain on and another interrupt will occur when RTE turns the interrupt system back on. This time, however, RTE will not be calling the driver for a time out since the time out period will not have expired again. If the original interrupt was a time out during normal I/O, the normal I/O request is allowed to complete before the time out is processed. Thus, normal I/O transfers should be done quickly, since they can delay the processing of event sense interrupts.

2-80. PROGRAMMING HINTS AND PITFALLS

2-81. RTE 6940A driver DVR61 has been designed to allow the user to do virtually any job he wants with a 6940A system, and do it in a multiprogramming environment. Because of its complexity and flexibility, however, there are some no-no's, boo-boo's and suggestions that should be observed to make life a little easier when multiprogramming the multiprogrammer.

2-82. First of all, make sure you give your alarm processing program(s) the highest possible priority. Assuming these programs are handling critical sense mode alarm interrupts, it is mandatory that they run quickly every time they are scheduled by the driver. Delay can be disastrous.

2-83. To further expedite the servicing of sense mode event interrupts, it is recommended that all normal I/O transfers be done as fast as possible. This is especially true of the interrupt ("normal" normal) read/write requests as they will suspend the sense mode (the multiprogrammer is not set to the interrupt

enable mode if the driver is processing normal I/O) or delay the scheduling of the event interrupt processing program if the sense mode was on in the past and an interrupt that is still waiting (in the time out loop) for service from your alarm processing program. Whenever possible, then, use "fast" I/O transfer requests (the Write with Handshake Flag, the Read Directs, and the Polls). If you have to use the interrupt system transfers (and you will, of course, if you need to know when an external device has responded to a programmed word; i.e., whenever you want to synchronize the program with the external device through the multiprogrammer which will probably be often if not always), then try to keep the buffers small so that the channel is not tied up too long.

2-84. Another aspect of the driver that should be thoroughly understood is that the driver automatically polls (twice) unit ØØ (always the 6940A) when a sense mode interrupt occurs. This automatic poll, then, assumes that the user's event sense cards are in the 6940A unit. If event sense cards are in any other unit(s), your alarm processing program has the responsibility of reading the cards and collecting interrupt data. Note that if an event sense card in a unit other than ØØ is responsible for a sense mode interrupt, the driver's polls on unit ØØ will not (of course) detect an interrupting card. The data passed to the alarm processing program, then, is not of much value. The driver indicates this fact (i.e., the slots in unit ØØ did not interrupt) by setting to 17_8 the slot identification information it passes to the processing program (ID segment words 2 and 4). There is, of course, no I/O slot 17_8 in the multiprogrammer. Note that the data passed to the processing program (ID segment words 3 and 5) is set to Ø.

NOTE

For all read/write requests that it completes in the initiator (i.e., the four polls, the read direct requests, and the write with Handshake Flag), the driver assumes that the TME mode is off. Consequently, each time it Gates the multiprogrammer, the driver checks that the Handshake Flag is returned within 50 µsec. If the Flag is late, the driver returns to RTE with a transmission error. If during the sense mode of operation a spurious Flag interrupts the driver, (the multiprogrammer is switched to LOCAL, for instance) the flag is processed as an "alarm" interrupt and multiprogrammer unit ØØ is polled. The Handshake Flag is checked during this Poll so that a Late Flag Error will be detected. Since the Flag Error occurred during the sense mode when normal I/O was not in process, the driver sets the slot address data in the ID Segment to 17_8 and the data in the ID Segment to 7777_8. Thus, if the slot address is 17_8, the user's alarm processing program can determine (from the data returned in the ID Segment) if the driver polled and found no cards reading in unit ØØ (data = ØØØØ) or had detected a Flag error (data equals 7777_8).

2-85. Another important consideration to remember when multiprogramming the multiprogrammer is a general requirement and is not related to whether or not the sense mode is used. That is, since the 6940A channel(s) can be controlled by multiple programs, any one program cannot expect the 6940A's control modes to be "remembered" between I/O transfers. If a program sets a particular mode for an I/O transfer and does another I/O transfer later, it cannot expect the multiprogrammer to be in the previously-programmed mode. Other programs have had access to the channel and may have programmed different modes. In fact, as has been mentioned, if the sense mode is used, the multiprogrammer is automatically set to the interrupt enable mode between normal I/O transfers.

2-86. One final suggestion. When the driver's continuator schedules your alarm processing program in response to a sense mode interrupt, the B-Register location of the program's ID segment (Word 11) contains the address of the ID Segment Temporary Area (Words 2-6). Thus, a FORTRAN processing program can make an immediate call to RMPAR to retrieve the interrupt data collected by the driver. For example:

```
PROG ALRM
DIM IP (5)
CALL RMPAR (IP)
```

would put the five data words from the ID Segment Temporary Area into array IP.

2-87. SAMPLE PROGRAMS

2-88. The following paragraphs provide examples of how to utilize driver DVR61 to accomplish various system functions. The sample programs are provided for illustrative purposes only and are not intended to be definitive of all ways in which the driver can be utilized. The driver is a very flexible I/O controller and is designed to satisfy a wide variety of system requirements. The examples are intended to suggest methods by which the user may implement his unique multiprogrammer system functions. For convenience, the examples all utilize FORTRAN EXEC calls to the driver. The applicable Assembly language statements can be substituted if desired. The first three examples illustrate normal I/O data transfer while the fourth example illustrates the sense mode of operation. It should be noted that the two basic modes of operation (normal I/O data transfer and sense mode of operation) are really separate in that the driver always suspends the sense mode (if programmed) when processing any I/O read/write or control request. Conversely, the driver will establish the sense mode (program the HP 6940A channel to the interrupt enable mode) whenever it is programmed on and I/O is not in process (and, of course, a previous alarm is not waiting for service). The sense mode, then, runs in the "cracks" between I/O requests and, thereby, allows the multiprogrammer to be fully occupied with I/O tasks at all times.

2-89. EXAMPLE 1 - SET OUTPUTS/READ INPUT(S) WHEN READY

2-90. The first sample program illustrates how to set a variety of devices through associated output cards, activate input feedback paths, and then leave the system returning only when one (or more, as desired) of the input devices is (are) ready to provide input(s). This kind of interaction suggests a test system in which reference values (voltage, resistance, etc.) are set and data collected when the test system indicates that valid data is ready. These functions can be implemented by combining the normal write request with either of the poll requests depending on user requirements.

Normal Write Request - initialize system; set desired output(s); activate inputs.

```
ICODE = 2       (write request type)
ICNWD = 100B    (binary mode + logical unit 8)
IBUFL = 13
IFUNC = 0       (normal read or write)
CALL EXEC (ICODE, ICNWD, IBUFR, IBUFL, IFUNC)
```

2-91. This EXEC call previously described transmits the 13-word IBUFR which, of course, must be previously created (using appropriate DIMENSION and DATA statements, for instance). For this example, then, assume IBUFR was configured and transmitted as follows:

IBUFR (1)	17ØØØØ	Control word to initialize system; select unit ØØ.
	Ø7ØØØØ	Deactivate input card in unit ØØ, slot 7
	16ØØØØ	Deactivate input card in unit ØØ, slot 14
	17Ø143	Control word with SYE and DTE on; select unit Ø3
	Ø7ØØØØ	Deactivate input card in unit Ø3, slot 7
	17Ø16Ø	Control word with SYE, DTE and TME on; select unit ØØ
	1Ø1111	Program output card in unit ØØ, slot 8 to 1111
	17Ø34Ø	Control word with ISL, SYE, and DTE on; select unit ØØ
	Ø7ØØØØ	Activate input card in unit ØØ, slot 7
	16ØØØØ	Activate input card in unit ØØ, slot $14_{1Ø}$
	17Ø343	Control word with ISL, SYE, and DTE on; select unit Ø3
IBUFR (13)	17Ø76Ø	Control word with IEN, ISL, SYE, DTE, and TME on; MP set to interrupt mode (output card still enabled with SYE on).

The timing mode is programmed [IBUFR (6)] to allow the output card to time out before activating input cards so that the output card's timing cycle will not interfere with input card timing. After the output card has timed out, then, the next phase of the output list is executed. The multiprogrammer is set to the input mode (ISL=1) with Handshake timing (TME=0) and the desired input cards activated by outputting the buffer address words (the input cards are activated because all words transmitted to the multiprogrammer by the normal write request are done so with an accompanying multiprogrammer Gate). Since TME is off, card activation is also done quickly (again, however, by way of the computer interrupt system). Upon activating the required input cards, the last word transmitted is a control word with IEN and TME both on. Thus, the interrupt input mode of operation is programmed with any one of the activated input cards allowed to generate an interrupt to the computer when it receives data from its associated device. It should be noted that the normal write request is not yet terminated and will not be until the first input card generates a computer interrupt. When it does, the driver's completion section is given service (through CIC, of course) whereupon it determines that all buffer words have been transmitted and, as a result, terminates the request. Upon termination, control is returned to RTE which subsequently returns control to the main program.

NOTE

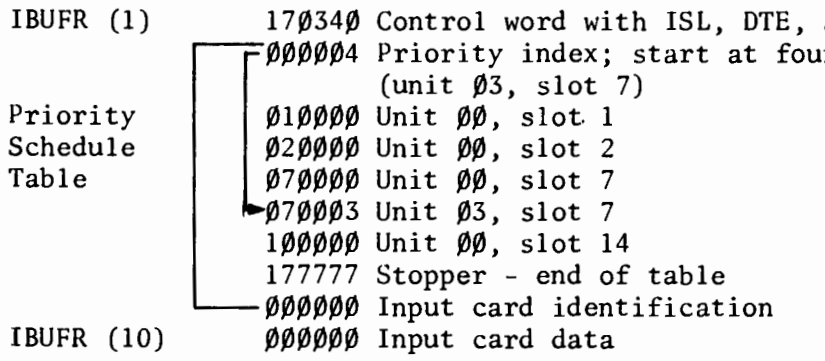
During the entire input sequence (from the time that ISL was first programmed on), the SYE function is programmed on. While SYE is not applicable for input cards, it is kept on to prevent the previously-programmed output card from being disabled. If SYE is turned off, the outputs of all output cards are disabled.

2-92. The main program has two options at this point. If input data is required as soon as it is ready, the poll to first input request can be programmed and the input data read. However, if it is either (1) not necessary to respond immediately to the input interrupt; i.e., input data can be accumulated from more than one input card and read at some later point in the program or if (2) a group of input cards might time-out simultaneously, then the poll all input request can be programmed.

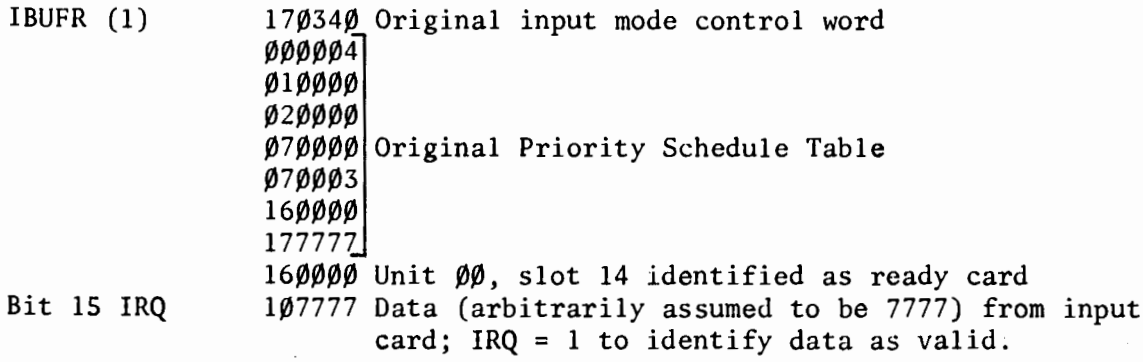
2-93. Assuming the main program requires data as received, the following processing and poll to first input request can be executed by the program:

```
ICODE = 1       (read request type)
ICNWD = 110B
IBUFL = 1Ø
IFUNC = 4       (poll to first input)
CALL EXEC (ICODE, ICNWD, IBUFR, IBUFL, IFUNC)
```

Before the EXEC call is issued, assume IBUFR was configured as follows:



Assuming that the unit ØØ, slot 14 input card had made the data-ready interrupt that ended the normal write request, IBUFR is configured as follows after the driver completes the poll to first EXEC call:

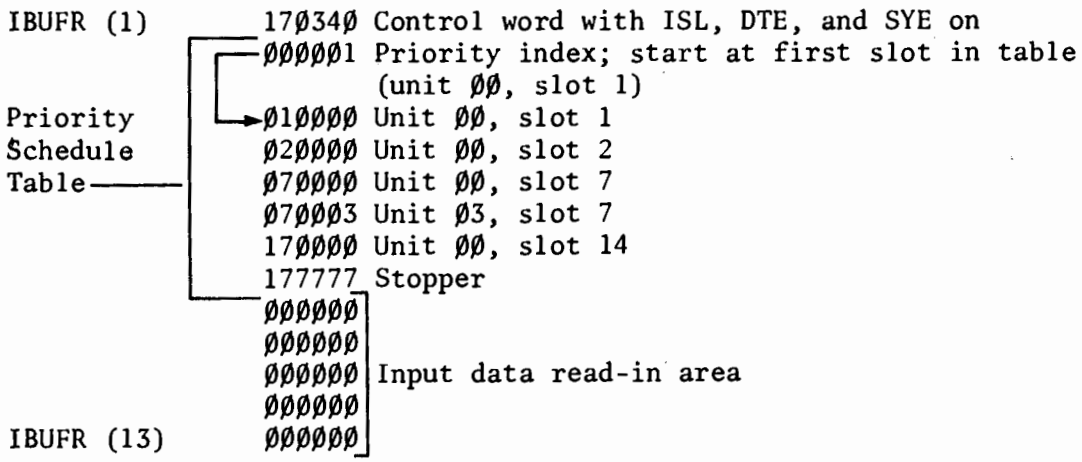


2-94. Thus, the poll-to-first-input returns the input card's data in the appropriate buffer read-in location. The card that was ready is also identified in the read-in area. Notice that, although the example shown is a relatively simple one, it does illustrate important features available at the option of the user. For one, the priority index points to the card that is expected to return data and thereby saves some time in the poll (search). However, since the poll is done with the multiprogrammer in the Handshake mode (TME = 0), the entire list could be executed rapidly. Notice, also, that the priority table includes cards (unit ØØ, slots 1 and 2) that were not even activated so that one priority table can be constructed for the system and utilized as required by changing the priority index. Some final points to remember: (1) all input cards are addressed without a multiprogrammer Gate so that if it is desired to reactivate the card that had inputted data, another request (a normal write, for example) must be programmed; and (2) the poll request does not leave the multiprogrammer in the interrupt input mode so that if it is desired to set the multiprogrammer back in this mode so as to repeat the input card interrupt/poll sequence for the next data input, the appropriate control word (IEN and TME on) must be issued in another request (again, another normal write could do this).

2-95. Assuming that input data from cards activated by the normal write is either not needed immediately by the main program or the cards are expected to time-out at the same time, the poll all request can be issued. The following processing and poll all request could be used to read in the data initiated by the previous normal write:

```
ICODE = 1       (read request type)
ICNWD = 110B
IBUFL = 13
IFUNC = 5       (poll all request)
CALL EXEC (ICODE, ICNWD, IBUFR, IBUFL, IFUNC).
```

The EXEC call polls buffer IBUFR which is similar to the buffer used in the poll to first input as follows:



After the driver completes the poll all call, IBUFR is configured as follows:

```
IBUFR (1)        17Ø34Ø Original input mode control word
                 ØØØØØ1
                 Ø1ØØØØ
                 Ø2ØØØØ
                 Ø7ØØØØ Original Priority Schedule Table
                 Ø7ØØØ3
                 16ØØØØ
                 177777
                 ØØØØØØ Unit ØØ, slot 1 data not ready
IRQ = 0 ———————  ØØ2525 Unit ØØ, slot 2 data not ready (card has
                        old data)
                 ØØØØØØ Unit ØØ, slot 7 data not ready
                 1Ø1111 Unit Ø3, slot 7 data ready
IRQ = 1 ———————  1Ø7777 Unit ØØ, slot 14 data ready
```

2-96. Several important aspects of the poll all request are illustrated in the above example. For one, after the request is finished, input card data is listed in the buffer read-in area in the order that the cards were polled. Since no identification as to which card's data is stored in a particular location is returned, the calling program must remember the order in which the cards were polled. Notice that the return input data has been arbitrarily chosen for the example and that the card in unit ØØ, slot 7 was assumed to have data received from a previous operation (the card was not activated during the previous normal write). Notice, that the unit Ø, slot 1 and 2 input cards were also not ready when the request was executed so that their IRQ bits are off. The other two input cards had ready data as indicated by the status (1) of their IRQ bits. Notice that input cards whose slot addresses require bit 15 to be on, do not return that bit on unless they actually have data (i.e., if card 14, unit ØØ had no data ready, IBUFR (13) would have been 00xxxxx). At the completion of the poll all request, then, the main program can examine bit 15 of each input card data location to determine if the corresponding card had sent new data (data received since the card was last activated). Finally, like the poll to first input request previously discussed, all input cards are read during the poll all without an accompanying multiprogrammer Gate so that no cards are re-activated.

2-97. EXAMPLE 2 - SET OUTPUTS AND/OR READ INPUTS IMMEDIATELY

2-98. The read direct requests can be used to set any number of output values after which any number of input words can be read from one input card. Of course, the requests can also be used without setting output values but only to read a block of words from a card. An example of the first application (set outputs/read inputs) is given in the following sample program which utilizes a read direct without Gate request. After the request is executed, the IRDIN area of IBUFR (which was previously created) contains the five words received from unit ØØ, slot 2. Notice that in specifying IBUFR, IRDIN must be EQUIVALENCED to IBUFR. For example: EQUIVALENCE [IRDIN (1), IBUFR (7)].

```
ICODE = 1       (read request type
ICNWD = 110B
IBUFL = 11
IFUNC = 2       (read direct without Gate)
CALL EXEC (ICODE, ICNWD, IBUFR, IBUFL, IFUNC)
```

```
IBUFR (1)       ØØØØØ4 Write count (four words)
                17Ø1ØØ Control word with DTE = 1; (TME = 0);
                       select unit ØØ
                1Ø7777 Set unit ØØ, slot 8 to 7777 (arbitrary value).
                17Ø24Ø Control word with SYE = 1; ISL = 1 (TME = 0);
                       select unit ØØ
                Ø2ØØØØ Address unit ØØ, slot 2 input card
                ØØØØØ5 Read count (five words)
IRDIN (1)       ØØØØØØ
                ØØØØØØ
                ØØØØØØ Read-in area (five word locations)
                ØØØØØØ
                ØØØØØØ
```

2-99. Several important aspects of the read direct requests are illustrated in the sample program. For instance, notice that all data transfers (output and input) are executed with the multiprogrammer set for Handshake Timing (TME = 0). This is an absolute requirement in that the driver checks the timing of the multiprogrammer Handshake Flag for all output words transmitted and, if the read direct with Gate request is used, for all input words received as well. Since Handshake Timing is employed, then, the Flag timing circuits of the output cards are ignored for the length of the request. However, if TME is subsequently programmed on, the previously programmed output cards may affect the multiprogrammer Flag as they have not been allowed to time out. Thus, if it is not desired to allow these cards to generate a Flag after the read direct is finished, make certain that all the cards have timed out before programming TME on. Of course, it may be desired to let the cards generate a Flag, in which case TME can be programmed on in a subsequent request (either of the write requests could be used to do this).

2-100. Another important consideration in using the read direct requests is that both are intended to input data from high speed devices (the setting of the output card by the sample program, IBUFR words 2 and 3, could have been omitted). Little or no data transfer timing control is established between the input card and its associated device. For the read direct without Gate request shown, in fact, no input timing is established since all input words are read without an accompanying multiprogrammer Gate. Thus, the input card's Gate to its device is not set either. If the read direct with Gate request is used, the multiprogrammer Gate is set for each input word transfer and, consequently, the input card Gate to its device is also set. Notice, further, that unless option jumper W10 is deleted, the digital input card (69431A) requires a device Flag to store data. This requirement should be evaluated with respect to the timing restrictions imposed by the Handshake Timing mode utilized by the read direct requests and the jumper deleted if necessary (the Operating and Service Manual for the HP 69431A describes in detail the options available with the card).

2-101. EXAMPLE 3 - READ OPERATOR DATA

2-102. The read operator data request is used to interact with the multiprogrammer operator in order to notify him that attention is required. The request can be used to report a system malfunction detected by the program, to request

system status or configuration data, or for any system function requiring operator intervention. The following is an example of the use of a read operator data function.

```
ICODE = 1       (read request type)
ICNWD = 110B
IBUFL = 2
IFUNC = 3       (read operator data)
CALL EXEC (ICODE, ICNWD, IBUFR, IBUFL, IFUNC)
```

```
IBUFR (1)       177ØØØ Control word to get operator's attention
                ØØØØØØ Operator data read-in area
```

After the call is executed, IBUFR is configured as follows:

```
IBUFR (1)       177ØØØ Original control word
                ØØØ6Ø5 Returned operator data
```

2-103. Notice that the output word transmitted by the driver to alert the operator is transmitted with the multiprogrammer in the Handshake Timing mode (Handshake Flag timing is monitored). Thus, the user must either have previously programmed the multiprogrammer with TME = 0 or the output word sent with the request must be a control word with IEN off (thus, the Handshake Flag is automatically returned by the multiprogrammer). Of course, the output word must be such that the operation of the system will not be adversely affected. Thus, bits 9-11 of the control word are used to alert the operator since these bits are not used (or processed) by the multiprogrammer. The operator responds to the programmed alert as follows:

1. Set multiprogrammer REMOTE/LOCAL switch to LOCAL.
2. Enter desired data into multiprogrammer switch register.
3. Touch RETURN DATA switch (this generates the multiprogrammer Flag).
4. Set REMOTE/LOCAL switch to REMOTE.

When the driver completion section is given service (through the computer interrupt system and via the CIC module of RTE) in response to the multiprogrammer Flag (step 3), it reads in the operator data (stored on the interface card) and then terminates the request.

2-104. Since the read operator request involves human intervention, do not attempt to program the multiprogrammer channel too soon after the read operator data request is issued. The operator must be given adequate time to manually switch the multiprogrammer from REMOTE to LOCAL and then back to REMOTE after entering the appropriate data. Thus, the driver will reject new requests if issued too soon since it is set to the busy state until the read operator data request is completed.

2-105. COMPARISON BETWEEN INTERRUPT AND NON-INTERRUPT DATA TRANSFER TIMING

2-106. To illustrate the difference in execution time between multiprogrammer data transfers that use the interrupt system and those that do not, an identical 70-word output buffer was transmitted several times using the Normal Write and Write with Handshake Flag requests. For both requests, moreover, the multiprogrammer was set to the Handshake mode (TME = 0). In every case, the Write with Handshake Flag was executed at least three times as fast as the Normal

Write. The reason for this, of course, is the fact that even with TME off, the Normal Write uses the computer interrupt system, thus incurring RTE system overhead for each word transferred. Thus, it is assumed that the time saving will be even greater for longer buffers. This time saving could be especially significant when utilizing the sense mode of operation in that the sense mode is suspended whenever normal I/O data transfers are in process.

2-107. EXAMPLE 4 - SENSE MODE PROGRAMMING

2-108. The sense mode allows event sense cards to monitor important events (input lines) during the time that I/O data transfers are not in process. The sense mode could be used, for instance, to enable event sense cards that are monitoring system environmental conditions which, if exceeded, might require that immediate steps be taken to alleviate the problem. Suppose, for instance, that one input to an event sense card is a relay contact that is controlled by a temperature sensing device that is monitoring the temperature of an industrial process. If the temperature exceeds a certain value, the relay contact closes and the event sense card detects an alarm. This alarm must be processed quickly in order, for example, to allow the processing program to output control data to a relay output card that will cause a decrease in temperature. The sense mode is especially designed to handle this problem, and any requirement for continuous monitoring of and rapid response to critical events. Of course, the response time will depend on the normal I/O data transfers programmed as well as the priority given to your alarm servicing program. The following example is an illustration of how the sense mode might be used.

2-109. Assume for the example that the HP 6940A channel is logical unit number 10_8 and that ten event sense cards are to be enabled in the sense mode. All cards, in addition, are installed in the HP 6940A, unit ØØ. This configuration takes advantage of the automatic polls that the driver makes in response to a sense mode interrupt so that your alarm processing program does not have to collect interrupt data itself. The event sense cards could, of course, be installed in other units in which case the processing program would have to issue a poll request to the driver to read the interrupt data.

2-110. In addition to the above, assume that all of the event sense cards have jumper W6 installed so that they will all be activated whenever the multiprogrammer's IEN mode is programmed. Of course, if you want the option of deactivating card(s), then you must remove W6 on the card(s) and individually activate them by addressing and gating them in the ISL mode (i.e., issuing a write request with the appropriate buffer). Note that there should not be any other type input cards (i.e., the HP 69431A's) active during the sense mode. That is, if the W6 jumper is installed on an HP 69431A, it will be activated each time the interrupt enable mode is programmed. Of course, if an HP 69431A returns a Flag to the computer, it will cause a sense mode interrupt just like an event sense card.

2-111. Assume further, that the W3 jumper on each event sense card is connected to detect inequality between the reference word stored on the card and the external data input. Finally, assume that the initial external data input to each card is to be all zeroes so that the initial reference word transmitted to each card will be zero.

2-112. To initialize the event sense cards, then, the program can first (before it turns on the sense mode) send the initial reference words to the event sense cards with the following Write with Handshake request:

```
ICODE = 2       (write request)
ICNWD = 110B
IBUFL = 11 10
IFUNC = 1       (write with Handshake Flag)
CALL IEXEC (ICODE, ICNWD, IBUFR, IBUFL, IFUNC)
```

The above EXEC call transmits the 11-word IBUFR which, of course, must be previously created. For the example, assume IBUFR was configured and transmitted as follows:

IBUFR (1)	17ØØ4Ø	Control word to select unit ØØ; keep SYE on for output cards.
	ØØØØØØ	
	Ø1ØØØØ	
	Ø2ØØØØ	
	Ø3ØØØØ	Data words transmitted to card slots ØØ-Ø9$_{10}$. The event-sense cards store reference words ØØØØ.
	Ø4ØØØØ	
	Ø5ØØØØ	
	Ø6ØØØØ	
	Ø7ØØØØ	
	1ØØØØØ	
IBUFR (11)	11ØØØØ	

Note a few things. The control word includes SYE in order to allow previously-programmed output cards to maintain their outputs. Also, TME is not programmed so that the HP 6940A Handshake Flag is returned for each data word output. After the event sense cards have been programmed with the initial reference words, then, the program can turn on the sense mode to activate all of the cards to enable them to interrupt if an event change occurs. The following Set Sense Mode I/O Control EXEC call can be issued to turn on the sense mode:

```
ICODE = 3
ICNWD = 2012B
CALL EXEC (ICODE, ICNWD)
```

2-113. After the above I/O Control EXEC call has been executed, the multiprogrammer assigned to logical unit 10$_{10}$ will be programmed to the interrupt enable mode (control word 17Ø26Ø$_8$) whenever normal I/O data transfers are not in process. Any I/O data transfer request directed to logical unit 1Ø will cause the driver to disable the interrupt enable mode (remove the Gate to the multiprogrammer) intil the request is completed. For purposes of this example, assume that some time after the sense mode was programmed on, an external input to an event sense card in unit ØØ changes and the card's CTF circuit causes a computer interrupt. The continuator, of course, is given service by RTE and it recognizes the interrupt as having occurred during the sense mode. The continuator, then polls unit ØØ; stores the slot number and data of the interrupting card in your processing program's ID Segment (words 2 and 3); schedules your processing program; clears the APB [EQT5 (1) = Ø] since an "alarm is pending" means that a sense mode interrupt occurred and the driver could not schedule the alarm processing program

(i.e., found it not dormant because it was scheduled in response to a previous interrupt); again polls unit ØØ, storing the interrupting card's slot and data in the processing program's ID Segment words 4 and 5; stores the channel number of the HP 6940A from which an interrupt occurred; and turns off the interrupt enable mode before exiting back to RTE.

2-114. It is now up to your alarm processing program to process the event. When the program runs (it should be of high enough priority that it runs quickly), it can retrieve the event data from its ID segment by making an immediate call to RMPAR (see Paragraph 2-74). As part of its alarm processing, the program should reset the alarm (send the appropriate reference word to the event sense card) before another request is directed to the driver which might turn the interrupt enable mode back on. Remember, the sense mode bit is still on so if the interrupt enable mode is programmed, the card that originally interrupted will immediately do so again. The processing program, then, could issue an event sense poll all request to clear the previous alarm (update and reactivate any card(s) that interrupted) as follows:

```
ICODE = 1       (Read request)
ICNWD = 110B
IBUFL = 23
IFUNC = 8       (Event Sense Poll All)
CALL EXEC (ICODE, ICNWD, IBUFR, IBUFL, IFUNC)
```

The above EXEC call polls all of the cards identified in IBUFR performing the following tasks: (1) reads in the data; (2) if a card had interrupted (its IRQ is on), transmits the data back to it as the new reference word; and then (3) reactivates the card (addresses and Gates the card in the ISL mode). For this example, IBUFR should be previously created as follows:

```
IBUFR (1)       17Ø24Ø    Poll all initial control word
                ØØØØØ1    Index pointing to next word
                ØØØØØØ
                Ø1ØØØØ  —— Address words for the ten event sense
                   .        cards in unit ØØ.
                   .
                   .
                11ØØØØ
                177777    Stopper
                ØØØØØØ
                        —— Ten word read-in area
IBUFR (23)      ØØØØØØ
```

2-115. After the event sense poll all request is executed, then, the event sense cards are updated and rearmed and the interrupt enable mode programmed on if the sense mode bit [EQT5 (Ø)] was not cleared.

2-116. DRIVER ORGANIZATION

2-117. As previously described, the DVR61 driver is organized into initiation/completion sections that are typical of most HP RTE interrupt drivers. Some requests, as has been noted, are executed entirely in the initiation section while some requests utilize the initiation and completion section of the driver and, as a result, the computer interrupt system. Figure 2-7 through 2-14 provide simplified overviews of the processing sequence employed by the driver to execute each request.

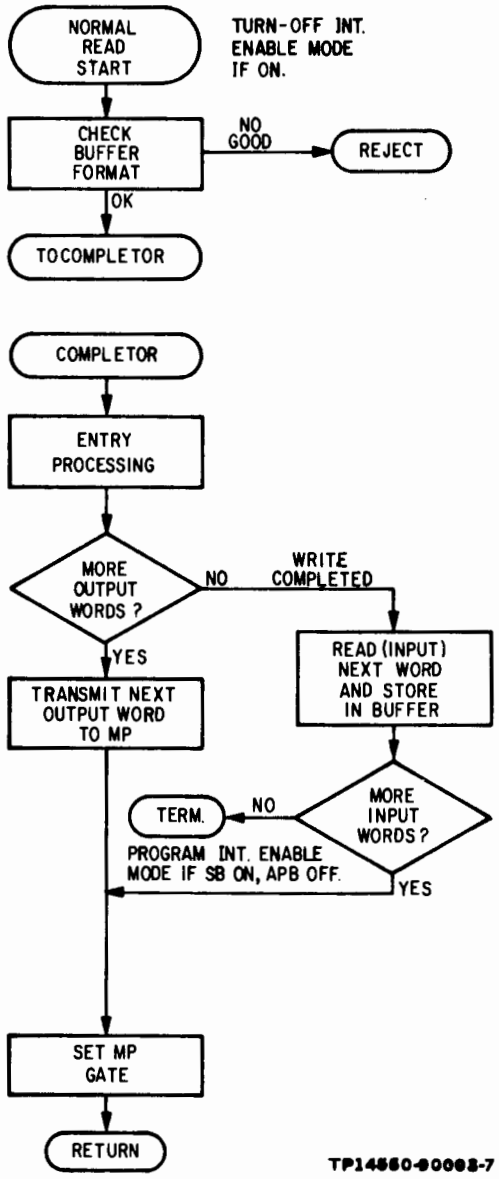


Figure 2-7. Normal Read Simplified Processing Sequence

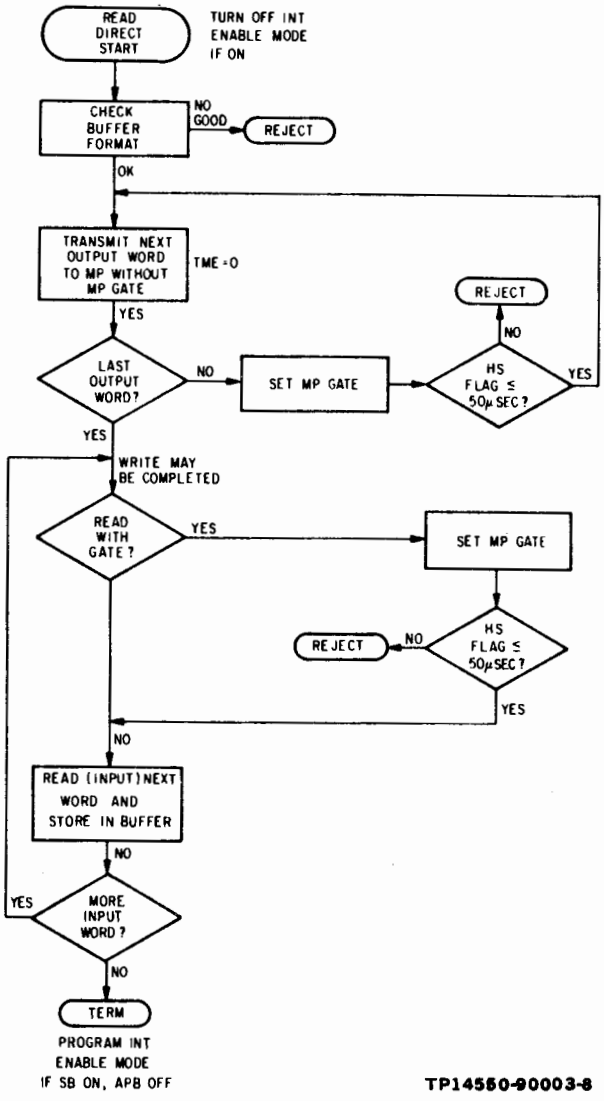


Figure 2-8. Read Direct Requests Simplified Processing Sequence

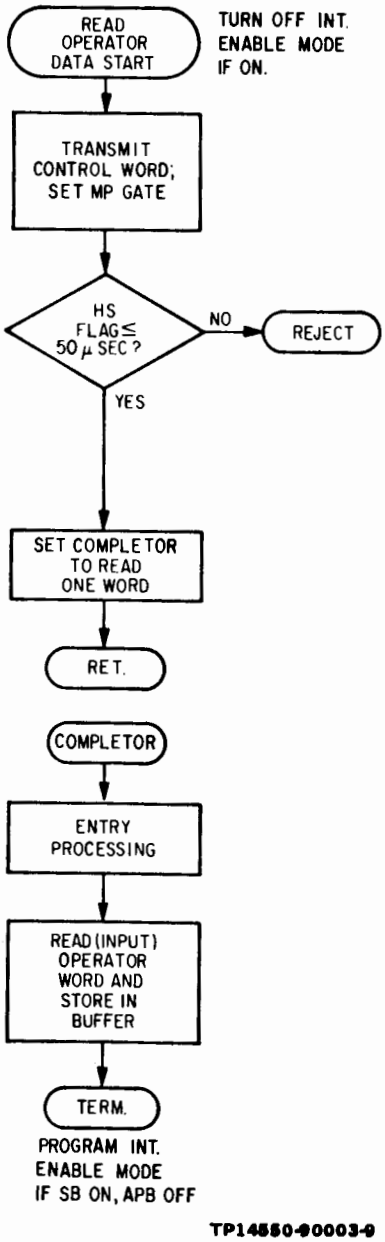


Figure 2-9. Read Operator Data Simplified Processing Sequence

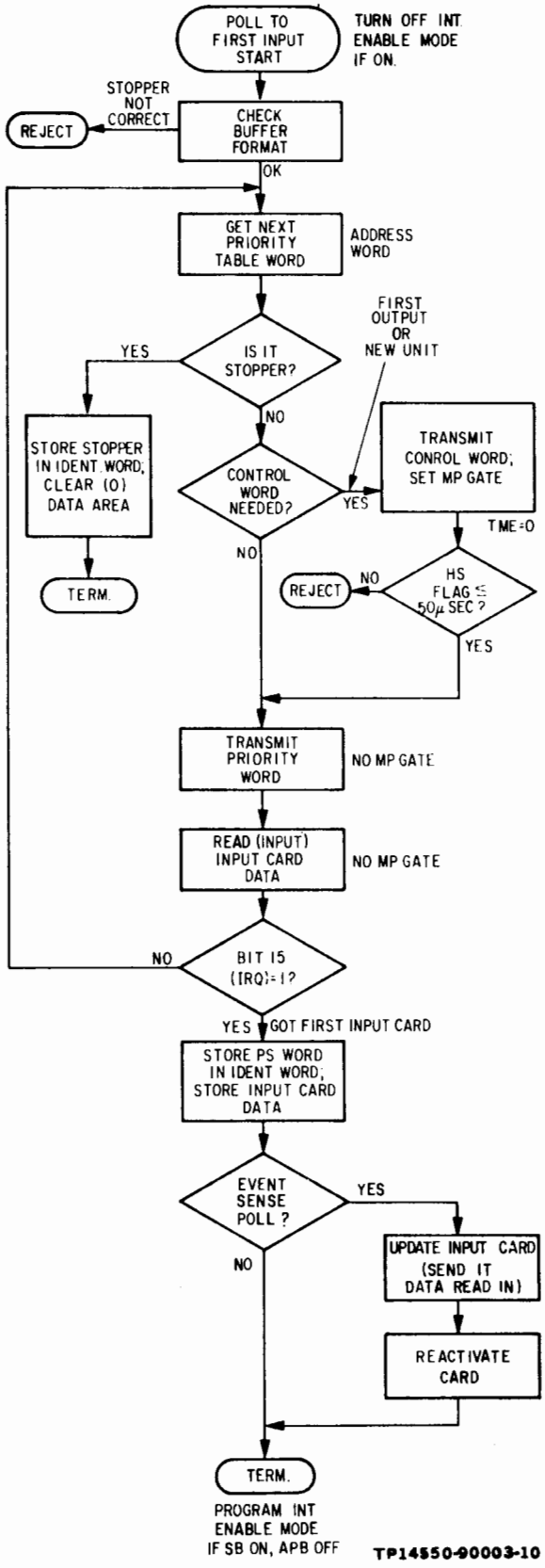


Figure 2-10. Poll to First Input Simplified Processing Sequence

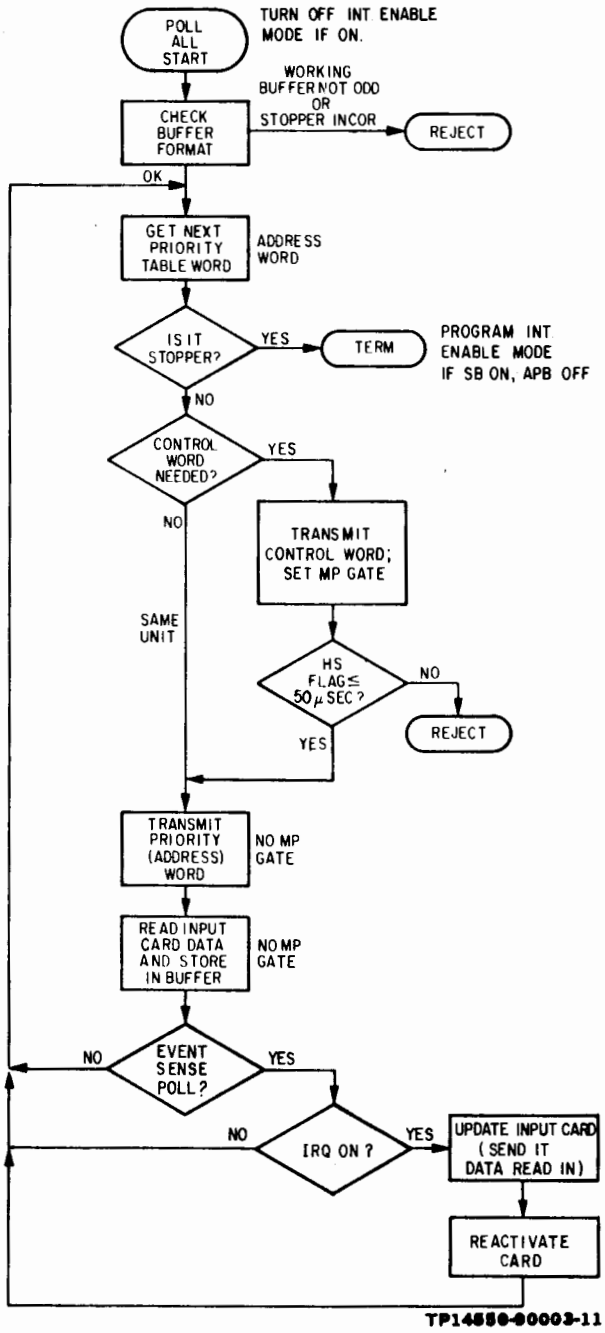


Figure 2-11. Poll All Simplified Processing Sequence

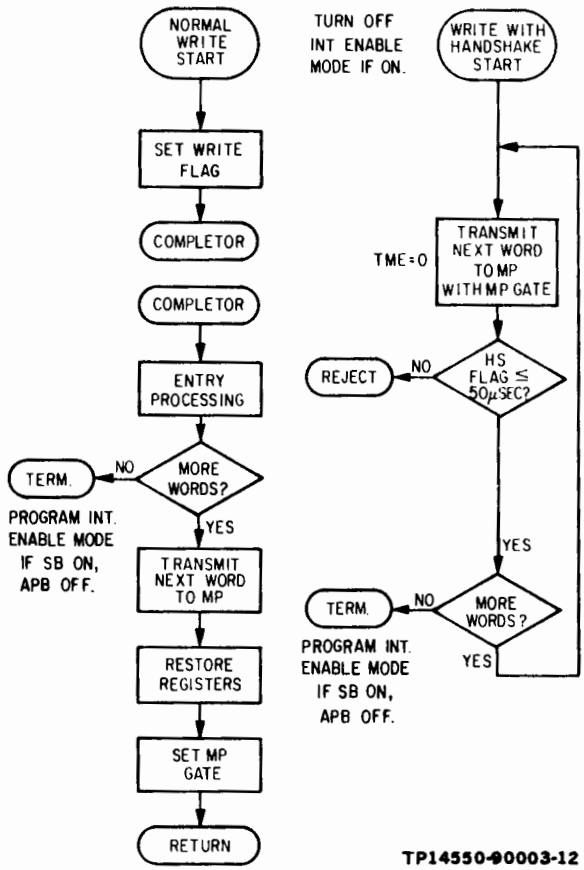


Figure 2-12. Write Requests Simplified Processing Sequence

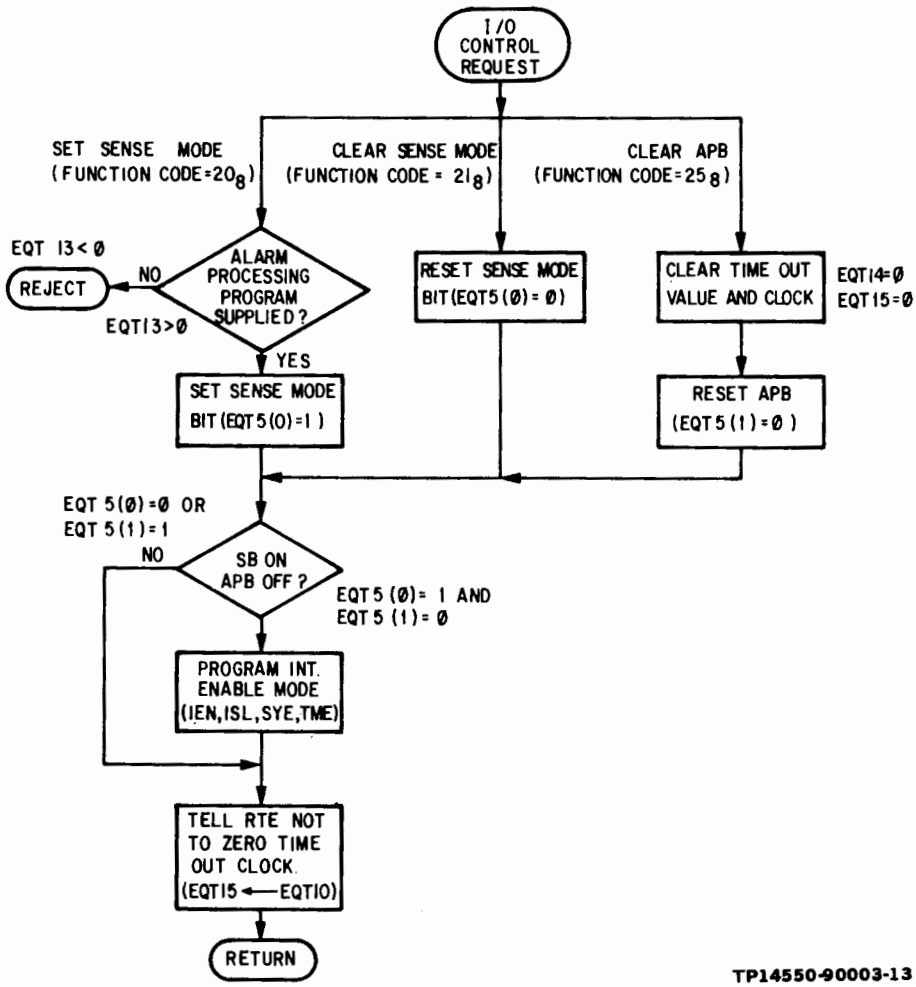


Figure 2-13. I/O Control Requests Simplified Processing Sequence

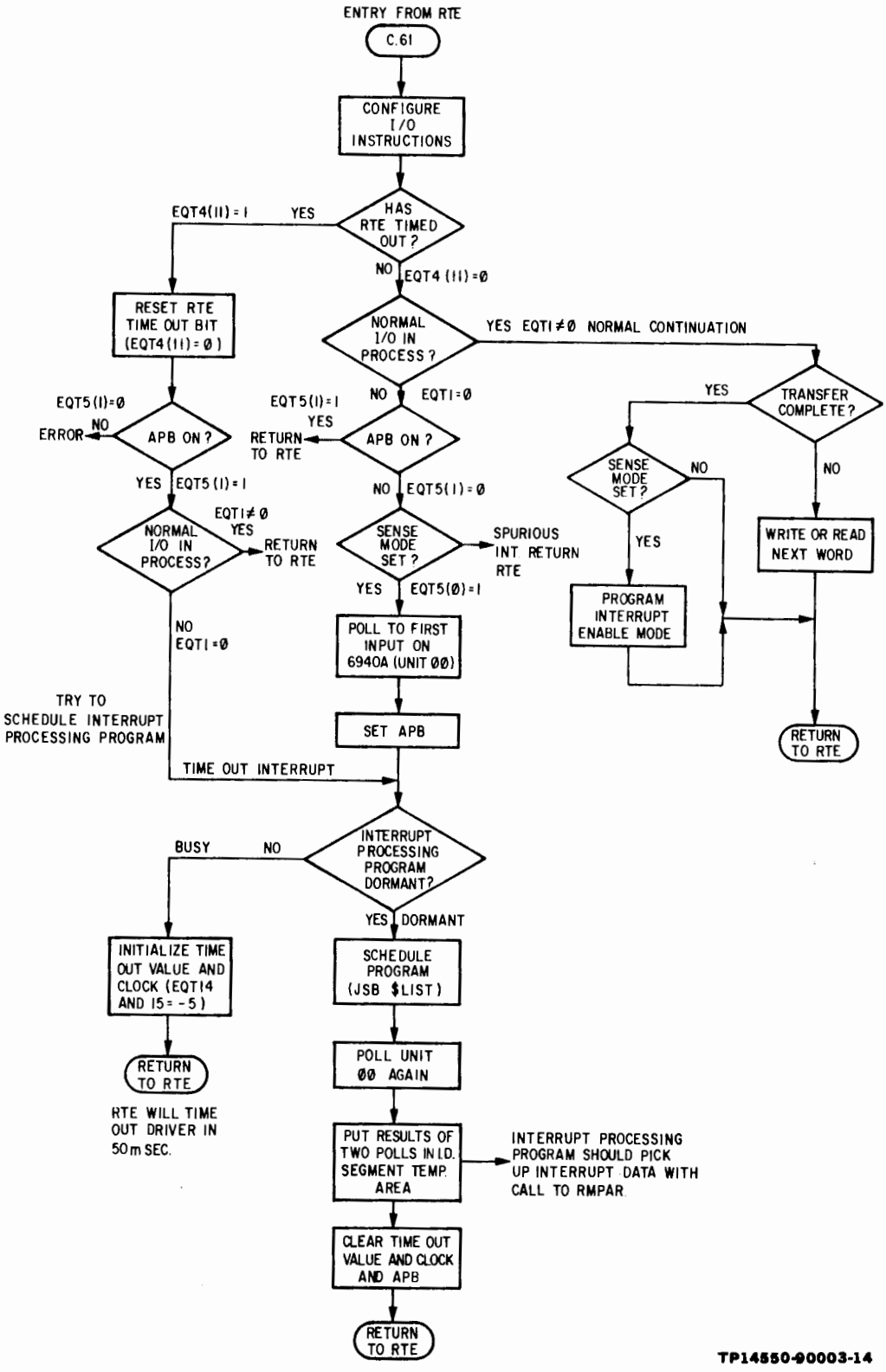


Figure 2-14. Continuator Simplified Processing Sequence

SECTION III
CONFIGURATION INFORMATION

3-1. GENERAL

3-2. This section provides configuration information for Driver DVR65 and is intended to augment the data provided in the Real-Time Executive Software System Programming and Operating Manual.

3-3. REAL-TIME GENERATION

3-4. DVR61 is incorporated into the RTE system at system generation. If none of the HP 6940A channels in the system are to be operated in the sense mode (i.e., event sense cards programmed to the interrupt enable mode whenever normal I/O is not in process), only DVR61 is required in the RTE system. However, if the sense mode is required, the user must incorporate an event sense interrupt processing program as well as DVR61 into the RTE system when it is generated. The interrupt processing program should be of high enough priority to assure that it will be executed whenever it is scheduled. To incorporate the interrupt processing program at system generation:

a. Supply the program (in relocatable form) during the Program Input Phase. It is recommended that the program be assigned to a core-resident area (type 1 or type 4).

b. During the Loading Phase, DVR61 is assigned to the HP 6940A channel(s) select code(s) in the Equipment Table Entry(s). Next, the EQT Entry(s) for the HP 6940A channel(s) is/are assigned Logical Unit Number(s) in the Device Reference Table. In the Interrupt Table, however, the select code(s) of the HP 6940A(s) are specified as PRG [program(s)], and the interrupt processing program name(s) for the HP 6940A channel(s) is/are assigned to the Interrupt Table entry(s). For example:

```
13, PRG, ALARM
```

would assign alarm program ALARM to HP 6940A I/O channel 13_8. The first time the driver is called on each of the HP 6940A channels for any request, it examines the Interrupt Table entry for the channel. If the entry is negative [RTE puts the negative of the program(s) ID segment address in the Interrupt Table], the driver puts the address of the interrupt processing program's ID segment into EQT13. Further, the driver replaces the Interrupt Table entry for the channel with the address of its EQT entry. Thus, the driver will be called for interrupts on the channel and, in addition, it can schedule the interrupt processing program (since it has the ID segment address) when a sense mode interrupt on the channel occurs and, also, pass interrupt data that it collects to the program.

c. To implement the sense mode of operation, the driver handles time outs for the HP 6940A channel(s). Thus, if the sense mode is to be used on a channel, the driver should not be assigned a time out parameter in its EQT entry. Further, the driver should not be assigned a time out value by the operator during system operation.

d. The sense mode alarm processing program supplied by the user should only be called by the driver (i.e., when a sense mode interrupt occurs). Therefore, a time value should not be assigned to the program when it is written (the time parameters in the program statement should be Ø) or by the operator during system operation.