HEWLETT PACKARD

# RTE-6/VM Serial Driver

## Reference Manual

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

Manual Part No. 92084-90050
E0693

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

## NOTICE

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

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

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



RESTRICTED RIGHTS LEGEND

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

# Printing History

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

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

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

| | | |
|---|---|---|
| First Edition | Jan 1989 | Rev. 5010 (Software Update 5.1) |
| Update 1 | Jul 1990 | Rev. 5020 (Software Update 5.2) |
| Second Edition | Jun 1993 | Rev. 6000 (Software Update 6.0) |

# Preface

This manual describes the serial I/O drivers supported by the RTE-6/VM Operating System. These drivers allow the operating system to communicate with peripheral devices via the computer's interface cards. The drivers are programmed via standard EXEC I/O requests.

This manual is written for the system programmer or system manager experienced with the RTE-6/VM Operating System and familiar with HP 1000 peripherals. The manual assumes a familiarity with the HP 1000 A-Series Computer I/O structure.

Because the situation may arise when it is necessary to write programs that will run on both RTE-6/VM and RTE-A, this manual also contains information on drivers supported by the RTE-A Operating System.

This manual contains the following chapters and appendixes:

| | |
|---|---|
| Chapter 1 | lists the drivers supported by RTE-6/VM and RTE-A, and discusses compatibility issues when writing programs to run on both systems. |
| Chapter 2 | describes the EXEC calls to use when writing programs to run on RTE-6/VM drivers. |
| Chapter 3 | explains the differences between DV800 and DVM00, and DVC00 and DVR00. |
| Appendix A | contains an example protocol chart for block mode ASCII reads and writes. |
| Appendix B | discusses system generation considerations that enable you to preconfigure the drivers for boot up. |
| Appendix C | contains a sample page mode application using Revision 5010 serial I/O drivers. |
| Appendix D | illustrates the ENQ/ACK handshake. |
| Appendix E | lists the HP character sets. |

# Table of Contents

## Chapter 1
## Introduction



## Chapter 2
## User-Level Interface

# List of Illustrations



# Tables

# 1

# Introduction

This chapter describes the Revision 5010 serial I/O drivers supported on the RTE-6/VM Operating System. The serial I/O drivers were introduced to gain the advantages of universal device driver compatibility, ease of use, speed, efficiency, increased applicability, and reduced table size. The transition of the earlier revision drivers to the Revision 5010 drivers is summarized in Table 1-1.

Because it is possible to write programs that run on both the RTE-6/VM and RTE-A Operating Systems, this chapter also contains information on the RTE-A serial I/O interface and device drivers. Table 1-2 lists the RTE-A serial drivers. For complete information on the RTE-A serial I/O interface and device drivers, refer to the *RTE-A Driver Reference Manual*, part number 92077-90011.

**Table 1-1. RTE-6/VM Serial Drivers**

| Driver | | Interface Card |
|---|---|---|
| **Previous** | **Rev. 5010** | |
| DVR00 | DVC00 | HP 12531C TTY |
| DVR00 | DVC00 | HP 12531D HS Terminal |
| DVR00 | DVC00 | HP 12800 CRT |
| DVM00 | DV800 | HP 12792C 8-Channel MUX |

**Table 1-2. RTE-A Serial Drivers**

| Driver | | Interface Card or Device |
|---|---|---|
| **Previous** | **Rev. 5010** | |
| DD.00 | DDC00 | Terminal |
| DD.20 | DDC01 | Terminal and CTU |
| IDM00 | ID800 | HP 12040 8-Channel MUX |
| IDM00 | ID801 | HP 12040 8-Channel MUX w/ modems |
| IDM00 | IDM00 | HP 37222 Integral Modem Card |
| IDM00 | ID400 | HP 12100A A400 4-Channel MUX |
| ID.00 | ID.00 | HP 12005 ASIC |
| ID.01 | ID.01 | HP 12005 ASIC w/ modems |

# RTE-6/VM and RTE-A Compatibility Information

When writing programs to run on both RTE-6/VM and RTE-A, it is important to keep compatibility in mind. Simple read and write capabilities are compatible in the Revision 5010 or later drivers. When writing programs that use more advanced driver features, consider the following areas.

## Timeout Bits

The RTE-6/VM drivers prior to Revision 5010 use bit 0 of the status byte to indicate that a timeout has occurred. The Revision 5010 or later RTE-6/VM drivers and the RTE-A drivers use bit 1. To determine which bit is being used, include the following code fragment in programs:

```
TimeOutMask = 2                          ! Assume bit 1 for RTE-A and new
if ( HpRte6( ) ) then                    ! RTE-6/VM drivers.  In RTE-6/VM,
  if ( .not.HpCrtSSRCDriver(LU) ) then   ! old drivers need bit 0
    TimeOutMask = 1
  endif
endif
```

## Program Scheduling

The serial drivers in both RTE-6/VM and RTE-A allow you to schedule a program upon unsolicited interrupt from the user. With RTE-6/VM drivers prior to Revision 5010, the program to be scheduled is determined during system generation and there is no HP-supplied way to change the selection online. With Revision 5010 or later drivers, the selected program can be changed online. However, direct calls to the drivers (the CN 20 requests) have different formats on the two systems because of operating system differences. Subroutines to hide the differences exist in the relocatable library (HpCrtSchedProg and HpCrtSchedProg_S).

A second consideration is that RTE-A provides a two-level program scheduling mechanism with both a primary and a secondary program, while RTE-6/VM provides only a single level. For compatible implementations, use the RTE-A primary level only.

## LU Number Restrictions

RTE-6/VM and RTE-A handle the problem of LU numbers greater than 63 in different ways because of the presence of a session control block in RTE-6/VM. There are subtleties involved when writing programs that are to control LUs outside your session, or if you are not in session. However, the general rule is that you must use XLUEX calls for all I/O if you wish to have portable programs. The user's terminal is an exception; for both operating systems ensure that I/O to LU 1 is mapped to the LU where the user is logged in.

## FIFO Mode and Type-Ahead Mode

FIFO mode, or its predecessor type-ahead mode, are not available on all interface cards. Programs that depend upon these modes restrict users to using only certain ports on the system. Depending on the complexity of your application, you may be able to write code to interchangeably use FIFO mode or type-ahead mode. More complex applications will require code that handles FIFO mode and type-ahead mode differently. The Revision C multiplexers have type-ahead and the Revision D multiplexers have FIFO. The following paragraphs describe the two modes; refer to Figure 1-1 for conceptual block diagrams.

Type-ahead allows input from a MUX port to be saved in a buffer until a read is posted to retrieve the data. This feature permits a limited amount of full-duplex I/O so that input and output occur simultaneously. The unusual aspect is that type-ahead has two buffers of up to 255 bytes each. When input data is received, it is saved in the active buffer until a terminator is received. Because the type-ahead is active when a read is not active, the terminator must be defined by a method other than the usual function bits in the EXEC call control word. The method used is two control calls: CN 37 (Set Read Type) and CN 36 (Set Read Length). Assuming that carriage return is selected as the terminator, the input data is saved in buffer 1 until the first carriage return and then in buffer 2 until the next one. If more data is received after that, it is discarded. This causes two carriage returns in a row consuming both buffers, even though only two bytes were received. In addition, the data is echoed as it is put in the type-ahead buffer even though the future read might not indicate to echo the data.

To summarize, type-ahead mode has the following characteristics:

- Two buffers up to 255 bytes long between the receiver and the CPU.
- Buffers are ‘filled’ when a terminator byte is received.
- Data echoes as received.
- The type of buffer termination can be selected.

On the Revision D MUX, however, a one-kilobyte buffer is implemented between the output of the UART and the user. The UART (Universal Asynchronous Receiver/Transmitter) on the MUX card serves as a serial-to-parallel converter for incoming data and a parallel-to-serial converter for outgoing data. When incoming data is received, it is put into the FIFO without echo. No interpretation of the data is done, as it is not yet known what kind of read will be removing the data from the FIFO. Thus the full kilobyte can be used without regard to ‘record’ lengths. When a read is posted by a program, the bytes are removed from the FIFO and interpreted according to the rules established by the current request. Each byte is also echoed, if echo is enabled, and compared against the current terminating conditions.

To summarize, FIFO mode has the following characteristics:

- The 1024-byte circular queue is logically between the output of the UART and the receiver.
- Characters are not echoed until a read gets the data.
- The concept of a FIFO buffer terminator does not apply, as the data is not examined until a read is posted and the receiver section retrieves the data from the FIFO.

Note that the Revision C MUX can be upgraded to the Revision D MUX by installation of new firmware. Contact your local Hewlett-Packard Sales Representative for ordering information.

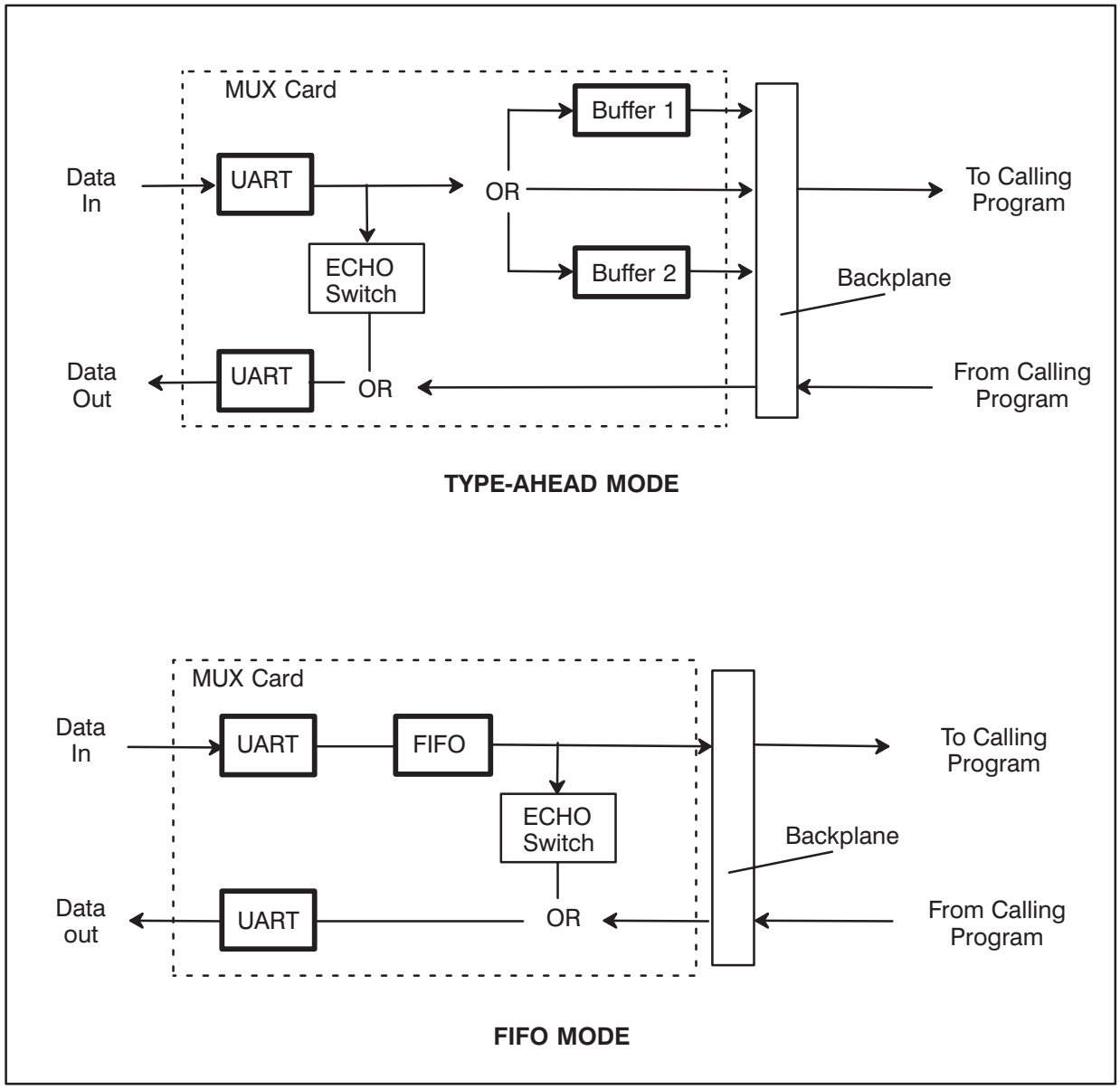



**Figure 1-1. Conceptual Block Diagram of Type-Ahead Mode and FIFO Mode**

## Protocols

Several of the protocols available on the Revision 5010 or later drivers, such as bi-directional Xon/Xoff and CPU-to-CPU, are not available on drivers prior to Revision 5010. Do not use these protocols if your code must be transportable to older revision drivers.

## Screen Mode

In screen mode reads, some older drivers return the Unit Separator characters to the user buffer. The newer drivers do not. To hide this difference, use the HpCrtStripChar routine from the relocatable library.

# 2

# User-Level Interface

This chapter describes the EXEC calls that can be used with the RTE-6/VM serial drivers. Unless otherwise noted, the following information applies to all the drivers. For general information on EXEC calls, refer to the *RTE-6/VM Programmer's Reference Manual*, part number 92084-90005.

## Read Request

The calling sequence for a read request is

```
CALL EXEC(1,cntwd,bufr,bufln[,pram3,pram4])
```

### BUFR and BUFLN

The *bufr* parameter is the user buffer that receives data from the read request. *bufr* cannot be a FORTRAN character data type. Refer to the HpCrtReadChar subroutine in the *RTE-A • RTE-6/VM Relocatable Libraries Reference Manual*, part number 92077-90037, for information on FORTRAN character variables.

The *bufln* parameter is the request length. It defines the maximum transfer size. Fewer characters may be transferred in an ASCII mode read. After the read, the B-Register contains the transmission log, a positive value in the same units as the original request length indicating the number of valid characters stored in the user buffer.

If *bufln* is positive, it indicates the maximum number of words to be transferred (0 to 32767 words, 0..077777B), half the number of characters. The transmission log is expressed in a positive number; it equals the rounded-up character count divided by 2. If *bufln* is negative, it indicates the number of characters to be transferred (1 to 32768 bytes, 177777B..100000B). The returned transmission log will be the positive number of characters received. In binary mode, a zero-length read terminates immediately with a transmission log of zero. In ASCII mode, a zero-length request terminates when the terminator or timeout is detected with a transmission log of zero.

If the input data ends on an odd (left) byte, the last word is padded with a blank (octal 40) in ASCII mode or a null (octal 0) in binary mode. This byte is *not* included in the transmission log computation. In all cases, the contents of the user buffer beyond the last valid word as indicated by the transmission log is undefined. There is no guarantee that the buffer has not been altered by any editing that may have occurred, nor should it be assumed that the terminator byte (if any) is present in the buffer.

## CNTWD (Read Request Control Word)

The read control word (*cntwd*) has the following form:

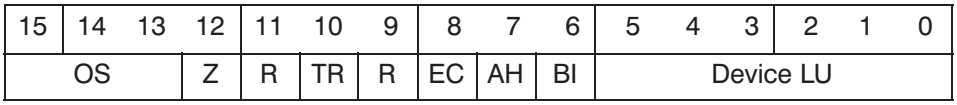



**Figure 2-1. Read Request Control Word (CNTWD)**

**OS**: RTE option bits; refer to the *RTE-6/VM Programmer's Reference Manual*, part number 92084-90005.

**Z:** Write/Read bit. If the Z bit is clear, *pram3* and *pram4* in the call are ignored.

If the Z bit is set, *pram3* and *pram4* in the call describe a buffer (referred to as the Z-buffer). The Z-buffer is written to the external device before a read or write, as described by *bufr* and *bufln*. *pram3* is the buffer address. *pram4* represents a character count, if negative, or a word count, if positive. The Z-buffer is sent in normal ASCII mode, with the data bytes followed by a driver generated Carriage Return Line Feed (CRLF), unless disabled by a trailing underscore.

When HP protocol is enabled, the control buffer is sent with ENQ/ACK handshaking enabled. The normal data buffer, described by *bufr* and *bufln*, is transferred according to the rules established by the remainder of the bits in the control word.

With Z-buffer calls, you can execute a command and a data transfer in a single call. A write call followed by a read call cannot always respond quickly enough to prevent loss of data, especially in the case where the external device has a quick "turnaround" time and the user program is pre-empted by other higher priority programs. The write/read call should be able to receive the data under almost all circumstances following the transmission of the Z-buffer because the I/O card switches from the transmit mode to the receive mode in less than one character's time.

**TR:** Transparency bit. During an ASCII read, this bit determines whether special character processing is done. When this bit is clear, the incoming data stream is examined for the presence of special characters, which cause the contents of the user buffer to be changed before the read is completed. This is sometimes referred to as input editing. If the transparency bit is set, the termination character is the only special character. The termination character can be altered. Refer to the section "Function Code 17B Definable Terminator" later in this chapter.

This bit and the binary bit (bit 6) indicate read request type as described in Table 2-1.

**R**: Reserved; should be set to 0.

**EC:** Echo bit. Echo is enabled when this bit is set and disabled when it is clear.

**AH:** Auto-Home bit. Auto-home has no effect in character mode transfers. Refer to the "Block Mode Read" later in this chapter.

**BI**: Binary bit. Binary reads are performed when the binary bit is set. A binary read ends when the user's request length has been satisfied, that is, the buffer has been filled or timed out. All characters received are saved in the buffer.

ASCII reads are performed when the binary bit is clear. An ASCII read ends only when a terminator character is detected in the incoming data stream or timeout. The terminator character is a delimiter only, and as such may not be saved in the user's buffer. If the buffer receives more characters than it can hold before it receives the terminator character, the excess characters are lost without notification.

This bit and the transparency bit (bit 10) indicate read request type as described in Table 2-1.

**Table 2-1. Character Mode Read Types**

| Read Type | TR BI | Terminating Condition | Description of Read |
|---|---|---|---|
| NORMAL ASCII | 0 0 | Detection of CR or EOT. | Allow special character recognition. If echo enabled, echo each incoming character as received. Send LF for terminating CR if echo is enabled, send CRLF if disabled. |
| | | Does not terminate on character count. | Send "\CRLF" for DEL if echo enabled, send nothing if disabled. |
| TRANSPARENT ASCII | 1 0 | Detection of CR (or other if CN 17 has changed it). | No special character recognition other than the terminator . If echo enabled, echo each incoming character as received. Send LF for terminating CR if echo is enabled, send CRLF if disabled. |
| | | Does not terminate on character count. | Note that the terminating condition may be changed with a CN 17 call. |
| BINARY | X 1 | Satisfy character count. | If echo enabled, echo each incoming character as received. |

Note: This table is for character mode transfers. For information concerning block mode transfers using HPCRT calls, see the Block Mode Read section later in this chapter.

## Special Characters

The special characters recognized in normal ASCII reads are:

| | | |
|---|---|---|
| Carriage Return | (octal 015) | (Ctrl-M) |
| Line Feed | (octal 012) | (Ctrl-J) |
| Backspace | (octal 010) | (Ctrl-H) |
| Delete | (octal 177) | |
| EOT | (octal 004) | (Ctrl-D) |

The special characters are described in the following subsections.

### Carriage Return (octal 015) (Ctrl-M)

This character is the read terminator for normal ASCII reads and the default terminator for transparent ASCII reads. The read request is complete when the carriage return is received and a CRLF sequence is output by the driver to move the cursor to the next line on the CRT.

### Line Feed (octal 012) (Ctrl-J)

This character causes different results, depending on the driver being used. DVC00 and DV800 leave line-feed characters in place. DVR05 strips line feeds from the incoming data. For applications where DVC00 and DV800 drivers may be used and line feeds are generated by the device connected to the serial interface, the utility subroutine HpCrtStripChar should be called to ensure proper operation.

### Backspace (octal 010) (Ctrl-H)

This character causes the preceding character in the buffer to be erased. Successive backspace characters remove a character at a time from the buffer until the buffer is empty. Note that if echo is enabled and the read is not transparent, the cursor moves to the left on the screen. The drivers then issue a blank, causing the character to disappear from the screen, followed by another backspace to reposition the cursor. Thus as each backspace is entered, a character is erased from the screen.

For backward compatibility, DVC00 also recognizes (octal 1) (Ctrl-A) and (octal 31) (Ctrl-Y) as backspace characters. If the character is Ctrl-A, an underscore character (_) instead of a space is sent as the erasing character.

### Delete (octal 177)

This character, which is labeled DEL or RUBOUT on some terminals, causes the driver to reset the buffer pointer. This results in the driver forgetting all the characters that have been entered into the buffer, and the user can “start over”. If echo is enabled and the read is not transparent, the driver acknowledges the delete key by sending a backslash (\) CRLF sequence to the terminal.

### EOT (octal 004) (Ctrl-D)

When this character is received and the read is a normal ASCII read, it causes the read to be terminated, the transmission log to be set to zero, and bit 5 to be set in the status bits. The cursor position is *not* altered (no CRLF is issued). Any data that the user entered prior to the Ctrl-D is lost.

## Break

The break condition terminates a read, write, or control request. DV800 performs program scheduling upon receipt of a break condition or an unsolicited character (an unsolicited character is any character that arrives when the driver is not executing a read).

Some driver/interface combinations have FIFO buffer modes that effectively leave a read pending at all times. When this mode is invoked, the BREAK key is a more reliable way to schedule the prompt program. The BREAK key is a key present on most keyboards that causes a long space condition to be sent on the communications line, approximately 250 milliseconds. BREAK is a condition on the line, not a character. BREAK is always recognized, even if a read is pending.

## ASCII vs. Binary Read Modes

One advantage of a normal ASCII read is that no action is taken until a carriage return is entered. You can change the input buffer to correct errors or decide that what was entered is not desired. This is essential in creating a "friendly" program. When single-character reads are used, each keystroke is evaluated as it is entered, thereby denying the user an opportunity to think about the input.

Additionally, there are hardware advantages to posting a read that terminates on a character instead of a count. For some I/O cards the CPU overhead is considerably less for an ASCII read than a binary read. Input via a series of single-character binary reads entails interaction up to the program level for each character. The program may not be the highest priority program and may have even been swapped out to disk, thereby causing dropped characters if the next read is not posted within one character time. Unless FIFO buffer mode is in effect, the system will lose characters when even moderate baud rates are used with single-character input.

For these reasons, you should avoid single-character reads in all but the most trivial cases (such as the permission prompts used by LI).

## Block Mode Read

The Auto-Home bit is used by block mode reads. Echo is inhibited in block mode, therefore, the Echo bit is ignored in block mode reads. During a block mode read, if the first character received is DC2, then a DC1 is sent. The TR bit is set to inhibit the CRLF normally sent at termination.

There are two types of block mode reads: line mode and page mode. The difference between line mode and page mode is the terminating condition. In line mode, the block mode read is terminated by a CR. In page mode, it is terminated by an RS.

Block mode can be set using the terminal's block mode key or the appropriate escape sequence. Refer to the documentation for your terminal to configure your terminal for line or page block mode. To ensure that the driver and the terminal are in the same mode of operation, a CN 25 call must be issued after each terminal mode change. HP block mode transfers are only available if HP protocol is selected and the terminal is configured with its G and H straps enabled. Refer to the CN 34 and CN 25 sections of this manual.

If there is a driver mismatch of the terminal mode and the driver mode, the terminal appears to lock up or hang. If the terminal is in block mode and the driver is in character mode, the terminal hangs, waiting for a DC1/DC2/DC1 handshake sequence that is not issued by the driver in character mode. You can escape from this condition by performing a soft reset of the terminal. If the terminal is in character mode and the driver is in block page mode, entering an RS character (Cntl-^) from the keyboard terminates the read. In both cases, the read eventually terminates if the LU has a non-zero timeout set. For this reason, it is strongly recommended that all terminals have timeouts assigned.

For your convenience, the HpCrt subroutines HpCrtPageMode, HpCrtLineMode, and HpCrtCharMode are available and should be used to set the terminal and initiate the CN 25 call. These subroutines do the required CN 25 call for the user. For more information about these subroutines, refer to the *RTE-A • RTE-6/VM Relocatable Libraries Reference Manual*, part number 92077-90037.

## AH: Auto-Home Bit

The auto-home bit is set to inform the driver that it should output a control sequence during the long handshake used by a terminal in *block page mode only*. The control sequence emitted is "Esc c Esc H", which is the command to lock the keyboard and home the cursor (including transmit only fields).

The purpose of the home cursor is to remove the restriction that you must manually position the cursor to home before pressing the ENTER key. The purpose of the locked keyboard is to prevent changing the data on the screen until the program permits it. This allows more reliable applications to be written. The keyboard lock condition will persist until specifically cleared by the application program with an explicit write of an "Esc b" sequence. In addition, you can re-enable the keyboard by performing a soft terminal reset, if necessary.

The auto-home bit is effective only on user initiated transfers (when you press the ENTER key). It has no effect on program initiated transfers (the program sends "Esc d" to simulate the ENTER key), because such transfers do not use the terminal's long handshake mode. Refer to Appendix A, "Example Protocol Charts" and Appendix C, "Sample Page Mode Applications" for details on record blocking and escape sequences.

## Special Status Read

```
CALL EXEC(1,ior(lu,3700b),status_buffer,length)
```

This read request reads the driver configuration and status words. It can be used to determine which driver is in use. The call is identified as special because the function bits are 37B and the length is 32 words (or 64 bytes). Table 2-2 contains a summary of the contents of each word.

Note that this call will I/O suspend if a read is pending or if the LU has been downed.

Words 1-3 contain the device driver name in ASCII. For example, DV800 or DV801.

Word 4 contains the revision code of the device driver. The left byte contains the major cycle revision level, and the right byte the minor. For example, Revision 12.62 would be stored as 006076B.

Word 5 is the current driver type from the EQT/DVT tables and may be different from the device driver type because of the effects of a CN 30 call.

Word 6 contains the address of the EQT/DVT. The formats of the I/O Tables are documented in the *RTE Operating System Driver Writing Manual,* part number 92200-93005.

Words 7-10 are always blank in RTE-6/VM but return the interface driver name and revision code in RTE-A.

Word 11 is the firmware revision code in the major/minor format. This is non-zero only for the intelligent I/O cards (8-channel MUX).

Words 12-14 contain the name of the interrupt program. If scheduling is currently enabled, the sixth byte is “E”, otherwise it is “D”.

Words 15-17 are always blank in RTE-6/VM but return the name of the secondary interrupt program in RTE-A.

Words 18-23 are the optional parameters required in the respective control calls to set the driver to the current conditions.

Words 24-31 are reserved.

Word 32 returns the driver communication word. This word contains the following:

| | |
|---|---|
| bit 6 | Reserved for scheduling HpMdm. |
| 7 | Next timeout is expected (line down delay and so forth). |
| 8 | Internal multi-buffer bit. |
| 11 | Program needs to be scheduled. |
| 13 | Terminal in page mode. |
| 14 | Terminal in block mode. |
| 15 | Last call was to a slave device. |

**Table 2-2. Special Status Read Words**

| | |
|---|---|
| 1 | Device driver name |
| 2 | ” |
| 3 | ” |
| 4 | Device driver revision code |
| 5 | DVT word 6 |
| 6 | DVT address |
| 7 | Interface driver name |
| 8 | ” |
| 9 | ” |
| 10 | Interface driver revision code |
| 11 | Firmware revision code |
| 12 | Primary interrupt program name |
| 13 | ” |
| 14 | ” |
| 15 | Secondary interrupt program name |
| 16 | ” |
| 17 | ” |
| 18 | Echo back of CN 17 parameter |
| 19 | Echo back of CN 22 parameter |
| 20 | Echo back of CN 30 parameter |
| 21 | Echo back of CN 31 parameter |
| 22 | Echo back of CN 33 parameter |
| 23 | Echo back of CN 34 parameter |
| 24..31 | Reserved |
| 32 | Echo back of DVT 20 |

# Write Request

The calling sequence for a write request is:

```
CALL EXEC(2,cntwd,bufr,bufln[,pram3,pram4])
```

## BUFR and BUFLN

The *bufr* parameter is the user's buffer containing the data to be written to the selected LU. *bufr* cannot be a FORTRAN character data type. Refer to the HpCrtSendChar subroutine section in the *RTE-A • RTE-6/VM Relocatable Libraries Reference Manual*, part number 92077-90037.

The *bufln* parameter is the length of the data in the user buffer. If the length parameter is positive, it indicates the number of words to be transferred (0 to 32767 words, 0..077777B). If the length parameter is negative, it indicates the number of characters to be transferred (1 to 32768 bytes, 177777B..100000B). *bufln* set to zero produces a CRLF sequence in ASCII mode; in binary mode, the write is suppressed. The trailing carriage return/line feed combination that is expected by most terminals is supplied by the driver, so the user buffer need contain only the data characters. If this action is not desired, it can be suppressed by setting the TR bit in *cntwd*.

## CNTWD (Write Request Control Word)

The write request control word (*cntwd*), has the following form:

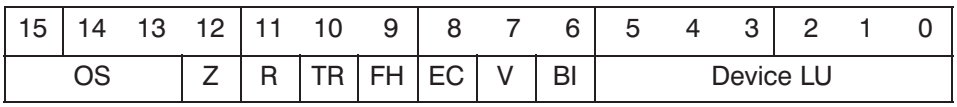



**Figure 2-2. Write Request Control Word (CNTWD)**

**OS**: Operating system specific bits; see the *RTE/6-VM Programmer's Reference Manual*, part number 92084-90005.

**Z**: Write/write bit. If clear, PRAM3 and PRAM4 of the call are ignored.

If set, the control buffer indicated by PRAM3 and PRAM4 are written to the external device in ASCII mode before the user buffer is written, as described in the Read Request section. The normal data buffer, described by BUFR and BUFLN, is transferred according to the rules established by the remainder of the bits in the control word.

**R**: Reserved; set to 0.

**TR**: Transparency bit. Used to alter the normal addition of a trailing Carriage Return/Line Feed (CRLF) to the user data. If the transparency bit is clear, then the characters in the user buffer are sent to the given LU without special processing, except for the last character. If it is an underscore (octal 137), then the underscore itself and the trailing CRLF are suppressed. In other words, for a request length of N characters where the Nth character is an underscore, N minus 1 characters are transferred. If the Nth character is not an underscore, N plus 2 characters are transmitted, the extra 2 characters being a trailing CRLF.

If the transparency bit is set, the special processing described above does not occur. Thus, exactly N characters are transferred with no additions or deletions.

**FH**: Force handshake bit. When in HP protocol mode, this bit causes an ENQ/ACK handshake to be executed before the user data buffer transfer begins.

Refer to the Example Protocol Charts appendix for an example of the use of the FH and BI bits in a write/write call to write to the graphics memory of an HP terminal.

**EC**: Echo bit. The echo bit is ignored on writes.

**V**: Printer Honesty mode bit. This bit is ignored for ports that are not configured as printers. If set, column 1 has no special meaning. If clear, column 1 is used for FORTRAN style carriage control.

**BI**: Binary bit. Inhibits HP protocol for this write. Causes the buffer described by BUFR and BUFLN to be transferred without the ENQ/ACK handshakes that would normally occur after each 80 bytes of output. If over 80 bytes are transmitted, then an ENQ/ACK handshake will be pending at the beginning of the next write. ENQ/ACK handshakes may be removed permanently with the appropriate change of protocol using a CN 34 call.

Refer to the Example Protocol Charts appendix for an example of the use of the FH and BI bits in a write/write call to write to the graphics memory of an HP terminal.

# Control Request

The calling sequence for a control request is

```
CALL EXEC(3,cntwd[,pram1])
```

## CNTWD (Control Request Control Word)

The control request control word (*cntwd*) is shown in Figure 2-3.

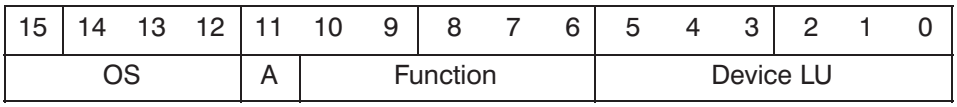



**Figure 2-3. Control Request Control Word (CNTWD)**

**OS:** these bits are defined in the *RTE-6/VM Programmer's Reference Manual*, part number 92084-90005.

**A:** RTE-6/VM ignores bit 11. This bit is used in RTE-A function calls.

**Function:** contains the function code that defines the control action to be performed. Function control codes are referred to by the octal value, so the five bits allow for control codes 00 to 37 (octal).

The following lists the control functions.

| Code | Function |
|---|---|
| 06B | Obtain dynamic status |
| 11B | Line spacing/page eject |
| 16B | Define baud rate group |
| 17B | Define terminator |
| 20B | Define/enable primary program scheduling |
| 21B | Disable program scheduling |
| 22B | Set device timeout |
| 25B | Read HP terminal straps into driver |
| 26B | Flush input buffers |
| 30B | Set port ID |
| 31B | Modem environment |
| 32B | Generate break |
| 33B | Configure driver responses |
| 34B | Set port protocol |
| 35B | Reset BRG (ID800/01 only) |

If function codes other than those listed above are issued, the driver rejects the call as an illegal request, or accepts it with no action, depending on the state of the "nice" bit set in the CN 34 call.

**Device LU:** contains the LU number to control.

## Function Code 6B: Dynamic Status

```
CALL EXEC(3,ior(lu,600b)[,pram1])
CALL ABREG(istatus,itlog)
CALL RMPAR(istat_array)
```

Function code 6B returns the requested port status in EQT word 5 and in the A-Register. The standard form of the EXEC 3 call is *pram1* set equal to zero or omitted. *pram1* can be set to a non-zero value to return other information. Refer to Dynamic Status Special Forms, later in this section.

The ABREG routine, documented in the *RTE-A • RTE-6/VM Relocatable Libraries Reference Manual*, part number 92077-90037, can be used to retrieve the contents of the A-Register.

The RMPAR routine, documented in the *RTE-6/VM Programmer's Reference Manual*, part number 92084-90005, can be used to retrieve extended status. Five words of status are returned in the five-word array *istat_array*. The format of *istat_array* is as follows:

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Word 1 | AV | | | | Device Type | | | | EF | BR | EM | LD | OF | PF | TO | E |
| Word 2 | 0 | | | | | | | | | | | | | | | |
| Word 3 | Length of Type Ahead Data Available | | | | | | | | | | | | | | | |
| Word 4 | 0 | | | | | | | | | | | | | | | |
| Word 5 | 0 | | | | Driver Revision Code | | | | | | | | | | | |

### EQT 5: Device Status

This status word is returned in the A-Register after unbuffered requests. The A-Register can be accessed from high level languages via an ABREG call. The status field in EQT 5 is updated by each call to the driver. The format of the status returned in EQT 5 is as follows:

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EQT 5 (A-Reg) | AV | | | | Device Type | | | | EF | BR | EM | LD | OF | PF | TO | E |

**Figure 2-4. Dynamic Status Control Word**

**AV:** Device availability. These bits are used by the system for I/O control. The DS operator command can also be used to examine the availability.

- If AV is 00 — the EQT is available for a new request to be initiated (the device is free to process a new request).
- If AV is 01 — the associated device has been set down by the driver or the operator. New requests will be suspended on the downed device.
- If AV is 10 — the device is busy processing an I/O request. New requests may be pending (that is, linked through word 2 of the EQT).
- If AV is 11 — the device is down, but busy with a request (such as an abort request).

**Device Type:** Driver device type. This 6-bit value describes the type of device associated with the current EQT. All device type values are initially established at generation. Refer to Table 2-5 for device type values.

**EF:** End of file (set by CTU drivers only).

**BR:** Break character detected on received data line.

**EM:** End of medium (set by detection of EOT in Normal ASCII read).

**LD:** Line down; valid for modem lines after first connect. Can be set by speed sense failures also.

**OF:** Overflow error; the application is losing data.

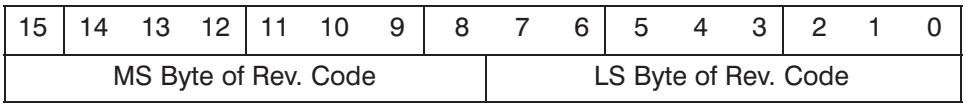
**PF:** Parity or framing error; there was a bit error in the data.

**TO:** Timeout by the device driver.

**E:** Error.

### Dynamic Status Special Forms

To read the firmware revision code, the dynamic status call optional parameter can be set to −1. The format of *istatus* (the A-Register) is then:




When the optional parameter is set to −2, the driver returns the constant 123456B in the B-Register for use in distinguishing Revision 5000 drivers from the previous drivers.

When the optional parameter is positive, the corresponding MUX memory location is read.

## Function Code 11B: Line Spacing/Page Eject

This control function emits a specified number of line feeds or performs a page eject, depending upon the call parameter.

```
CALL EXEC(3,ior(lu,1100b)[,NumLines])
```

emits the number of line feeds specified by a positive parameter. The default is one line. Some drivers limit *NumLines* to 128 line feeds.

```
CALL EXEC(3,ior(lu,1100b),-1)
CALL EXEC(3,ior(lu,1100b),-2)
```

Either of the above calls performs a page eject, but only on hardcopy devices as defined by the H bit in the CN 34 call. The effect on a terminal is to skip only one line. When the parameter is −1, the form feed is conditional. A parameter of −2 performs an unconditional form feed. If backward compatibility is needed, all form feeds can be made unconditional by setting bit 10 in the CN 34 call.

## Function Code 16B: Define Baud Rate Group

Function code 16B applies only to DV800. This control call reconfigures the BRG (baud rate group) to a range other than the predefined BRG setting (see CN 30 for the default setting). This control request should be used only if the desired BRG range cannot be obtained from the CN 30 control call, and it must be used before any ports on a given BRG are initialized. If ports are already initialized on the same BRG, then the BRG must be reset before the CN 16 can be executed. See the CN 35 command to reset a BRG.

```
CALL  EXEC(3,ior(LU,1600b),BrgRange)
```

sets either baud rate group to a preconfigured setting for the CN 30 control code.

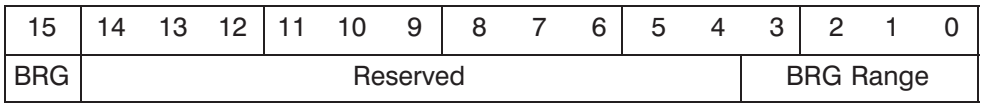



**Figure 2-5. Define Baud Rate Group**

**BRG:** 0 Set Baud Rate Group 0
1 Set Baud Rate Group 1

The assignment of ports to a Baud Rate Group (BRG) is determined by jumpers in the cable connector hood.

**BRG Range:** These bits set a given baud rate group to one of the following ranges:

| BRG Range | Possible Baud Rates | | | |
|---|---|---|---|---|
| 0B | [ | Reserved | | ] |
| 1B | [ | 1800 * | −− | −− ] |
| 2B | [ | 134 * | −− | −− ] |
| 3B | [ | 14,400 | −− | −− ] |
| 4B | [ | 9600 | 19,200 | 38,400 * ] |
| 5B | [ | 4800 * | 9600 * | 19,200 * ] |
| 6B | [ | 2400 | 4800 | 9600 ] |
| 7B | [ | 1200 | 2400 * | 4800 ] |
| 10B | [ | 300 * | 600 * | 1200 * ] |
| 11B | [ | 75 * | 150 * | 300 ] |
| 12B | [ | −− | −− | 110 * ] |
| 13B | [ | 600 * | −− | −− ] |

* If you specify a given baud rate using the CN 30B command, and the BRG is not assigned, the starred range is selected. For example, if you specify 9600 baud, range 5B is selected.

## Function Code 17B: Define Termination Character

```
CALL EXEC(3,ior(lu,1700b),Terminator)
```

You have the option to redefine the terminator used in all subsequent transparent ASCII reads.

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | S | E | Terminator | | | | | | |

**Figure 2-6. Definable Terminator**

**Reserved:** Reserved, should be set to 0.

**S:** Schedule bit.

0 Does not enable scheduling.

1 Enable scheduling upon receipt of the terminator character (bits 0 through 7), if FIFO mode is enabled and per-character scheduling is not enabled (the CN 30 parameter is equal to 100000B).

**E:** Enable bit.

0 Transparent ASCII reads are terminated by a CR, as described in the "Read Request" section.

1 All subsequent transparent ASCII reads terminate upon receipt of the character specified in bits 7 through 0. This also disables the CRLF that is normally output upon receipt of the terminator of an ASCII read.

**Terminator:** Termination character. These bits define the new transparent ASCII read termination character.

You can use the CN 17 to set up transparent ASCII reads that do not generate a CRLF. If a CRLF is needed, execute the call with a parameter of 415B to set the termination character to 15B (Carriage Return).

## Function Code 20B: Enable Program Scheduling

This control call enables program scheduling (see Figure 2-7) upon recognition of certain interrupting conditions. Those conditions vary depending upon which driver is in use, as explained below.

### Program Scheduling Conditions

DVC00 performs program scheduling upon receipt of an unsolicited character. DV800 performs program scheduling depending on the CN 33 selection.

An unsolicited character is any character that arrives when the driver is not executing a read. Some driver/interface combinations have FIFO buffer modes that effectively leave a read

pending at all times. When this mode is invoked, the BREAK key is a more reliable way to schedule the prompt program. The BREAK key is a key present on most keyboards that causes a long space condition to be sent on the communications line, approximately 250 milliseconds. BREAK is a condition on the line, not a character. BREAK is always recognized, even if a read is pending.

When the program is scheduled, the B-Register is set to the EQT 4 address.

### Pass Program Name

To alter which program is scheduled upon interrupt, the following call can be used:

```
CALL EXEC(3,ior(lu,2000b),index)
```

This call passes the index to an internal table of program names in the optional parameters.

To enable a program that is already known, either from the system generation or from a call to HpCrtSchedProg, the following form of the EXEC call can be used:

```
CALL EXEC(3,ior(lu,2000B))
```

### RTE Compatibility: Pass Program Name

To facilitate portability between operating systems, it is suggested that the utility subroutine HpCrtSchedProg in HpCrt.lib be used in place of the CN 20 calls.

```
CALL HpCrtSchedProg(lu,5hPRMRY[,2hPR])   or   HpCrtSchedProd_S
```

The schedule table for the MUX driver in RTE-6/VM is located in the system tables area with the following format:

```
$sctb Dec NumberOfWords
       asc 2, PR
       asc 2, MP
       asc 2, T
```

The first word is the number of words allowed by the user at generation time. The minimum default value is five words, which allows for PRMPT and one other name. The table size can be increased by changing the second parameter when replying to the question "`# OF LU MAPPINGS?`" in the answer file. Each additional name requires 2.5 words, rounded up to an integer. For example, allowing 5 names in the table increases the table size to 14 (2.5 x 5 plus 1 for the length).

Refer to the System Generation Considerations appendix for details on how programs can be specified at system generation.

## Function Code 21B: Disable Program Scheduling

```
CALL EXEC(3,ior(lu,2100B))
```

This call disables program scheduling enabled by function code 20B. It does not change the name of the interrupt schedule program, so a subsequent function 20B call with no parameters re-enables the same program. Refer to Figure 2-7 for a state diagram of CN 20 and 21.

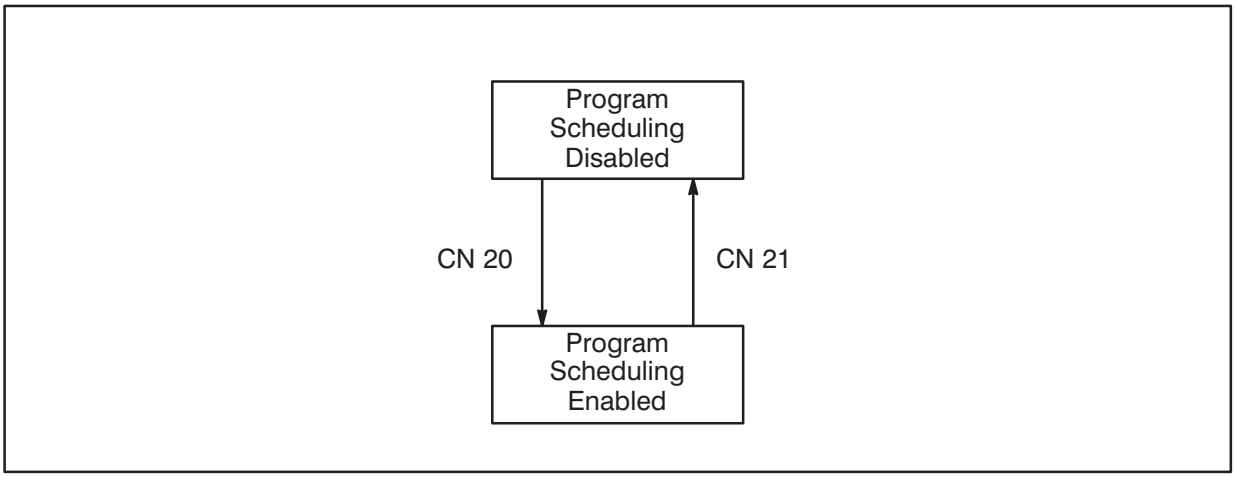



**Figure 2-7. Program Scheduling**

## Function Code 22B: Set Device Timeout

```
CALL EXEC(3,ior(lu,2200B),NewTimeOut)
CALL ABREG(dummy,OldTimeOut)
```

This call changes the value of the device timeout. *NewTimeOut* is the number of centiseconds (100ths of a second) allowed for a read or a write to complete. If a nonzero value is passed, it is complemented and then used as a count-up timer. If *NewTimeOut* is set to 1, the device timeout equals .01 seconds. Likewise, setting *NewTimeOut* to 32767 sets the device timeout to 327.67 seconds. Longer timeouts are possible, but they appear odd when expressed in decimal because of the lack of an unsigned integer data type. Numbers from −32768 through −1 produce timeouts in the range 327.68 seconds to 655.35 seconds.

A timeout value of 0 disables timeout processing, giving an infinite timeout. *This is not recommended.* Because read calls are blocking I/O, messages cannot be written to a terminal that has a read outstanding. In addition, a timeout ensures that the driver is allowed to detect and correct certain error conditions. When a read call times out, the driver returns the status word, with bit 1 set, to the program. The status word is available in the A-Register, which can be accessed from high level languages via the ABREG call. All interactive programs should check this bit and take appropriate action upon timeout (usually they should loop to re-prompt the user).

This control call has a similar effect as the TO operator command:

```
CI> TO,lu,NewTimeOut
```

except that all values are accepted. The TO command does not allow values 1 through 499 on a terminal.

## Function Code 23B: Flush Output

This call is supported only by DVC00 and is available for backward compatibility.

When the driver receives this call, it ignores all further action requests until one of the following happens:

- The queue on the EQT is empty.
- An input request is received.
- A restore control request (CN 24) is received.

This call is used with hardcopy devices such as TTYs that get backlogged with output requests that are buffered by the system. If a high priority program (such as CI) issues this request, all pending output is dumped and the terminal is then ready for the next input or other action.

## Function Code 24B: Restore Output Processing

This call is supported only by DVC00 and is available for backward compatibility.

This call terminates the CN 23 prior to a full flush. It is intended for use with programs that do not want their output flushed by a CN 23.

## Function Code 25B: Read HP Terminal Straps

```
CALL EXEC(3,ior(lu,2500B))
```

This call must be issued each time the operating mode of an HP terminal is changed, to allow the driver to update the internal flags that indicate what protocol to use in a read call. To make this easier, the utility subroutines HpCrtPageMode, HpCrtLineMode, and HpCrtCharMode are available. Note that this EXEC call clears FIFO buffers.

## Function Code 26B: Flush Input Buffers

```
CALL EXEC(3,ior(lu,2600B))
```

This function clears the input buffers of the interface cards. If FIFO buffer mode is enabled, it clears any data that is being held on the multiplexer cards. This has the same effect as re-issuing the “enter FIFO buffer mode” command. For all drivers that support FIFO mode, this call clears the hardware and software error bits.

## Function Code 30B: Set Port ID

```
CALL EXEC(3,ior(lu,3000B),Portword)
```

This function establishes the logical connection between the LU and the physical device connected to the card. It is used by the multiplexer drivers to map LUs to the ports and to set the baud rate on cards when possible.

### DVC00

The *Portword* parameter must be set to 0 for DVC00. DVC00 supports only:

- 8 bits per character
- hardware selected baud rate
- 1 or 2 stop bits, set by a hardware jumper on the HP 12966A BACI card.
- no parity

If *Portword* is set to a non-zero value, DVC00 will reject the control call with an error 15B return.

### DV800

All configuration parameters can be specified at generation. The values of the EQT extension words 1 through 4 initialize the CN 17, CN 20, CN 30, CN 33, and CN 34 parameters for the MUX. These values may be changed in the welcome file or at any time.

The specified configuration occurs on the first request to the interface driver (such as a read or write request) on a per LU basis. Note that this configuration does not occur when it is initiated from the remote device (such as a carriage return before initializing the port). In the welcome file, ports may be initialized by a "CN *lu* 6B", "CN *lu* 11B", or a write request to the particular port.

---

**Note** Function code 30B must be issued before any other request is sent to a specified port. Any requests sent to an LU prior to function code 30B are ignored by the driver, with the exception that CN 6 requests are processed, to aid in identifying the drivers (see CN 6,−2 option). Also, the LU number given should specify a unique device on the MUX. If the request is invalid, or if a conflict exists, an error is issued and the request rejected. The driver parameters can be set so that the port is preconfigured (refer to the System Generation Considerations appendix).

---

The *Portword* parameter for DV800 has the following format:

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CF | | M | x | SB | | | P | | R | S | | | PO | | |

**CF:** Character Framing bits. These bits indicate the number of data bits per character as follows:

| 15 | 14 | No. of Bits/Char |
|---|---|---|
| 0 | 0 | 8 bits |
| 0 | 1 | 7 bits |
| 1 | 0 | 6 bits |
| 1 | 1 | 5 bits |

When using 5-bit data, the upper three bits must be set to zero. Otherwise, undesirable results may occur due to limitations in the 8-channel MUX hardware.

**M:** Modem control bit. Modem control is not supported on the RTE-6/VM MUX.

**x:** This bit is not set by the user; it is returned by the MUX driver. This field was used by previous drivers to inform the driver and the firmware of the distribution of the two baud rate clocks to the UARTs on the 8-channel MUX cards. The wiring in the connector hood determines this bit. There are several different cables for the MUX cards, with different wiring arrangements. It is also possible that the wiring may have been changed in the field. To alleviate this problem, DV800 determines the BRG wiring during initialization.

The sensed value is returned in this bit as follows (any user-supplied value is ignored):

bit 12 = 0 : generator 0
bit 12 = 1 : generator 1

**SB:** Stop bit selection bits. These bits indicate the number of stop bits as follows:

| 11 | 10 | No. of Stop Bits |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 2 |
| 1 | 0 | 1.5 |
| 1 | 1 | (reserved) |

**P:** Parity checking bits. If parity checking hardware is not available on a card, it can be generated or checked in the user buffer by use of the subroutines HpCrtParity_Gen and HpCrtParity_Chk. Parity checking is *not* disabled by binary reads. If parity checking is enabled, it is the programmer's responsibility to check the parity bit in the status. Standard HP utility programs, such as CI or FMGR, do not check for parity error.

| 9 | 8 | |
|---|---|---|
| 0 | 0 | no parity |
| 0 | 1 | no parity |
| 1 | 0 | odd parity |
| 1 | 1 | even parity |

**R:** Reserved; set to 0. If this bit is set, the driver reports a configuration error.

**S:** Speed, baud rate selection bits. These bits select the baud rate as follows:

| Bits 6..3 | |
|---|---|
| 00b (0000) | n/a (DV800 will reject control call with error 15B return) |
| 01b (0001) | 600 baud |
| 02b (0010) | 75 baud |
| 03b (0011) | 110 baud |
| 04b (0100) | 134.5 baud |
| 05b (0101) | 150 baud |
| 06b (0110) | 300 baud |
| 07b (0111) | 1200 baud |
| 10b (1000) | 1800 baud |
| 11b (1001) | 2400 baud |
| 12b (1010) | 4800 baud |
| 13b (1011) | 9600 baud |
| 14b (1100) | 19.2K baud |
| 15b (1101) | 38.4K baud * |
| 16b (1110) | 14.4K baud * |
| 17b (1111) | speed sense |

* Because RS-232 connections have a recommended upper frequency limit of 19.2K baud, the RS-422 circuitry on the cards should be used for all settings higher than 19.2K.

**PO:** Port number bits.

| 2 | 1 | 0 | |
|---|---|---|---|
| 0 | 0 | 0 | port 0 |
| 0 | 0 | 1 | port 1 |
| 0 | 1 | 0 | port 2 |
| 0 | 1 | 1 | port 3 |
| 1 | 0 | 0 | port 4 |
| 1 | 0 | 1 | port 5 |
| 1 | 1 | 0 | port 6 |
| 1 | 1 | 1 | port 7 |

## Default BRG Ranges (DV800 Only)

There are interactions between channels that are in the same baud rate group on the 8-channel MUX. The baud rates of all channels on a BRG must have baud rates in 1, 1/2, and 1/4 ratios. For example, if the highest baud rate on BRG 1 is 9600 baud, then 4800 and 2400 baud are also available for use by other ports in the same group. If 19.2K is the highest, then 9600 and 4800 are available, and so on.

The BRG range is determined by the baud rate of the first port accessed on the BRG. If this automatic range selection is not desired, the CN 16 call may be used. Refer to the Function Code 16B section in this chapter.

For example, ports 4 and 5 are on the same BRG and require 4800 and 2400 baud, respectively. If port 5 is set to 1200 baud first, then port 4 cannot be set to 4800 baud, because the BRG range was automatically set to [ 300 −−− 1200 ]. This problem can be overcome by first setting the BRG range to [ 1200 2400 4800 ] using the CN 16 control code. It will not necessarily be

corrected by using speed sensing. This will cause the BRG range to be determined by the first port whose speed is sensed. Table 2-3 lists the BRG ranges.

A baud rate selection of 17B causes the port to perform speed sensing. This allows the user's terminal to determine the appropriate baud rate. If the port is enabled for HP protocol, the driver sends an ENQ at each available baud rate, then waits half a second for an ACK to be returned.

When an ACK is received correctly, the baud rate is then known, and the driver sends a carriage return to the terminal. Other than random characters that may appear on the screen as a result of the terminal receiving the ENQ characters at the wrong baud rate, the process is automatic and the user is not aware that it is occurring.

If the port is configured for non-HP protocol, the user must supply CR characters until the interface card can detect the appropriate baud rate. This may require several tries, as the card is sampling at 1-second intervals. When the baud rate is detected, a CRLF is echoed to the terminal.

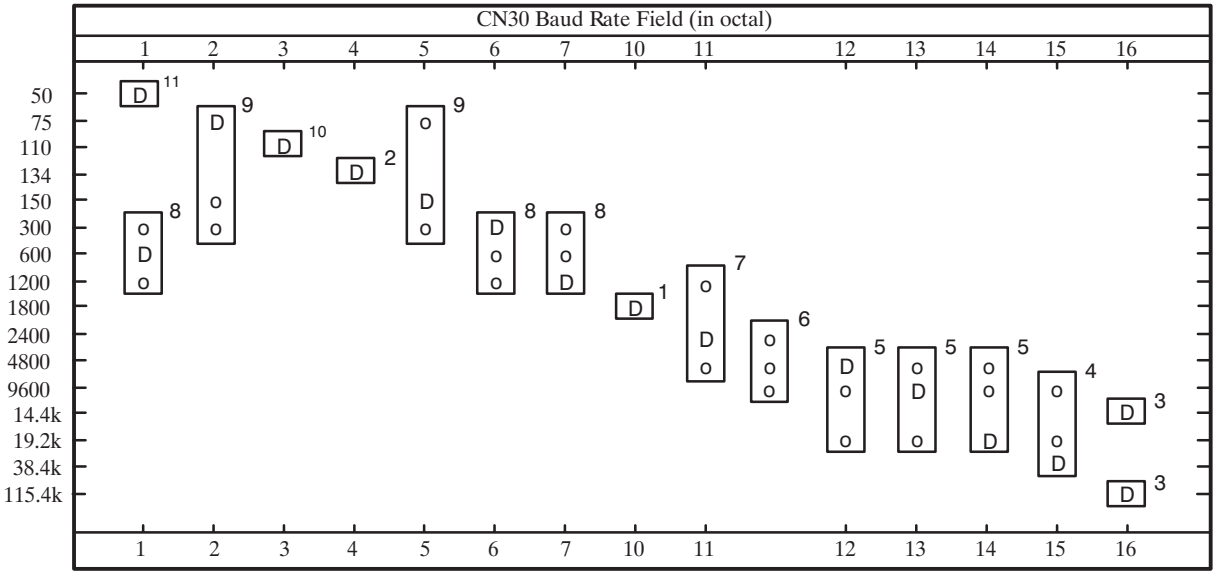
The new baud rate can be read by using the special status read described earlier in this chapter. If the card is unable to detect or match the baud rate within the read timeout, the baud rate is left at 17B and the line down bit is set in EQT 5. (The programmer should ensure that the timeout is set to a reasonable value, such as two minutes, before issuing the speed sense command.) The programmer can detect speed sense failure from the CN 30 field of a special status read. Until the speed sense failure is corrected by another CN 30 call, all other requests are rejected. Asynchronous interrupts and breaks will cause speed sensing to occur again.

---

**Note** The baud rates that are possible are limited by the hardware configuration of the cards (and in some cases the interface cable hoods) plus any modems, line drivers, and so on, in the communications line, and the destination equipment. To operate at high baud rates, the CRT and the driver must use a handshake protocol. Refer to the Function Code 34B section that follows.

---

**Table 2-3. BRG Ranges**




Note: The boxes enclose baud rates that are generated by the same BRG setting. The number to the upper right of each box is the BRG range selection (in decimal) for use in the CN 16 call. Within a given box, the default baud rate is shown by a 'D'. The other baud rates that are available with the BRG set to the given value are shown by 'o'.

For example, if the first port configured is set to 9600 baud by the CN 30 call (CN *lu* 30B 13), the default BRG selection will be 5. Once that port is configured, the other ports that share the BRG would be restricted to 4800, 9600, or 19.2K baud.

This is a reasonable default in most cases. However, assume you have 2400 baud terminals that must be connected to the same MUX. A CN 16 call could be included in the Welcome file (before the first port is configured) to force the BRG range to 6. This would allow the ports driven by that BRG to be configured to 2400, 4800, or 9600 baud instead.

Prior to Revision 5.19 of the MUX firmware, 50 baud rate and 115.4k baud were available as selections 1 and 16, respectively. At Revision 5.19, these were changed to produce 600 and 14.4k baud instead.

## Function Code 32B: Generate Break

Function code 32B allows for programmatic breaks. This function applies only to DV800.

```
CALL  EXEC(3,ior(LU,3200b)[,time])
```

where: *time* specifies the break duration. If *time* is equal to 0 or omitted, the transmit data line is held low for approximately 250 milliseconds.

If *time* is not equal to zero, it specifies the number of "character times" for the break duration. "character times" is the time required to transmit one character at the current BAUD rate and framing. Values from 1 to 254 are legal.

## Function Code 33B: FIFO Buffer Mode Control

FIFO buffer mode is available only on DV800. The FIFO is 1024 characters per port on the 8-channel MUX.

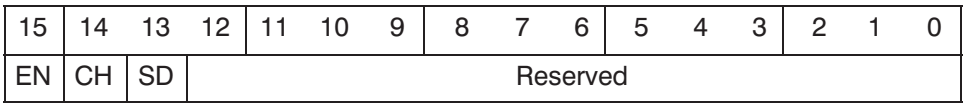



**Figure 2-8. FIFO Buffer Mode Control**

**EN:** FIFO mode enable.

0 No FIFO buffering. System attention is gained by pressing any key when a read is not pending. The remaining bits are ignored if bit 15 is not set.

1 Enable FIFO buffering. If data is received without a pending read, it is saved until the next read is posted, unless more data is received than can be held on the card.

**CH:** Character-by-character scheduling.

0 Accumulate characters, but take no action until a read is posted or a break is detected.

1 Enable program scheduling for each character that is received.

**SD:** Save data on break.

0 Clear the buffer before scheduling the interrupt program when a break is detected.

1 Keep the FIFO buffer data, even when a break is detected.

**Reserved:** Reserved; set to 0.

A buffered read mode can be implemented for “single character” style editors by the algorithm illustrated in Figure 2-9. The following is a step-by-step representation of the algorithm. It is a useful technique when you need full duplex I/O.

1 Enable FIFO mode.
2 Set timeout to a value short enough to satisfy the response time constraints.
3 Perform a dynamic status call (CN 6) to see if any characters are in the FIFO.
4a If characters are available, post a read for that many characters; process the data.
4b If no characters are available, post a read for one character.
5 When the read completes, check for a timeout.
6a If timeout, go to step 7.
6b If no timeout, 1 or more characters have been received; process the data.
7 See if outbound data is available.
8 If data is available, change timeout to a large enough value to allow the write to complete; write the data.
9 Perform any other processing necessary.
10 Go to step 2.

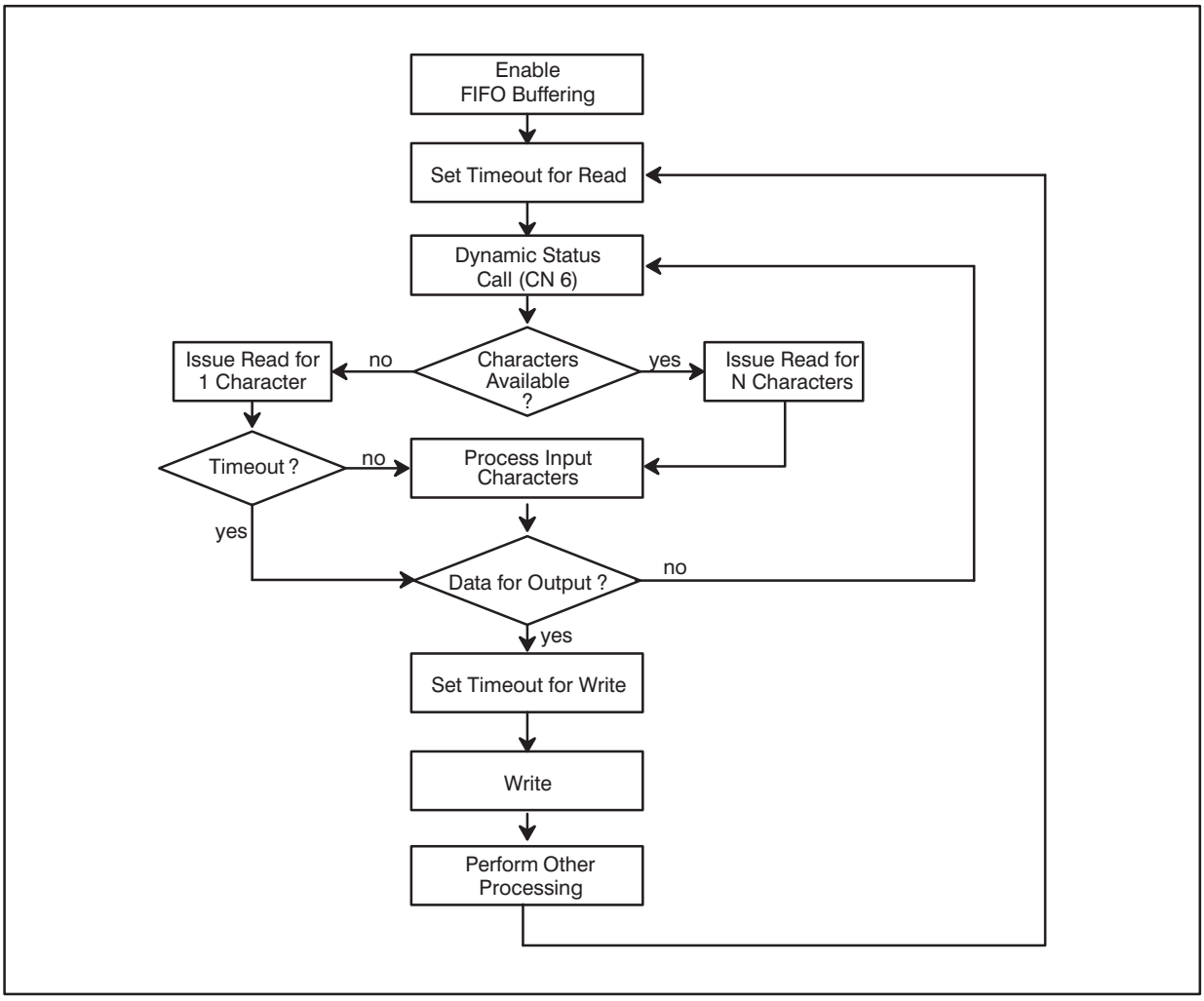



**Figure 2-9. Buffered Read Mode Algorithm**

## Function Code 34B: Set Port Protocol

```
CALL EXEC(3,ior(lu,3400b),Protocol_Word)
```

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | | | N | H | U | RD | P | Protocol | | | | | | | |

**R:** Reserved; set to zero.

**N:** Nice bit.

0 Unimplemented control requests are rejected.

1 Unimplemented control requests are accepted but ignored.

**H:** Hardcopy/spacing mode.

0 The port is connected to a CRT.

1 The port is connected to a hardcopy device. This enables page eject processing.

**U:** Unconditional page ejects.

0 CN11,−1 page ejects are conditional.

1 CN11,−1 page ejects are unconditional.

**RD:** Return data on timeout.

---

**Caution** The RD Bit (bit 9) should not be set for interactive ports, as it leaves the possibility that destructive commands could be given to the system if the user inadvertently allows the terminal to timeout in the middle of a command. For example, if the user intended to enter the command “PU,@.lst”, but was called away from the terminal, it could be seen as “PU,@”.

---

0 No data is returned, transmission log is zero.

1 When a timeout is detected, the transmission log will indicate the number of data bytes in the buffer.

**P:** Printer/paper position.

0 Port is connected to an interactive terminal.

1 Port is connected to a printer and is, therefore, not interactive.

This bit changes the meaning of the H bit (bit 11) to be 0 = post spacing and 1 = pre-spacing. Carriage control is a matrix of capabilities defined by bits 8, 11, and 10 as described in Table 2-4.

**Table 2-4. Carriage Control Capabilities**

| Bit 8 (P-bit) | Bit 11 (H-bit) | Bit 10 (U-bit) | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | x | CRT | PRMPT | No CCTL | No CN 11s |
| 0 | 1 | 0 | TTY | PRMPT | No CCTL | Conditional CN 11s |
| 0 | 1 | 1 | TTY | PRMPT | No CCTL | Unconditional CN 11s |
| 1 | 0 | 0 | printer | no PRMPT | post-spacing CCTL | Conditional CN 11s |
| 1 | 0 | 1 | printer | no PRMPT | post-spacing CCTL | Unconditional CN 11s |
| 1 | 1 | 0 | printer | no PRMPT | pre-spacing CCTL | Conditional CN 11s |
| 1 | 1 | 1 | printer | no PRMPT | pre-spacing CCTL | Unconditional CN 11s |

CCTL means carriage control via column 1 as used by the FORTRAN formatter.
CCTL can be disabled by bit 7 of the CNTWD in the write call.

In addition to the above, bits 8, 11, and 1 determine the driver device type as described in Table 2-5.

**Table 2-5. Driver Device Types**

| Bit 8 (P-bit) | Bit 11 (H-bit) | Bit 1 | Type | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | non-HP CRT |
| 0 | 0 | 1 | 5 | HP CRT |
| 0 | 1 | 0 | 6 | non-HP hardcopy terminal (TTY) |
| 0 | 1 | 1 | 6 | HP hardcopy terminal (HP 2635) |
| 1 | x | 0 | 12 | printer that does not use ENQ/ACK protocol |
| 1 | x | 1 | 12 | printer that uses ENQ/ACK protocol |

**Protocol:** Bits 7 through 0 define the protocol as follows:

| Bits 7 thru 0 | DVC00 | DV800 |
|---|---|---|
| 000b: TTY | ok | ok |
| 001b: Xon/Xoff | N/A | ok |
| 002b: HP | N/A | ok |
| 003b: HP Xon/Xoff | N/A | ok |
| 004b: CPU-to-CPU | N/A | ok |
| 202b: Half HP | N/A | ok |
| 203b: Half HP Xon | N/A | ok |

The protocols are described in the paragraphs that follow.

---

**Caution** The protocols listed above are those supported by Hewlett-Packard. They provide a solution for most configurations. Other bit combinations are possible, but may cause unpredictable and undesired results.

---

The driver type in the DVT is changed to reflect the protocol selection. Driver type 5 is used to indicate that block mode operation is possible. Therefore, the driver type is not changed to 5 if the hardcopy bit is set. Driver type 6 is assigned for interactive hardcopy terminals. Driver type 12 is assigned for printers.

Reissuing the protocol command causes the port to enter the known state of transmit enable, for recovery from error situations where the CPU is waiting for an Xon from a device that will not send one.

The protocols available are:

TTY PROTOCOL. No flow control handshaking, normal CRLF processing. This mode is primarily for interactive use with "dumb" terminals.

Xon/Xoff. This bi-directional protocol allows both the port and external devices to issue Xon and Xoff characters to pace the flow of incoming data. When the receiving end approaches a buffer full condition, the Xoff character is transmitted to suspend the flow of data. When buffer space is again available, the Xon character is transmitted to resume data flow. Excess Xon characters are discarded by the port in this mode, thus it is *not possible* to receive Xon characters as part of the data record.

It is possible to get into deadlock situations with this protocol. For example, assume that the port is connected to a line printer and the line printer runs out of paper. When the paper-out condition is detected, the printer transmits an Xoff to the CPU to stop the flow of data. If the user turns off the printer to reload paper, as is required by some printers, then turns it back on, expecting data to be printed, nothing will happen. The CPU is waiting for an Xon to give it permission to proceed.

Some printers have a front panel button that causes an Xon to be sent, while others send an Xon every time they are placed online. However, most do not have either feature. Because of this, if the port times out (beware of zero timeouts) during an Xon/Xoff write, the driver internally enables itself to the Xon state, and then goes down (but only if the hardcopy bit is set).

In FIFO buffer mode, the interface card sends an Xoff when 15 bytes of buffer storage remain. A second Xoff is sent when only 10 bytes remain. An Xon is sent when the buffer level drops below 20 characters available.

The MUX card suspends output when it receives an Xoff . If it then receives a BREAK (which usually indicates that the user wants to schedule the CM program), it reacts as if an Xon had been received. If the MUX did not do this, the prompt could not be displayed to the user.

HP PROTOCOL. Read pacing is by DC1 and DC1/DC2/DC1 software handshakes. Write pacing is by ENQ/ACK software handshakes. See the example protocol charts in Appendix A for further information.

HP-Xon/Xoff. This type of protocol is a combination of HP and Xon/Xoff protocols. When this mode is enabled, the Xon/Xoff characters can cause interactions with EDIT/1000. Therefore, alternative characters must be used in place of ctrl-S and ctrl-Q. See the *EDIT/1000 User's Manual*, part number 92074-90001.

CPU-TO-CPU. This protocol has no handshake; flow control must be done by higher level software. The particular advantage of this protocol over TTY is that it disables echoes at all times, overriding the echo bit in the read CNTWD, and it does not append a line feed on write completion or generate a CRLF on read completion.

For normal ASCII reads, backspace and delete are processed as usual, except that no characters are echoed. Although this mode can be used to drive terminals that use local echo and auto-line feed (sometimes erroneously called "half-duplex"), it is primarily intended for non-interactive ports, as when driving "black boxes" or for terminal emulation.

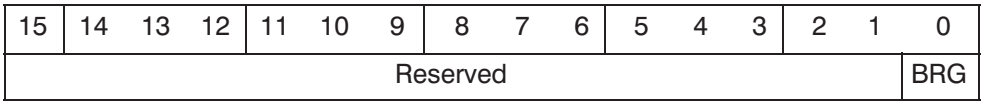
HALF HP. This selection is used to disable the ENQ/ACK handshake of the HP protocol while preserving the D1 pacing on read requests. It is used with satellite links, statistical multiplexers, and some high-speed modems.

HALF HP + Xon/Xoff. This protocol disables the ENQ/ACK portion of HP protocol and enables Xon/Xoff.

## Function Code 35B: Reset a Baud Rate Group

```
CALL EXEC(3,ior(lu,3500b),BRG_Control)
```

The format of control word *BRG_Control* is as follows:




**BRG:** 0 – Reset BRG #0
1 – Reset BRG #1

This control call allows a programmatic reset of a BRG. The current values assigned to the BRG are cleared, allowing the BRG to be set to a preconfigured range by a CN 16 call. The hardware is not reset until the BRG is assigned a new value.

# 3

# Comparison to Previous Drivers

This chapter describes the differences between the Revision 5010 serial I/O drivers supported on the RTE-6/VM Operating System and previous drivers.

## Comparing DV800 and DVM00

DV800 has the following features:

- Supports TTY protocol.
- Supports CPU to CPU protocol.
- Backspace erases characters.
- Rubout produces \CRLF.
- Supports bidirectional Xon/Xoff.
- Performs speed sensing.

The following are differences between DV800 and DVM00:

- DVM00 does not have a write/read or write/write capability. DV800 implements these capabilities so that the feature is available even in driver bypass mode.
- DVM00 terminates normal ASCII reads on RS and DC2. DV800 does not.
- DVM00 uses PRAM3 and PRAM4 for the pace character and number of interrupts to ignore. Because DV800 processes the Z-buffers, this is not possible. DV800 uses the Z-buffer for pacing and the driver communications area for ignore count.
- Disable ENQ/ACK is controlled by CN 34.
- CN 6 (dynamic status request) allows a more compatible form.
- CN 31 and CN 32 (modem connect and disconnect) are combined as CN 31.
- CN 33 (configure driver response) cannot control how device is downed when modem goes down.
- CN 30 now senses the BRG wiring in the hood.

# Comparing DVC00 and DVR00

DVC00 implements the Z-buffer transfers.

- CN 17 calls are supported.
- CN 20 changed to the new convention.
- CN 23 and CN 24 are supported for backward compatibility.
- CN 34 bits 9, 10, and 11 are implemented.

The terminal interface cards used with DVC00 are susceptible to overrun because they have no on-card buffering. The received character must be read by the driver before the first data bit of the next character arrives, which is sometimes too little time for the driver to respond. The driver can detect this overrun condition, and it responds by issuing a backslash, carriage return, line feed sequence and then echoing as much of the line as was correctly received. The user can continue from that point to complete the line.

# A

# Example Protocol Charts

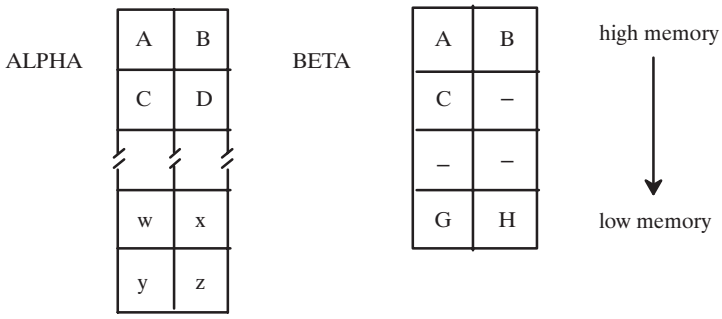
For the following examples, we will assume that the user has a buffer called ALPHA that contains the upper and lowercase alphabet sequences:

```
..../....1..../....2..../....3..../....4..../....5..  ← character number
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz  ← buffer contents
```

We will also make use of another buffer called BETA that has underscore characters (octal 137) as follows:

```
A B C _ _ _ G H
```

In memory, the data would look like the following:

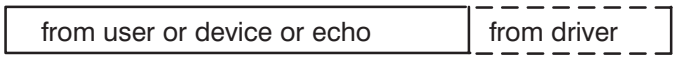



For the read call examples, a 40-word (80-byte) input buffer called IBUF is assumed.

The protocol charts given are representative of what would be observed on the serial data lines using an HP 4953, 4955, or 4951 serial data analyzer. The top line of each chart is the data from the CPU to the device, and the bottom line is the data from the device back to the CPU.

The blank character (octal 40) is represented by " ● " in the protocol charts. A character whose value is not known is represented by " ■ ".

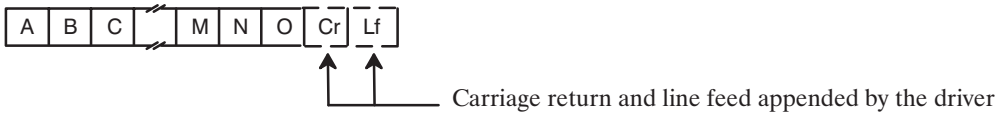
Characters from the user buffer, from the external device, or echoed by the driver are shown in solid outline boxes. Characters generated by the driver are shown in dotted outline boxes.



**Example 1:**

```
Call  EXEC(2,LU,Alpha,-15)
```

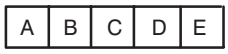
normal ASCII write of 15 bytes to a protocol 0 (TTY) device




**Example 2:**

```
Call  EXEC(2,ior(LU,2000b),Alpha,-5)
```

transparent ASCII write of 5 bytes to a protocol 0 (TTY) device




This time the CRLF was not appended by the driver because the transparency bit was set.

**Example 3:**

```
Call  EXEC(2,LU,Beta,2)
```

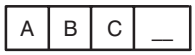
normal ASCII write of 2 words to a protocol 0 (TTY) device


This time the CRLF was not appended by the driver because the last data byte was an underscore. Note that the underscore was also suppressed.

**Example 4:**

```
Call  EXEC(2,LU,Beta,-5)
```

normal ASCII write of 5 bytes to a protocol 0 (TTY) device




Again the CRLF was suppressed by the driver because the last data byte was an underscore. Note that only the last underscore was suppressed. An underscore in any position other than the last character position is data.

**Example 5:**

```
Call EXEC(2,LU,Alpha,26)
```

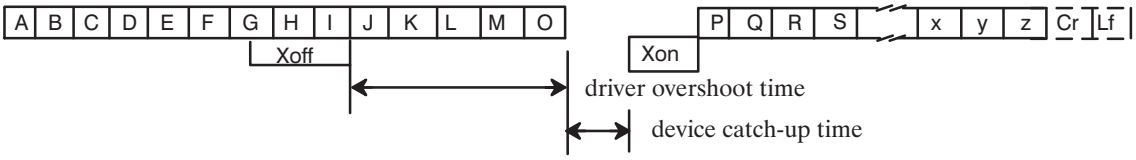
normal ASCII write of 52 bytes to a protocol 2 (HP) device


At least once every 80 bytes of output, the driver will suspend data transmission to perform an ENQ/ACK handshake with the terminal. The terminal will not respond with an ACK until it has processed all the characters preceding the ENQ, so the driver is assured that it is okay to resume transmission of data. The handshake does not necessarily occur in a fixed location in each record because the drivers keep a running 80-byte counter without regard to the records. This uncertainty can be eliminated by using the Force Handshake bit in the write call. The handshake can be suppressed for one line by performing the write in binary mode.

**Example 6:**

```
Call EXEC(2,LU,Alpha,26)
```

normal ASCII write of 52 bytes to a protocol 1 (Xon/Xoff) device




The Xoff character is asynchronous to the transmitted data. It indicates that the buffer of the receiving device is becoming full and that the transmitter should pause. When the receiving device has emptied its buffers to a sufficient degree, it sends an Xon character to the transmitting device to resume the data flow.

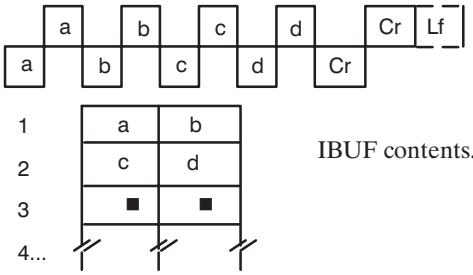
Alternate names:

Xon ↔ DC1 ↔ Ctrl-Q ↔ ^Q ↔ octal 21
Xoff ↔ DC3 ↔ Ctrl-S ↔ ^S ↔ octal 23

**Example 7:**

```
Call  EXEC(1,ior(LU,400b),Ibuf,-80)
Call  Abreg(iStatus,iTransLog)
```

normal ASCII read for 80 bytes with echo from protocol 0 (TTY) device

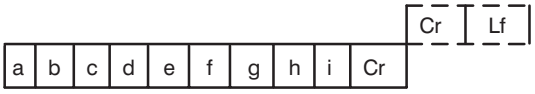



IBUF contents. Transmission log = 4.

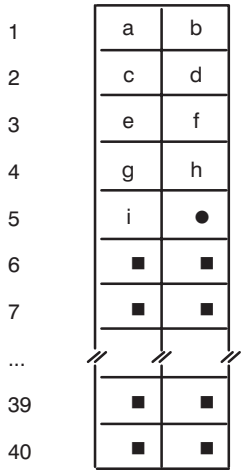
**Example 8:**

```
Call  EXEC(1,LU,Ibuf,-80)
Call  ABREG(iStatus,iTransLog)
```

normal ASCII read for 80 bytes without echo from protocol 0 (TTY) device




The resulting transmission log would be 9, and IBUF would contain:



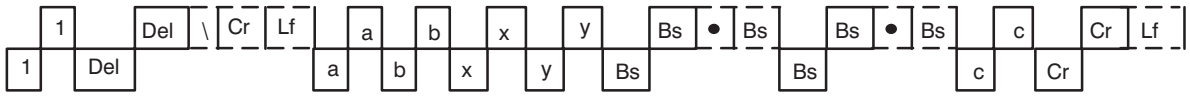

The right byte of the last word is blank padded.

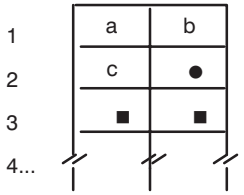
The contents of these words is unspecified. The user should not depend upon their contents to be unaltered. The terminator byte (in this case the carriage return) is not stored in the buffer.

**Example 9:**

```
Call  EXEC(1,ior(LU,400b),Ibuf,-80)
Call  ABREG(iStatus,iTransLog)
```

normal ASCII read for 80 bytes with echo from protocol 0 (TTY) device showing editing characters



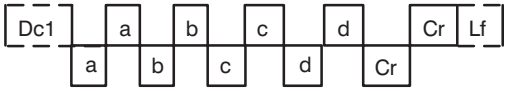

IBUF contents. Transmission log = 3.

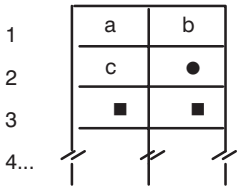
Right byte is blank padded because this is an ASCII read.

**Example 10:**

```
Call  EXEC(1,ior(LU,400b),Ibuf,-80)
Call  ABREG(iStatus,iTransLog)
```

normal ASCII read for 80 bytes with echo from protocol 2 (HP) device



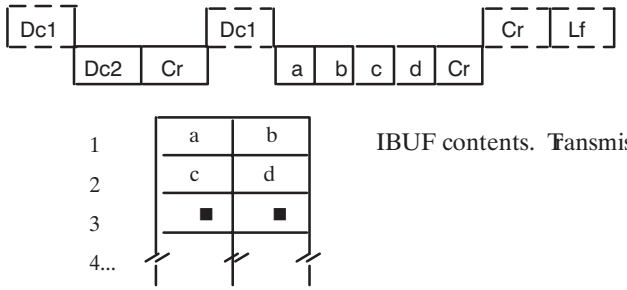

IBUF contents. Transmission log = 3.

Right byte is blank padded because this is an ASCII read.

**Example 11:**

```
Call  EXEC(1,LU,Ibuf,-80)
Call  ABREG(iStatus,iTransLog)
```

normal ASCII read for 80 bytes without echo from protocol 2 (HP) device Terminal strapped for line mode and cursor on a line containing “abcd” when the user presses ENTER.

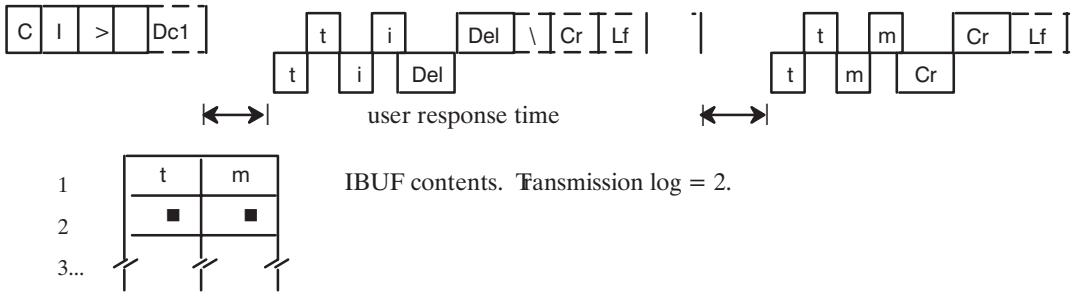



IBUF contents. Transmission log = 4.

**Example 12:**

```
Call  EXEC(1,ior(LU,10400b),ibuf,-80,5hCI> _,-5)
Call  ABREG(iStatus,iTransLog)
```

normal ASCII write/read for 80 bytes with echo from protocol 2 (HP) device

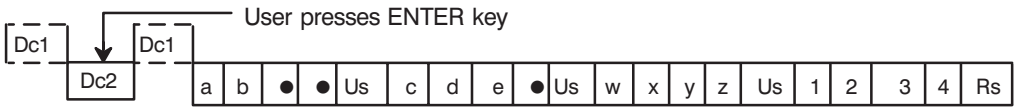



IBUF contents. Transmission log = 2.

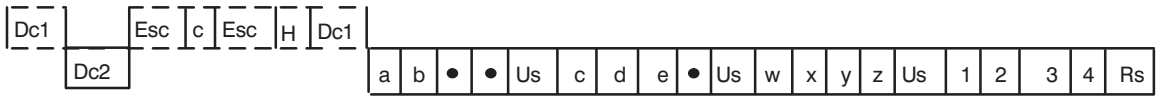
**Example 13:**

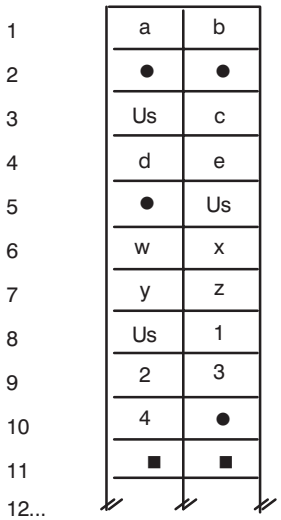
```
Call  EXEC(1,Cntwd,Ibuf,-80)
Call  ABREG(iStatus,iTransLog)
```

normal ASCII read for 80 bytes without echo from protocol 2 (HP) device Terminal strapped for page mode, with a form containing 4 unprotected fields of 4 bytes each.




Same read with Auto-Home bit set.




IBUF contents. Transmission log = 20.

The fields can be accessed individually by calling either HpCrtGetField_S or HpCrtGetField_I.

For example,

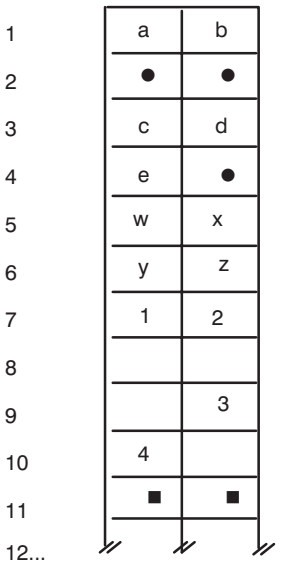
```
Flag  =  HpCrtGetField_S(ibuf,iTransLog,3,String)
```

sets String to “wxyz” and Flag would be .TRUE. because the third field contains data.

In addition, for backward compatibility with DVA05, the call:

```
NewLog  =  HpCrtStripChar(ibuf,iTransLog,37b)
```

performs an in-place conversion of the data buffer by deleting the Unit Separator characters (octal 37).

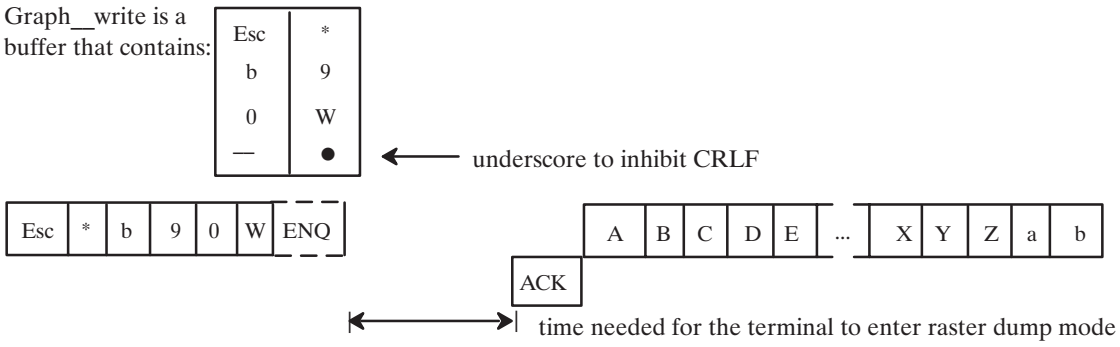



IBUF contents. NewLog = 16.

**Example 14:**

```
Call  EXEC(2,ior(LU,13100b),Alpha,-90,Graph_write,-7)
Call  Abreg(iStatus,iTransLog)
```

transparent binary write/write for 90 bytes forcing handshake to protocol 2 (HP) device. Note that the graphics escape sequence is in the Z-buffer.



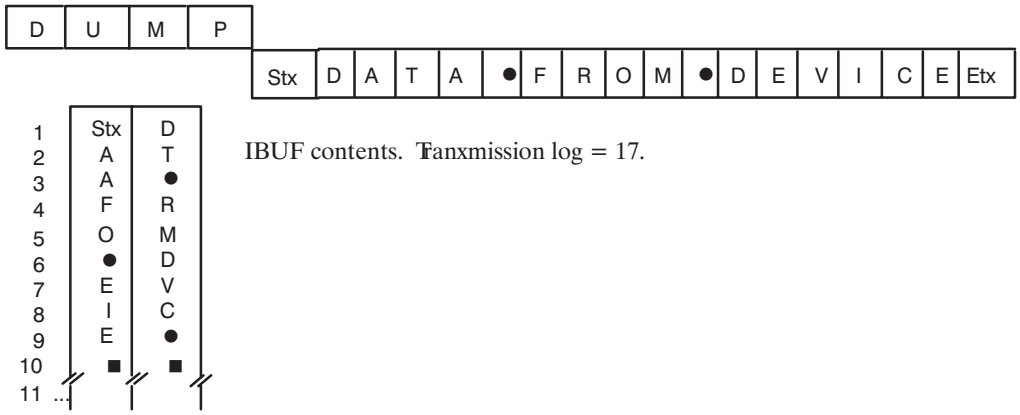

The 90 bytes are sent without intervening handshakes that would spoil the graphics data.

See Chapter 3 of the *HP 2648 Graphics Terminal Reference Manual*, part number 02648-90002, for information on raster dumps.

**Example 15:**

```
Call  EXEC(3,ior(LU,1700b),403b)
Call  EXEC(1,ior(LU,12000b),ibuf,-30,5hDUMP_,-5)
```

ASCII transparent write/read of 30 bytes from a protocol 4 (CPU to CPU) device with terminator configured to be an ETX.

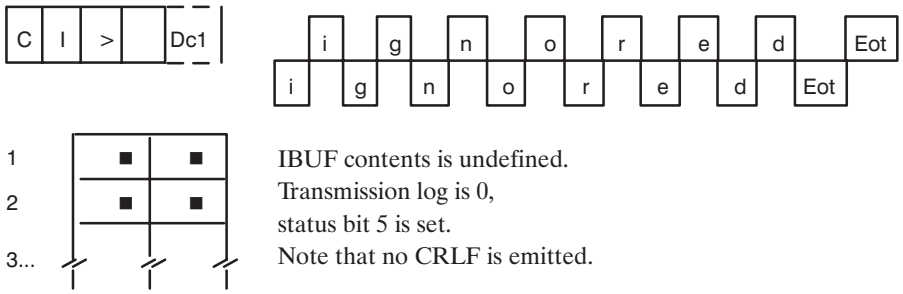



IBUF contents. Tranxmission log = 17.

**Example 16:**

```
Call  EXEC(1,ior(LU,10400b),ibuf,-80,5hCI> _,-5)
Call  ABREG(iStatus,iTransLog)
```

normal ASCII write/read of 80 bytes with echo from protocol 2 (HP) device. The user enters "ignored", which echoes, and then enters a ctrl-D. The data is ignored.

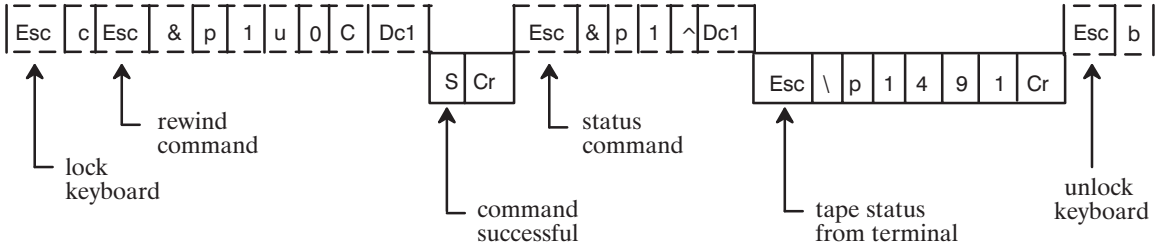



IBUF contents is undefined.
Transmission log is 0,
status bit 5 is set.
Note that no CRLF is emitted.

**Example 17:**

```
Call EXEC(3,ior(LU,400b))
```

rewind cartridge tape (example is for left tape)

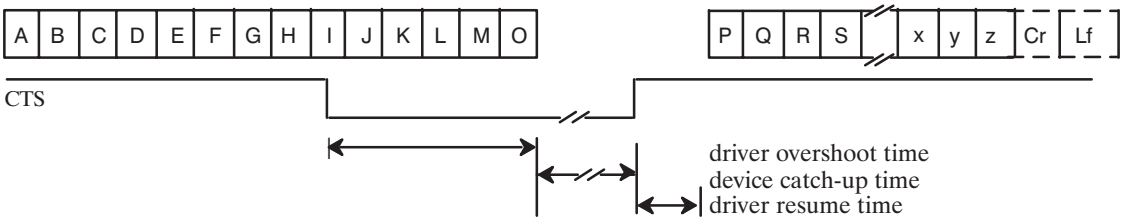



All motion commands are followed by a status check. An ENQ/ACK handshake could appear anywhere in the driver output sequences with no ill effect.

**Example 18:**

```
Call EXEC(2,LU,Alpha,26)
```

normal ASCII write of 52 bytes to a protocol 8 (hardware handshake) device




When the receiving device sets the CTS line low, it indicates that the buffers of the receiving device are becoming full, and that the transmitter should pause. When the receiving device has emptied its buffers to a sufficient degree, it sets the CTS line high to signal the transmitting device to resume the data flow.

# B

# System Generation Considerations

## Generation Instructions for DV800

DV800 consists of the two relocatable files DV800_0.REL and DV800_1.REL. DV800_1 is a superset of DV800_0. DV800_0 is unable to drive slave devices (the CTUs or internal or external printers). You cannot generate both in the same system because you will get duplicate entry point names.

To generate an RTE-6/VM system with DV800, make the following modifications to your answer file:

1. DV800_0.REL or DV800_1.REL must be relocated with the other drivers.

2. Make EQT entries for the D-MUX. The D-MUX drivers have GEN records within the driver that specify values such as EQT Extension size, buffering, and timeout.

   The D-MUX driver does not require DMA, so the MUX can be installed in any slot (above the privilege fence, if you have one) including in an I/O extender that does not have DCPC installed.

   The GEN records have a suffix that indicates with which port the EQT is associated. They also cause the proper choice of protocol based upon the name. For example, to use HP terminals on all eight ports:

```
24,DV800,Gr=Hp_Term:0     *EQT 6  D-MUX port 0
24,DV800,Gr=Hp_Term:1     *EQT 7  D-MUX port 1
24,DV800,Gr=Hp_Term:2     *EQT 8  D-MUX port 2
24,DV800,Gr=Hp_Term:3     *EQT 9  D-MUX port 3
24,DV800,Gr=Hp_Term:4     *EQT 10 D-MUX port 4
24,DV800,Gr=Hp_Term:5     *EQT 11 D-MUX port 5
24,DV800,Gr=Hp_Term:6     *EQT 12 D-MUX port 6
24,DV800,Gr=Hp_Term:7     *EQT 13 D-MUX port 7
```

The GEN records available are:

| | |
|---|---|
| Hp_Term:0..7 | CRT that uses HP ENQ/ACK protocol, performs speed sense, PRMPT enabled, device type 5 |
| Term:0..7 | CRT that uses Xon/Xoff protocol, performs speed sense, PRMPT enabled, device type 0 |
| HP_Printer:0..7 | non-interactive printers that use HP protocol, 9600 baud, program scheduling disabled, device type 12 |
| Printer:0..7 | non-interactive printers that use Xon/Xoff protocol, 9600 baud, program scheduling disabled, device type 12 |
| Hp2635:0..7 | hardcopy terminal that uses HP protocol, performs speed sense, PRMPT is enabled, device type 6 |

To decide which GEN record to use, determine which of the modes above is closest to that needed by the device you will connect to a given port. Probably the most general class is "Term:x", so it is a good choice for "black boxes". You can use control calls to tailor the port after it is booted. If you wish, you can choose to not use the default GEN records and provide all the parameters for the EQT entry.

The RTE-6/VM generator allows you to preset the EQT Extension words, so you can specify the auto-configure parameters yourself. For example:

```
24,Dv800,t=1000,b,x=16,140004b,0,117b,100002b
```

| configures an EQT for: | | |
|---|---|---|
| | Timeout = 10 seconds | (t=1000) |
| | Buffered output | (Buffered bit set) |
| | FIFO mode enabled, char-by-char scheduling, and CPU-to-CPU protocol | (EQx 1 = 140004b) |
| | No special termination | (EQx 2 = 0) |
| | 2400 baud; port 7 | (EQx 3 = 117b) |
| | Schedule program 2 enabled | (EQx 4 = 100002b) |

Refer to the EQT Extension Word Usage at the end of this appendix for extension word formats.

3. Create LUs for the D-MUX by making entries in the LU table. For example:

```
6,0     *D-MUX in SC 24 <LU  30>
7,0     *D-MUX in SC 24 <LU  31>
8,0     *D-MUX in SC 24 <LU  32>
9,0     *D-MUX in SC 24 <LU  33>
```

4. Put the card in the interrupt table, pointing to the MUX pre-driver, for example:

```
24,ent,MPrDv    *D-MUX
```

# Generation Instructions for DVC00

This section provides configuration information for driver DVC00 and is intended to augment the data provided in the *RTE-6/VM System Manager's Manual*, part number 92084-90009.

## Real-Time Generation

Due to the variety of input/output devices that can be controlled by DVC00, it is not possible to provide specific configuration information. Instead, Table B-2 supplies you with typical device configurations that will assist you in configuring your own system. Particular attention should be made to "Note A", detailing the use of DVC01 and DVC02 with a photoreader and punch, respectively.

## HP 2615A Keyboard Display Terminal Processing

The HP 2615A requires that null characters be output before each line. Therefore, it must be configured with an odd channel number.

## EOT/Timeout Options

Driver DVC00 intercepts both a timeout and EOT. The caller always receives the status appropriate to the condition; however, the system need not be notified of that condition. The driver takes the subchannel number assignment, set by the operator during generation or set with the LU command, and performs a Logical OR with both it and the error code. The result is placed in the A-Register prior to the completion return. This has the effect of telling the system what to do with the device when a timeout or EOT occurs. Thus, the subchannel entry in the DRT Table defines both the possible errors sent to the system and the device type, as shown in Table B-1. For example, if a timeout is to set an HP 2762B as down, bits 3 and 0 must be set (type is 2762B), and bit 2 must also be set (timeout downs device). The result of setting bits 3, 2, and 0 to 1 gives a subchannel number of 13 (octal 15). If a timeout is to set the 2762A as down, bit 2 must be set (timeout downs device), and bit 0 must also be set (type is 2762A). Setting bits 2 and 0 gives a subchannel number of 5.

The NOT-READY error caused by low tape on the punch is detected at initiation and will cause the punch to be set down, regardless of the subchannel setting. This implies that a teleprinter or photoreader with a subchannel number of 0 or 1 is set down only by an operator "DN" request.

**Table B-1. DRT Table Entry**

| Subchannel Function | | | |
|---|---|---|---|
| Bit 3<br>"8" | Bit 2<br>"4" | Bit 1<br>"2" | Bit 0<br>"1" |
| HP 2762B | Timeout downs device | EOT downs device | HP 2762A, 2762B or 2615 |

* For an HP 2762B, bit 0 must also be set.

**Table B-2. Typical Device Configurations (DVC00)**

| Device | EQT Entry | DRT Entry | Interrupt Table |
|---|---|---|---|
| 1 – Aux TTY (or CRT) | 11,DVC00, B | 7 = 1 | 11,EQT, 1 |
| 2 – Aux TTY | 12,DVC00 | 8 = 2, 1 | 12,EQT, 2 |
| 3 – Aux TTY | 13,DVC00 | 9 = 3 | 13,EQT, 3 |
| 4 – Punch | 14,DVC02, B, T | 10 = 4, 4 | 14,EQT, 4 |
| 5 – Photoreader | 15,DVC01, T | 11 = 5, 6 | 15,EQT, 5 |
| 6 – Printer | 16,DVC06, B, T | 12 = 6, 13 | 16,EQT, 6 |

**NOTES**

A. At generation time, EOT entries for both photoreader and punch must be specified as follows:

Photoreader – xx, DVC01 (,T )
Punch – yy, DVC02 (,B,T )

Where xx and yy are the device select codes and the bracketed parameters are optional. Note that DVC01 and DVC02 must be used (even though the driver is designated as DVC00) in order for this driver to properly recognize the specific type of device with which it is communicating.

B. Device 1 is an auxiliary TTY or CRT and is buffered.

C. Device 2 is an auxiliary TTY. Because it is an odd-numbered subchannel, it is treated like an HP 2762 Terminal Printer or an HP 2615 or HP 2640 Terminal.

D. Device 3 is an auxiliary TTY that has been designated as a terminal. When enabled, any character input from the terminal schedules the serial program as set by HpCrtSchedProg.

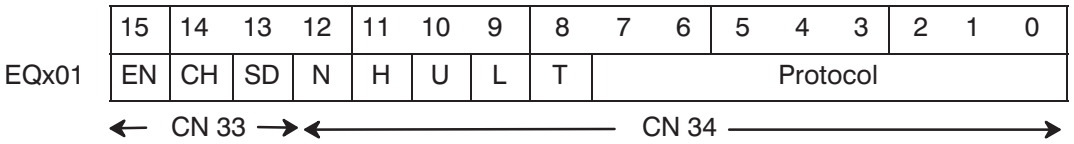
E. Device 4 is a buffered punch with a subchannel equal to four, so it will be downed when a timeout condition occurs.

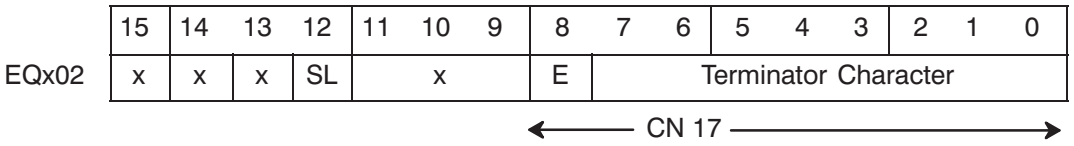
F. Device 5 is a photoreader with a subchannel of 6, so it will be downed on either a timeout or an end-of-tape condition.

G. Device 6 is a serial printer set up with subchannel 13 to get null character stalls and is downed on timeout.

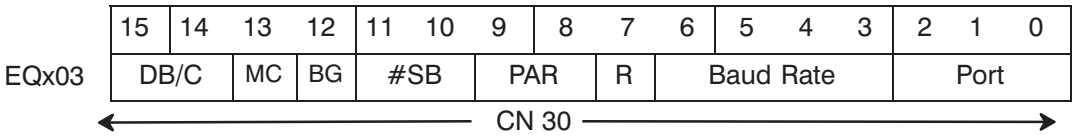
# EQT Extension Word Usage in DV800

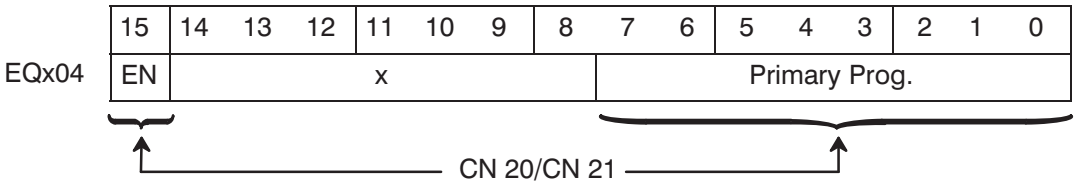
The following is a description of the EQT Extension word usage for DV800. For individual bit definitions, refer to the description of the appropriate CN function code in Chapter 2 of this manual.




Note that bit 12, when set to 1, provides speed sense on logon.




Extension size is 26 words for port 0, 16 words for ports 1 . . . 7

For the Master EQT (the EQT for port 0), there are 10 additional words, as follows:

| | |
|---|---|
| Eqx17 | Raw Card Status (updated by the predriver) |
| Eqx18 | Card State word |
| Eqx19 | EQT address for port 0 |
| Eqx20 | EQT address for port 1 |
| Eqx21 | EQT address for port 2 |
| Eqx22 | EQT address for port 3 |
| Eqx23 | EQT address for port 4 |
| Eqx24 | EQT address for port 5 |
| Eqx25 | EQT address for port 6 |
| Eqx26 | EQT address for port 7 |

# C

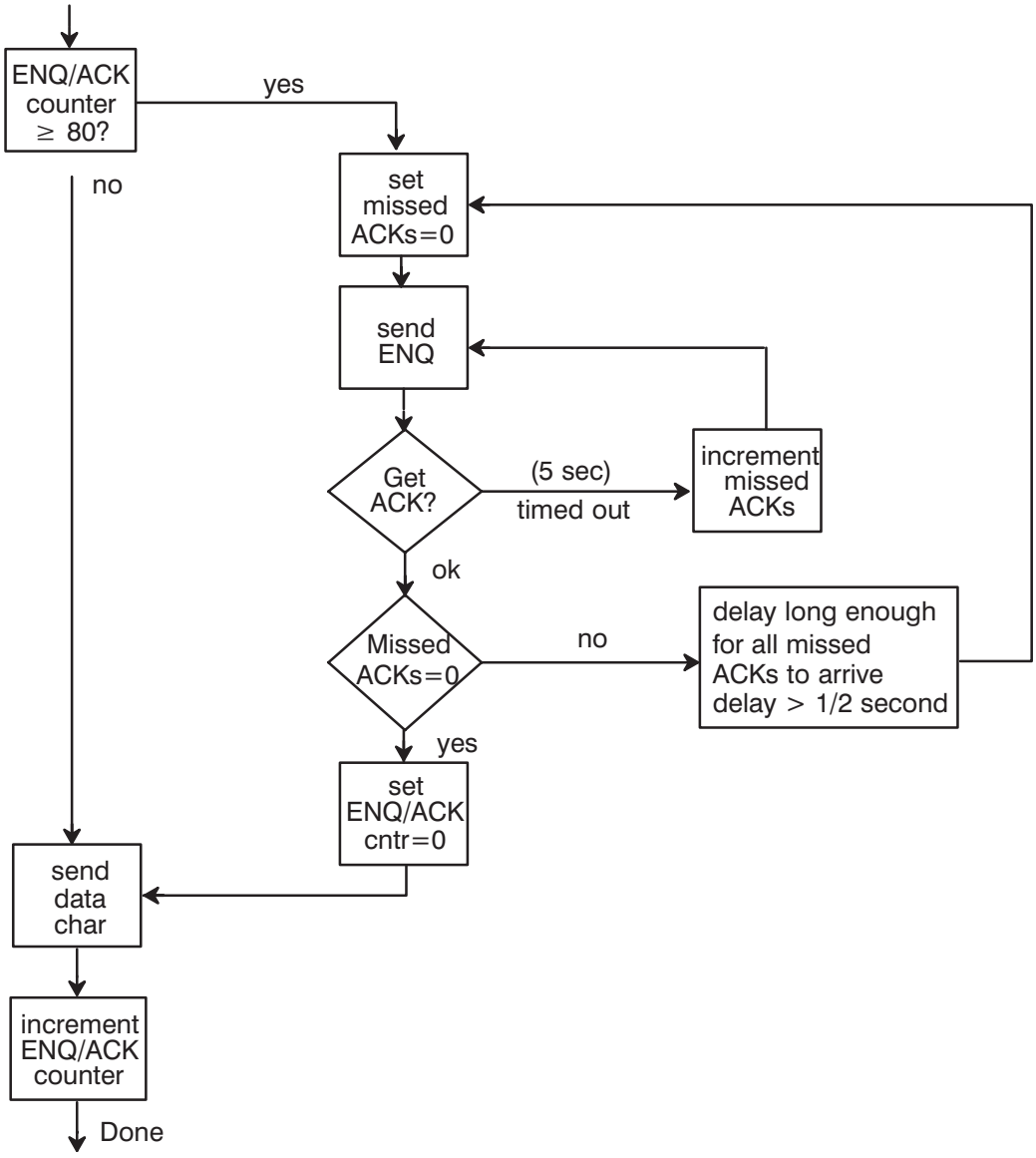
# Example Page Mode Application

```
      character*80 FieldString                  ! to receive one field
      integer*2 Clear(3)                        ! escape sequence
      integer*2 Ibuf(200)                       ! to receive raw screen data
        data Clear /15510b,15512b,15542b/       ! home cursor, clear form,
                                                ! and unlock keyboard
                                                ! (Esc H Esc J Esc b)
100   call EXEC(2,LU,Form,FormLength)           ! put the form on the screen
      call HpCrtPageMode(LU,.true.)             ! put terminal in block page
                                                ! mode, protected forms enabled

200   call EXEC(2,ior(LU,2000b),Clear,-6)       ! clear the unprotected fields,
                                                ! enable the keyboard
      call EXEC(1,ior(LU,200b),                 ! do the block mode read of the
      >          ibuf,-400)                     ! unprotected part of the form
                                                ! auto-home will lock the keys
        call abreg(istatus,itrans_log)          ! get driver status
      if ( iand(istatus,1).eq.1  .or.           ! check for timeout
      >     itrans_log.eq.0           ) then    ! or zero transmission log
        go to 100                               ! which can happen only if
      endif                                     ! the user messed up the CRT
                                                ! so go through set-up again
      flag = HpCrtGetField_S(ibuf,itrans_log,   ! get data from first
      >                      1,fieldstring)     ! unprotected field
      if ( flag ) then
        if ( fieldstring.eq. 'EN' ) then        ! look for end command
          go to 300                             ! to escape application
        endif
      endif

*****************************************
*  Process the data from the screen here  *
*****************************************

      go to 200                                 ! do another screen
300   call HpCrtCharMode(LU)                    ! back to normal terminal mode
      call EXEC(2,ior(LU,2000b),Clear,-6)       ! clear the screen and enable
                                                ! keyboard before terminating
      end
```

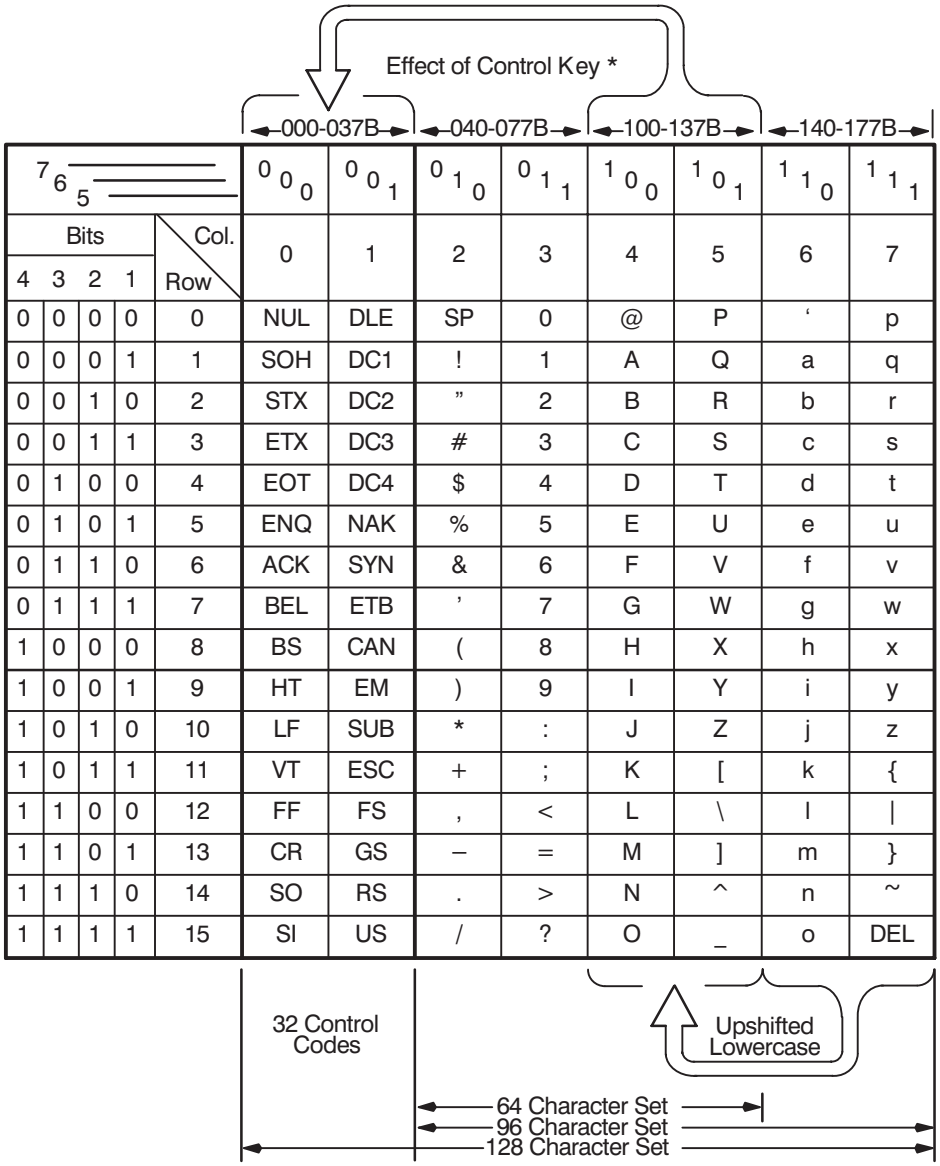
# D

# ENQ/ACK Handshake Details

The following flowchart shows the algorithm used for the ENQ/ACK handshakes to pace writes in HP mode.



# HP Character Set




| 7 6 5 / Bits 4 3 2 1 / Row | 0 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1 0 0 | 1 0 1 | 1 1 0 | 1 1 1 |
|---|---|---|---|---|---|---|---|---|
| Col. | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 0 0 0 0 | NUL | DLE | SP | 0 | @ | P | ‘ | p |
| 0 0 0 1 1 | SOH | DC1 | ! | 1 | A | Q | a | q |
| 0 0 1 0 2 | STX | DC2 | ” | 2 | B | R | b | r |
| 0 0 1 1 3 | ETX | DC3 | # | 3 | C | S | c | s |
| 0 1 0 0 4 | EOT | DC4 | $ | 4 | D | T | d | t |
| 0 1 0 1 5 | ENQ | NAK | % | 5 | E | U | e | u |
| 0 1 1 0 6 | ACK | SYN | & | 6 | F | V | f | v |
| 0 1 1 1 7 | BEL | ETB | ’ | 7 | G | W | g | w |
| 1 0 0 0 8 | BS | CAN | ( | 8 | H | X | h | x |
| 1 0 0 1 9 | HT | EM | ) | 9 | I | Y | i | y |
| 1 0 1 0 10 | LF | SUB | * | : | J | Z | j | z |
| 1 0 1 1 11 | VT | ESC | + | ; | K | [ | k | { |
| 1 1 0 0 12 | FF | FS | , | < | L | \ | l | \| |
| 1 1 0 1 13 | CR | GS | – | = | M | ] | m | } |
| 1 1 0 1 14 | SO | RS | . | > | N | ^ | n | ~ |
| 1 1 1 1 15 | SI | US | / | ? | O | _ | o | DEL |

Example: The representation for the character “K” (column 4, row 11) is

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|
| Binary | 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| Octal | 1 | | 1 | | | 3 | |

Note: * Depressing the Control Key while typing an uppercase letter produces the corresponding control code on most terminals. For example, Control-H is a backspace.

## Table E-1. Hewlett-Packard Character Set for Computer Systems

This table shows Hewlett-Packard's implementation of ANS X3.4-1968 (USASCII) and ANS X3.32-1973. Some devices may substitute alternate characters from those shown in this chart (for example, Line Drawing Set or Scandinavian font). Consult the manual for your device.

The left and right byte columns show the octal patterns in a 16-bit word when the character occupies bits 8 to 14 (left byte) or 0 to 6 (right byte) and the rest of the bits are zero. To find the pattern of two characters in the same word, add the two values. For example, "AB" produces the octal pattern 040502. (The parity bits are zero in this chart.)

The octal values 0 through 37 and 177 are control codes. The octal values 40 through 176 are character codes.

| Decimal Value | Octal Values | | Mnemonic | Graphic[1] | Meaning |
|---|---|---|---|---|---|
| | Left Byte | Right Byte | | | |
| 0 | 000000 | 000000 | NUL | $N_U$ | Null |
| 1 | 000400 | 000001 | SOH | $S_H$ | Start of Heading |
| 2 | 001000 | 000002 | STX | $S_X$ | Start of Text |
| 3 | 001400 | 000003 | EXT | $E_X$ | End of Text |
| 4 | 002000 | 000004 | EOT | $E_T$ | End of Transmission |
| 5 | 002400 | 000005 | ENQ | $E_Q$ | Enquiry |
| 6 | 003000 | 000006 | ACK | $A_K$ | Acknowledge |
| 7 | 003400 | 000007 | BEL |  | Bell, Attention Signal |
| 8 | 004000 | 000010 | BS | $B_S$ | Backspace |
| 9 | 004400 | 000011 | HT | $H_T$ | Horizontal Tabulation |
| 10 | 005000 | 000012 | LF | $L_F$ | Line Feed |
| 11 | 005400 | 000013 | VT | $V_T$ | Vertical Tabulation |
| 12 | 006000 | 000014 | FF | $F_F$ | Form Feed |
| 13 | 006400 | 000015 | CR | $C_R$ | Carriage Return |
| 14 | 007000 | 000016 | SO | $S_O$ | Shift Out } Alternate |
| 15 | 007400 | 000017 | SI | $S_I$ | Shift In } Character Set |
| 16 | 010000 | 000020 | DLE | $D_L$ | Data Link Escape |
| 17 | 010400 | 000021 | DC1 | $D_1$ | Device Control 1 (X-ON) |
| 18 | 011000 | 000022 | DC2 | $D_2$ | Device Control 2 (TAPE) |
| 19 | 011400 | 000023 | DC3 | $D_3$ | Device Control 3 (X-OFF) |
| 20 | 012000 | 000024 | DC4 | $D_4$ | Device Control 4 ($\overline{\text{TAPE}}$) |
| 21 | 012400 | 000025 | NAK | $N_K$ | Negative Acknowledge |
| 22 | 013000 | 000026 | SYN | $S_Y$ | Synchronous Idle |
| 23 | 013400 | 000027 | ETB | $E_B$ | End of Transmission Block |
| 24 | 014000 | 000030 | CAN | $C_N$ | Cancel |
| 25 | 014400 | 000031 | EM | $E_M$ | End of Medium |
| 26 | 015000 | 000032 | SUB | $S_B$ | Substitute |
| 27 | 015400 | 000033 | ESC | $E_C$ | Escape[2] |
| 28 | 016000 | 000034 | FS | $F_S$ | File Separator |
| 29 | 016400 | 000035 | GS | $G_S$ | Group Separator |
| 30 | 017000 | 000036 | RS | $R_S$ | Record Separator |
| 31 | 017400 | 000037 | US | $U_S$ | Unit Separator |
| 127 | 077400 | 000177 | DEL | ■ | Delete. Rubout[3] |

**Table E-1. Hewlett-Packard Character Set for Computer Systems (continued)**

| Decimal Value | Octal Values | | Character | Meaning |
|---|---|---|---|---|
| | Left Byte | Right Byte | | |
| 32 | 020000 | 000040 | | Space, Blank |
| 33 | 020400 | 000041 | ! | Exclamation Point |
| 34 | 021000 | 000042 | ” | Quotation Mark |
| 35 | 021400 | 000043 | # | Number Sign, Pound Sign |
| 36 | 022000 | 000044 | $ | Dollar Sign |
| 37 | 022400 | 000045 | % | Percent |
| 38 | 023000 | 000046 | & | Ampersand, And Sign |
| 39 | 023400 | 000047 | ’ | Apostrophe, Acute Accent |
| 40 | 024000 | 000050 | ( | Left (opening) Parenthesis |
| 41 | 024400 | 000051 | ) | Right (closing) Parenthesis |
| 42 | 025000 | 000052 | * | Asterisk, Star |
| 43 | 025400 | 000053 | + | Plus |
| 44 | 026000 | 000054 | , | Comma, Cedilla |
| 45 | 026400 | 000055 | − | Hyphen, Minus, Dash |
| 46 | 027000 | 000056 | . | Period, Decimal Point |
| 47 | 027400 | 000057 | / | Slash, Slant |
| 48 | 030000 | 000060 | 0 | Digits, Numbers |
| 49 | 030400 | 000061 | 1 | |
| 50 | 031000 | 000062 | 2 | |
| 51 | 031400 | 000063 | 3 | |
| 52 | 032000 | 000064 | 4 | |
| 53 | 032400 | 000065 | 5 | |
| 54 | 033000 | 000066 | 6 | |
| 55 | 033400 | 000067 | 7 | |
| 56 | 034000 | 000070 | 8 | |
| 57 | 034400 | 000071 | 9 | |
| 58 | 035000 | 000072 | : | Colon |
| 59 | 035400 | 000073 | ; | Semicolon |
| 60 | 036000 | 000074 | < | Less Than |
| 61 | 036400 | 000075 | = | Equals |
| 62 | 037000 | 000076 | > | Greater Than |
| 63 | 037400 | 000077 | ? | Question Mark |

**Table E-1. Hewlett-Packard Character Set for Computer Systems (continued)**

| Decimal Value | Octal Values | | Character | Meaning |
|---|---|---|---|---|
| | Left Byte | Right Byte | | |
| 64 | 040000 | 000100 | @ | Commercial At |
| 65 | 040400 | 000101 | A | |
| 66 | 041000 | 000102 | B | |
| 67 | 041400 | 000103 | C | |
| 68 | 042000 | 000104 | D | |
| 69 | 042400 | 000105 | E | |
| 70 | 043000 | 000106 | F | |
| 71 | 043400 | 000107 | G | |
| 72 | 044000 | 000110 | H | |
| 73 | 044400 | 000111 | I | |
| 74 | 045000 | 000112 | J | |
| 75 | 045400 | 000113 | K | |
| 76 | 046000 | 000114 | L | |
| 77 | 046400 | 000115 | M | |
| 78 | 047000 | 000116 | N | Uppercase Letters |
| 79 | 047400 | 000117 | O | |
| 80 | 050000 | 000120 | P | |
| 81 | 050400 | 000121 | Q | |
| 82 | 051000 | 000122 | R | |
| 83 | 051400 | 000123 | S | |
| 84 | 052000 | 000124 | T | |
| 85 | 052400 | 000125 | U | |
| 86 | 053000 | 000126 | V | |
| 87 | 053400 | 000127 | W | |
| 88 | 054000 | 000130 | X | |
| 89 | 054400 | 000131 | Y | |
| 90 | 055000 | 000132 | Z | |
| 91 | 055400 | 000133 | [ | Left (opening) Bracket |
| 92 | 056000 | 000134 | \ | Backslash. Reverse Slant |
| 93 | 056400 | 000135 | ] | Right (closing) Bracket |
| 94 | 057000 | 000136 | ^ ↑ | Caret. Circumflex: Up Arrow[4] |
| 95 | 057400 | 000137 | _ ← | Underline: Back Arrow[4] |

**Table E-1. Hewlett-Packard Character Set for Computer Systems (continued)**

| Decimal Value | Octal Values | | Character | Meaning |
|---|---|---|---|---|
| | Left Byte | Right Byte | | |
| 96 | 060000 | 000140 | ‘ | Grave Accent[5] |
| 97 | 060400 | 000141 | a | |
| 98 | 061000 | 000142 | b | |
| 99 | 061400 | 000143 | c | |
| 100 | 062000 | 000144 | d | |
| 101 | 062400 | 000145 | e | |
| 102 | 063000 | 000146 | f | |
| 103 | 063400 | 000147 | g | |
| 104 | 064000 | 000150 | h | |
| 105 | 064400 | 000151 | i | |
| 106 | 065000 | 000152 | j | |
| 107 | 065400 | 000153 | k | |
| 108 | 066000 | 000154 | l | |
| 109 | 066400 | 000155 | m | |
| 110 | 067000 | 000156 | n | Lowercase Letters[5] |
| 111 | 067400 | 000157 | o | |
| 112 | 070000 | 000160 | p | |
| 113 | 070400 | 000161 | q | |
| 114 | 071000 | 000162 | r | |
| 115 | 071400 | 000163 | s | |
| 116 | 072000 | 000164 | t | |
| 117 | 072400 | 000165 | u | |
| 118 | 073000 | 000166 | v | |
| 119 | 073400 | 000167 | w | |
| 120 | 074000 | 000170 | x | |
| 121 | 074400 | 000171 | y | |
| 122 | 075000 | 000172 | z | |
| 123 | 075400 | 000173 | { | Left (opening) Brace[5] |
| 124 | 076000 | 000174 | \| | Vertical Line[5] |
| 125 | 076400 | 000175 | } | Right (closing) Brace[5] |
| 126 | 077000 | 000176 | ~ | Tilde, Overline[5] |

Note 1: This is the standard display representation. The software and hardware in your system determine if the control code is displayed, executed, or ignored. Some devices display all control codes as “@” or space.
Note 2: Escape is the first character of a special control sequence. For example, ESC followed by “J” clears the display on an HP 2640 terminal.
Note 3: Delete may be displayed as “_”, “@”, or space.
Note 4: Normally, the caret and underline are displayed. Some devices substitute the up arrow and the back arrow.
Note 5: Some devices upshift lowercase letters and symbols (‘ through ~) to the corresponding uppercase character (@ through ^). For example, the left brace would be converted to a left bracket.

**Table E-2. HP 7970B BCD-ASCII Conversion**

| Symbol | BCD (Octal Code) | ASCII Equivalent (Octal Code) | Symbol | BCD (Octal Code) | ASCII Equivalent (Octal Code) |
|---|---|---|---|---|---|
| (space) | 20 | 040 | @ | 14 | 100 |
| ! | 52 | 041 | A | 61 | 101 |
| ” | 37 | 042 | B | 62 | 102 |
| # | 13 | 043 | C | 63 | 103 |
| $ | 53 | 044 | D | 64 | 104 |
| % | 57 | 045 | E | 65 | 105 |
| & | †[1] | 046 | F | 66 | 106 |
| ’ | 35 | 047 | G | 67 | 107 |
| ( | 34 | 050 | H | 70 | 110 |
| ) | 74 | 051 | I | 71 | 111 |
| * | 54 | 052 | J | 41 | 112 |
| + | 60 | 053 | K | 42 | 113 |
| , | 33 | 054 | L | 43 | 114 |
| – | 40 | 055 | M | 44 | 115 |
| . | 73 | 056 | N | 45 | 116 |
| / | 21 | 057 | O | 46 | 117 |
| 0 | 12 | 060 | P | 47 | 120 |
| 1 | 01 | 061 | Q | 50 | 121 |
| 2 | 02 | 062 | R | 51 | 122 |
| 3 | 03 | 063 | S | 22 | 123 |
| 4 | 04 | 064 | T | 23 | 124 |
| 5 | 05 | 065 | U | 24 | 125 |
| 6 | 06 | 066 | V | 25 | 126 |
| 7 | 07 | 067 | W | 26 | 127 |
| 8 | 10 | 070 | X | 27 | 130 |
| 9 | 11 | 071 | Y | 30 | 131 |
| : | 15 | 072 | Z | 31 | 132 |
| ; | 56 | 073 | [ | 75 | 133 |
| < | 76 | 074 | \ | 36 | 134 |
| = | 17 | 075 | ] | 55 | 135 |
| > | 16 | 076 | ↑ | 77 | 136 |
| ? | 72 | 077 | ← | 32 | 137 |

Note 1: †The ASCII code 046 is converted to the BCD code for a space (20) when writing data onto a 7-track tape.

# Index