# HP 91731A Asynchronous Multiplexer Subsystem

## User's Guide

SOFTWARE REV. 1926

HEWLETT PACKARD

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

Updated 7/79
Printed in U.S.A. 12/78

PART NO. 91731-90001

# PUBLICATION NOTICE

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

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

## PUBLICATION HISTORY

Original ........................................ Dec 78 (Software Rev. Code 1901)
Change 1 ........................................ Jul 1979 (Software Rev. Code 1926)



**Library Index Number**
**2RTE.320.91731-90001**

CONTENTS

## ILLUSTRATIONS



## TABLES

SECTION 1

# DESCRIPTION

The Hewlett-Packard 91731A Asynchronous Multiplexer Subsystem is one of the software drivers for the HP Real-Time Executive (RTE) Operating System. The Subsystem provides simultaneous communication with up to sixteen asynchronous bit-serial devices through the HP 12920B Asynchronous Multiplexer. The peripheral devices can be either hardwired to the Multiplexer or connected through full-duplex modems.

A multiplexer allows signals from several lines to be channeled through a single port. In this case, the single port is in the I/O system of an HP 1000 Computer System, and the multiple lines are communication lines between the System and terminals or other devices.

There are many types of multiplexers; for simplicity, consider the Asynchronous Multiplexer to be like sixteen buffered Teletype interfaces. There are differences between the simple Teletype interface and the Multiplexer, but the mode of communication is the same: asynchronous bit serial.

Asynchronous bit-serial devices, numbering in the hundreds, comprise such devices as keyboards, displays, hardcopy terminals, printers, and tape cassette drives. The Multiplexer Subsystem specifically supports certain Hewlett-Packard terminals, including the Cartridge Tape Units and Block Mode on the supported terminals that have those features. The Subsystem also supports an Auxiliary Printer connected to a terminal.

## SUBSYSTEM AND MULTIPLEXER

The HP 91731A Asynchronous Multiplexer Subsystem contains the software that an RTE Operating System uses to communicate with I/O devices through the HP 12920 Multiplexer. The Multiplexer is the interface hardware that physically connects 16 RS-232 lines to three I/O slots in an HP 1000 Computer System. The Multiplexer Subsystem supports one or two Multiplexers, interfacing up to 32 lines.

## HP 91731A ASYNCHRONOUS MULTIPLEXER SUBSYSTEM

The Multiplexer Subsystem consists of one Physical Driver, one Logical Driver, and the Manuals in the following list:

| | Name | Part Number |
|---|---|---|
| For one HP 12920B: | | |
| Physical Driver DVS00 | %DVS0N | 91731-16001 |
| Logical Driver LDVR5 | | |
| Block Mode | %LD5AN | 91731-16002 |
| Block Mode and Cassettes | %LD5AZ | 91731-16003 |
| For two HP 12920B: | | |
| Physical Driver DVS00 | %DVS0Z | 91731-16004 |
| Logical Driver LDVR5 | | |
| Block Mode | %LD5BN | 91731-16005 |
| Block Mode and Cassettes | %LD5BZ | 91731-16006 |
| Source: | | |
| Physical Driver DVS00 | &DVS00 | 91731-18001 |
| Logical Driver LDVR5 | | |
| For one HP 12920B | &LDV5A | 91731-18002 |
| For two HP 12920B | &LDV5B | 91731-18003 |
| Manuals: | | |
| HP 91731A User's Guide | | 91731-90001 |
| HP 91731A Software Numbering Catalog | | 91731-90002 |
| HP 91731A Configuration Guide | | 91731-90003 |

## HP 12920B ASYNCHRONOUS MULTIPLEXER

The HP 12920B Asynchronous Multiplexer consists of three I/O cards, a connector panel for RS-232 connectors, and interconnecting cables.

The Multiplexer has two data boards, Upper Data PCA (12920-60001) and Lower Data PCA (12920-60002) which multiplex the 16 input data lines into one data path and demultiplex one output data path into 16 data lines. There is a Control Board (12922-60001) which sends and receives the line control signals for the 16 terminals or modems that are connected to the Multiplexer.

A Connector Panel (30062-60017) provides the 16 RS-232 connectors and five connectors to connect the terminals and modems to the control and data boards. Only three control connectors are used in the configuration supported by the HP 91731A Subsystem.

A Data Cable (12921-60003), Control Cable (12922-60003), and a Test Cable (30062-60003) complete the HP 12920B Multiplexer hardware supported by the HP 91731A Asynchronous Multiplexer Subsystem.

## LOGICAL AND PHYSICAL DRIVERS

RTE drivers usually control only one type of device each. The Multiplexer, however, can control many types of devices through one I/O port if all the devices are RS-232 compatible.

A single driver capable of handling all these devices would be large and complex, because, while all the devices may be RS-232 compatible and work through similar modems, they may have many different control sequences and line protocols to handle different functions, such as controlling auxiliary printers or cartridge tape units.

To simplify the driver and to regulate the size according to the application, the HP 91731A consists of a logical driver and a physical driver.

The logical driver handles the device dependent functions such as communications protocol, special codes the device understands, and special characters the device sends.

The physical driver handles the device-independent RS-232 functions that are common to all devices. These functions include the line data rate, character format, number of start/stop bits, and modem connect/disconnect. The physical driver also remembers which Multiplexer port requires which type of logical driver.

The logical driver is attached to the physical driver by passing the entry point address of the logical driver and LU number of the Multiplexer port to the physical driver.

## PHYSICAL DRIVER DVS00

DVS00 is the physical driver portion of the HP 91731A Subsystem. It also contains a default Teletype (TTY) logical driver that is used if another logical driver is not attached to the physical driver by the user.

The default TTY driver supports character mode operation and provides a set of default conditions to allow for ease of use and operation. You can change any or all of the default conditions by using control requests described in Section 2 of this manual.

## -LOGICAL AND PHYSICAL DRIVERS-

The default TTY driver that is contained in DVS00 is primarily designed to communicate with a simple terminal similar in features and interface characteristics to a teletype. When the default TTY driver is in use, the characters are input and output in standard 8-bit ASCII.

Some special character processing on input allows the use of carriage return/line feed (CR/LF) as a record separator, CNTL-D as an End Of Tape, deleting characters in the users buffer on backspace, canceling an entry on a rub-out, and interpretation of CNTL-A, CNTL-H, and CNTL-Y as backspace.

On output, the use of left arrow (underline, octal 137) to suppress CR LF and the ability to strike any key to interrupt output are supported. Also, striking any key will schedule a program, if designated during system generation. **When using the default TTY driver, the driver type is 00, for programs that test for this information.**

### LOGICAL DRIVER LDVR5

LDVR5 is a logical driver that operates with the Physical Driver in DVS00. LDVR5 supports keyboard input in either character or block mode. Using LDVR5 with a properly equipped HP terminal, you can input from, output to and control Cartridge Tape Units and Auxiliary Printers. A Block Mode read can be initiated on a 2645A or 2648A terminal from a user program. **When using logical driver LDVR5, the driver type is 05.**

## OPERATING ENVIRONMENT

The HP 91731A Asynchronous Multiplexer Subsystem operates in an HP 1000 Computer System, allowing the user to expand the system I/O capability beyond the unexpanded hardware limitations. One Subsystem connects up to 16 terminals to the Operating System, with optional modem links, while using only three physical select codes. Two Subsystems accommodate up to 32 terminals using six select codes.

## SYSTEM REQUIREMENTS

Inclusion of the HP 91731A Multiplexer Subsystem in an HP 1000 Computer System places certain requirements on the system. The paragraphs that follow summarize hardware and software requirements.

### Hardware

Use of the Subsystem requires that the following hardware be included in the HP 1000 Computer System:

An HP 1000 M, E, or F-Series Computer.

An HP 12920B Interface Kit consisting of two data boards, a control board, a connector panel, and the associated cabling.

An HP 12539C Time Base Generator.

An HP 12620A Breadboard Interface Kit.

### Software

The Asynchronous Multiplexer Subsystem operates in one of the following operating systems:

| | |
|---|---|
| RTE IV | (92067A) |
| RTE M(III) | (92064A) |

However, when using the Multiplexer Subsystem in one of these operating systems, the following restrictions must be observed (for more details, see the HP 91731A Configuration Guide):

RJE 1000 (91780A): may not be run while Subsystem is active.

DS/1000 (91740A/B, 91741A): Privileged data communications may not be used while Subsystem is active.

HP ATS: Subsystem not compatible with HP Automatic Test System.

DATACAP/1000 is not supported by the Multiplexer Subsystem.

Writable Control Store (HP 12971A WCS Kit): Loading WCS Board via DCPC may cause incoming data to Multiplexer to be lost.

Any HP or User code that is privileged (non-interruptable): Incoming data can be lost if a program turns off the interrupt system.

DVS00 must be generated into the operating system as a privileged driver. It requires approximately 4000 bytes of storage plus 52 bytes of Equipment Table (EQT) space in the system driver area in RTE-IV for each port configured into the system.

LDVR5 is also generated into the operating system; however, it resides in the Subsystem Global Area (SSGA) of the operating system. LDVR5 requires approximatly 2970 bytes of storage.

## COMMUNICATION LINE ENVIRONMENT

An important point about the Multiplexer is that it can operate simultaneously and independently on any mix of asynchronous communications lines regardless of the differences in bit rate, line discipline, or operating mode as long as the aggregate baud rate is within that supported by RTE. The thread of commonality between the different communication lines and devices usable by the Multiplexer is that data is transferred one bit at a time.

## Bit-Serial Transmission

In bit-serial transmission, the characters are made up of a group of five to eight bits. Each character is preceded by a start bit and followed by one or more stop bits, depending on the format. Since the Subsystem provides for the character size and the start and stop bit configuration to be programmatically set for each channel, the Multiplexer Subsystem can run any mix of bit-serial devices simultaneously.

## Selectable Transmission Speeds

In bit-serial transmission, characters are sent in a timed sequence of bits. A bit is a unit of information, the term "bit" standing for "binary digit". Often confused with "bit" is "baud", the shortest duration signaling element. Generally, though, "bit" and "baud" are equal, and the terms can be used interchangeably.

There are at least a dozen common speeds at which asynchronous bit-serial devices operate. The speed is the device bit or baud rate. The Asynchronous Multiplexer Subsystem will support 251 separate, different, transmit and receive speeds between 56 and 2400 baud for each channel. The Subsystem is also capable of automatically sensing speeds of 110, 150, 300, 600, 1200, and 2400 baud.

## Full-Duplex Operation

Bit-serial transmission uses only a single wire and a ground return to transfer information. Because of the cost advantage of using a single wire or a pair, this mode of operation has been used since the invention of Morse code. Today's asynchronous bit-serial devices use three wires, one for data out, one for data in, and one for the common return. This three-wire configuration allows the use of full-duplex communication equipment.

Full duplex means that two devices communicating with each other can be transmitting and receiving simultaneously. The data being received can be different from the data sent. Some applications use this feature to print out a different character from the one typed, or to separate input and output into independent information streams.

Usually, as in the case of a Teletype, the full duplex operation is used to echo back the character that was typed. This provides a simple and very effective method of error control. A bit error on the line echoes a character different from the one that was typed, and the error can be detected visually at the terminal.

## Asynchronous Transmission

Asynchronous transmission means that the receiving and transmitting ends can have independent clocks to strobe the bits in or out. The clocks must be going at approximately the same frequency, but can differ by one percent before transmission failure occurs.

Since the clocks are not in synchronization, the transmission mode is termed "asynchronous". In order to know when to shift a bit in, the receiver must be able to synchronize itself with the bit stream. It does this by detecting a start bit and timing itself from this transition to the middle of the remaining bits in the character. It does not matter how much time elapses between the stop bits of one character and the start bit of the next character, because the receiver always resynchronizes itself.

## TERMINALS

Table 1-1 lists the HP terminals that are supported by the Multiplexer Subsystem. While terminals by other manufacturers may be compatible, they are not supported by Hewlett-Packard. The table shows the compatibility of each terminal with DVS00 and with LDVR5 in combination with DVS00.

NOTE

> If a terminal is compatible with the default TTY driver included in DVS00, and that terminal has the ability to function in both block and character mode, then that terminal must be in character mode when using the default TTY driver.

Table 1-1. HP Terminals Supported by HP 91731A

| Terminal Model Number | DVS00 Default TTY Driver | LDVR5/DVS00 Combined |
|---|---|---|
| 2621A/P | YES #+ | YES |
| 2631A Opt 40 | NO | YES |
| 2635A | NO | YES |
| 2640A/B | YES + | YES |
| 2645A | YES *+ | YES |
| 2648A | YES *+ | YES |

\# The auxiliary printer is not subchannel 4, as on 2640-series terminals; output directed to the printer must first be written to the display, then dumped to the printer.

* DVS00 will not access a Cartridge Tape Unit or Auxiliary Printer.

\+ DVS00 operates in character mode only; block mode not supported.

MODEMS

At times it is necessary to communicate with a device over telephone lines or a radio link. In these cases, a modulator-demodulator (modem) pair interfaces with the multiplexer subsystem. By providing important control lines and creating an interface which can be set to give an interrupt on rising, falling, or both edges of a status line signal, control and use of full-duplex modems is supported and maintained.

Modems are controlled by DVS00. Operation of LDVR5 is independent of modem usage.

The Multiplexer Subsystem supports a Bell type 103 or 212 modem, or a Vadic Corporation VA3400 modem. The Bell type 103 is a full duplex, frequency-shift keyed, 0- to 300-baud modem. The Bell type 212 is similar to the 103 except that the data rate is selectable at either 0-300, or 1200 baud. The VA3400 is similar to the 212 dataset, but supports 300, 600 and 1200 baud; however, it can only communicate with another VA3400 modem.

Use of these modems with the Multiplexer Subsystem is supported by Hewlett-Packard in the United States and Canada. In areas outside of the United States and Canada, check with the public telephone authority or independent modem supplier and the local Hewlett-Packard representative to determine modem compatibility with DVS00 and the 12920B interface.

# SECTION 2

# PROGRAMMING AND OPERATION

Operating the Multiplexer Subsystem consists of using EXEC control requests or the File Manager CN command to initialize the terminals, then programming read and write requests to make data transfers through the Multiplexer.

After initialization, you should not normally be concerned with whether a terminal is connected through the Multiplexer or connected directly to the computer. Data transfer requests are either read and write statements in a high-level language program, or EXEC calls from either an Assembly language program or a high-level language program.

## DRIVER RESPONSIBILITIES

This section describes the responsibilities of the physical driver (DVS00) and the logical driver (LDVR5).

### PHYSICAL DRIVER DVS00

The physical driver is responsible for the following:

a. Controlling the HP 12920B interface boards.

b. Making all timing requests, both for itself and for the logical driver.

c. Detecting line errors, parity errors, and break conditions.

d. Calling the logical driver whenever the physical driver:

    Gets a new I/O request,
    Receives a character,
    Sends a character,
    Detects a line error, or
    Senses a time-out of interest to the logical driver.

e. Deciding when an I/O request is complete, or if a control request is to be handled by the logical driver.

f. Maintaining a status indication in EQT word 5 (EQT5); the EQT status conditions are:

| Bits | Meaning |
|---|---|
| 7 | Buffer Flush |
| 6 | BREAK key hit (write only) |
| 5 | End-Of-Tape status |
| 4 | Time-Out |
| 3 | Speed Sense Mode - in progress |
| 2 | Bad Comm Line |
| 1 | Pause Mode (user interrupted output) |
| 0 | Terminal Enabled (1), Disabled (0) |

## LOGICAL DRIVER LDVR5

LDVR5 is a logical driver specifically designed to provide special functions for terminals connected to the Multiplexer. These functions include Block Mode operation and CTU or Auxiliary Printer utilization. Logical drivers in general are responsible for the following:

a. Handling communication processing that is not completely general.

b. Deciding when an I/O request is complete (exception: certain control requests).

c. Maintaining the transmission log of the number of characters or words sent or received by a write or read request.

d. Detecting errors that are defined as errors by the logical driver, such as the violation of message syntax. The errors reported to the logical driver by the physical driver are taken into consideration but are handled as the logical driver dictates. For example, it might take "n" physical errors of a certain kind to result in one logical error.

e. Packing and storing of characters on read; unpacking characters on write.

## REQUEST FORMAT

Input, output, and control requests to the HP 91731A Multiplexer Subsystem are generally in the form of RTE EXEC calls. EXEC can be called from Assembly language programs or from high-level language programs.

The input request is a "read" request. The output request is a "write" request. Read and write requests use EXEC calls, while control requests can be initiated either by using the EXEC call, or by using the File Manager CN command.

## EXEC CALLING SEQUENCES

This section contains general instructions for calling EXEC from RTE Assembly language programs and from RTE FORTRAN IV programs. For other high-level languages, refer to the RTE Programmer's Reference Manual for your RTE Operating system and to the appropriate language reference manual. Refer to this manual for the specific parameter uses.

In the following examples, the only names that must be used as given are EXEC and ABREG. Other names, such as ICODE, ICNWD, and IPR1, are simply mnemonics used in this manual and in other RTE manuals to identify the parameters of the calls. For these mnemonics, you can use any names that suit your purpose.

-REQUEST FORMAT-

Calling EXEC from Assembly Language

The Assembly language calling sequence for EXEC calls is as follows:

```
              .
              .
              .
              EXT EXEC    Declare EXEC as an external
              .
              .
              .
              JSB EXEC    Transfer control to RTE
              DEF RTN     Return address
              DEF ICODE   Request code
              DEF ICNWD   Control word
              DEF IPR1    Parameter 1, optional
              DEF IPR2    Parameter 2, optional
        RTN   Return Point
              .
              .
              .
        ICODE DEC n       n = 1 for read, 2 for write, 3 for control
        ICNWD OCT cnwd    Value depends on function; the lower
                            six bits always contain the LU number
                            assigned to the port
        IPR1  OCT pr1     Use depends on
        IPR2  OCT pr2       type of call
              .
              .
              .
```

On return the A- and B-Registers can be examined for various functions depending on the type of call.

The return point, labeled RTN, must follow the DEF to the last parameter used. EXEC uses this address to calculate the number of parameters passed for those calls that have optional parameters.

Calling EXEC from RTE FORTRAN IV

To call EXEC as a subroutine from RTE FORTRAN IV, use the following calling sequence:

```
              .
              .
              .
      ICODE = n
      ICNWD = cnwd
      IPR1  = pr1
      IPR2  = pr2
              .
              .
              .
      CALL EXEC(ICODE,ICNWD,IPR1,IPR2)
      CALL ABREG(IA,IB)
              .
              .
              .
```

EXEC can also be called as a function from RTE FORTRAN IV, using the following calling sequence:

```
              .
              .
              .
      DIMENSION IREG(2)
      EQUIVALENCE (REG,IREG),(IA,IREG),(IB,IREG(2))
              .
              .
              .
      REG = EXEC(ICODE,ICNWD,IPR1,IPR2)
              .
              .
              .
```

The values of ICODE, ICNWD, IPR1, and IPR2 are as defined for the Assembly language call.

The two different methods of calling EXEC from FORTRAN illustrate the two ways of getting the A- and B-Register returns from the EXEC call. In either case, IA and IB will contain the values returned in the A- and B-Registers respectively. Consult the following sections for the meanings of these returned values for each individual call.

In the following sections the examples used to illustrate the calls are written using the REG = EXEC(... method. This method can be used to make read, write, and control requests. For each control request, there is also an example showing how to make the call from the File Manager, using the CN command.

-REQUEST FORMAT-

EXEC CALL PARAMETERS

Request code ICODE and control word ICNWD, in that order, are the first two parameters of the EXEC call. Other parameters, when needed, are described for each request in the following sections.

Request Code ICODE

Parameter ICODE identifies the type of request. There are four types described in this manual:

```
ICODE= 1       Read request
ICODE= 2       Write request
ICODE= 3       Control request
ICODE= 13      I/O Status request
```

Control Word ICNWD

Control word ICNWD contains a function code and the logical unit number (LU) of the device to which the request is directed. Structure the control word as follows:

```
 15             10          6  5              0
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
| 0  0  0  0  0| F  F  F  F  F| L  L  L  L  L  L|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
               |<--function-->|<--logical unit->|
                     code            number
```

For example, if the function code is octal 5 and the LU number is decimal 10, the value of ICNWD can be programmed:

ICNWD = 10 + 500B

where the B suffix identifies "500" as an octal number.

## Function Code

The octal value of the required function code is given for each of the request descriptions in the following sections. Use whatever method you prefer to place the value in bits 10 through 6 of control word ICNWD. For example, if the value is octal 5, add 500B to the value of the LU number, as illustrated in the previous paragraph. ICNWD = LU when the function code is zero.

## Logical Unit Number LU

The logical unit number is the system address for the I/O device to which you are directing the request. Your system generation listing lists the LU numbers of all I/O devices integrated into the system. In the case of the Multiplexer Subsystem, each LU number corresponds to a Multiplexer port.

## CONTROL REQUESTS

Control requests have the following general format:

```
REG = EXEC(ICODE,ICNWD,IPR1)
```

When issuing a control command from the File Manager the following format is used:

```
:CN,lu[,fn[,pr]]

     lu = LU number
     fn = function code
     pr = optional parameter
```

For example, to find file 19 on a CTU assigned to LU 23:

```
:CN,23,27B,19
```

### PARAMETERS

Request code ICODE has a value of 3 for all control requests. Each control request has a different function code in ICNWD, and the value of parameter IPR1 depends on the function code. Table 2-1 is a summary of the function codes and their meanings, their availability with LDVR5 and DVS00, and the applicability of optional parameter IPR1.

### RETURNS

In the Equipment Table (EQT), word five (EQT5) is the status word. On return from a control request, the A-Register contains the device status from EQT5, if the EQT is unbuffered. If the EQT is buffered, the A-Register is meaningless. In both cases the B-Register is meaningless.

Table 2-1. Control Request Function Codes and Parameters

| Code | Description | LDVR5 | DVS00 | IPR1 |
|---|---|---|---|---|
| 01 | Write end-of-file (EOF) to CTU | Yes | No | No |
| 02 | Backspace one record on CTU | Yes | No | No |
| 03 | Forward space one record on CTU | Yes | No | No |
| 04 | Rewind CTU | Yes | No | No |
| 05 | Set receive speed parameter | Yes | Yes | Yes |
| 07 | Set end-of-tape status | Yes | Yes | No |
| 10 | Write EOF if not just previously written | Yes | No | No |
| 11 | Space printer or CRT | Yes | No | Yes |
| 12 | Set CR/LF delay | No | Yes | Yes |
| 13 | Forward space one file on CTU | Yes | No | No |
| 14 | Backspace one file on CTU | Yes | No | No |
| 16 | Set send speed parameter | Yes | Yes | Yes |
| 17 | Set speed sense mode | Yes | Yes | Yes |
| 20 | Enable terminal | Yes | Yes | Yes |
| 21 | Disable terminal | Yes | Yes | No |
| 22 | Set time-out | Yes | No | Yes |
| 23 | Flush buffer | Yes | Yes | No |
| 24 | Unflush buffer, restore output processing | Yes | Yes | No |
| 25 | Attach logical driver | Yes | Yes | Yes |
| 26 | Write end-of-data (EOD) to CTU | Yes | No | No |
| 27 | Locate absolute file on CTU | Yes | No | Yes |
| 30 | Close line for modem | Yes | Yes | No |
| 31 | Open line for modem | Yes | Yes | Yes |
| 32 | Update Terminal Status | Yes | No | No |

## INITIALIZATION CONTROL REQUESTS

Port initialization consists of a sequence of control calls that first set up conditions for the physical driver and then issue an Enable Terminal request. Conditions that are to be the same as the corresponding default condition need not be specified at initialization time. Conditions that are to be different from the default condition must be set using individual control calls.

The default conditions are:

| | |
|---|---|
| Baud rate: | 110 for transmit and receive |
| CR/LF Delay: | None |
| Hardwired Terminal: | Modems not in use |
| Logical Driver: | Default TTY Driver |
| Character size: | 8 bits |
| Parity: | None |
| Backspace Mode: | Echo underscore on backspace |
| Time-out Action: | Down device |
| Echo: | On |

The conditions to be set up individually are the transmit and receive baud rates, CR/LF delay, and modem usage. Also, if desired, a logical driver can be attached to the physical driver at initialization time. Conditions that are included in the Enable Terminal request are character size, parity, backspace mode, time-out action, and echo.

Note that the default conditions are valid only at boot-up. Once a condition has been changed, the new value becomes the default value until the system is rebooted.

## BAUD RATE SPECIFICATION

The Multiplexer transmit and receive baud rates are derived from a numerical rate parameter (IRP) specified by the user. The formula for calculating the actual baud rate from this rate parameter is as follows:

$$BAUD = 14400./(IRP+1)$$

$$\text{where: } 0 <= IRP <= 255$$

There are 256 discrete baud rates available. IRP = 0 is 14400 baud and IRP = 255 (377 octal) is 56.25 baud. However, the maximum baud rate that is physically reachable by the multiplexer hardware is 2400 baud on any one terminal. Not all of the remaining 251 baud rates available are integer baud rates. But, since this is an asynchronous communication link, a one per-cent error between the terminal and multiplexer baud rate is acceptable.

To find the IRP required for any given baud rate, calculate:

IRP = (14400/BAUD)-1

Then use the resultant IRP (an integer) in the original formula to calculate the actual baud rate and verify that the actual baud rate is within one per-cent of the desired baud rate.

For example to obtain a multiplexer output at 110 baud:

(14400/110)-1 = 129.91

IRP = 129

BAUD = 14400/(129+1)

BAUD = 110.77

The baud rate 110.77 is within one per-cent (for example, plus or minus 1.1) of 110 baud, which is acceptable. A better value for the IRP would have been 130, which yields an actual baud rate of 109.92 baud.

All possible baud rates between 56 and 2400 baud are not available; however, all the common ones, including all the available baud rates on HP terminals at 2400 baud and below, are usable.

SET RECEIVE SPEED

By issuing an EXEC control request to a Multiplexer logical unit, the user can set the baud rate at which the Multiplexer expects to receive data from a terminal. The terminal must be physically set to transmit at that speed.

The function code is octal 5. To form the control word add the LU number of the terminal to 500B. An additional parameter is required to pass the rate parameter (IRP).

A typical calling sequence to set the receive speed would be:

```
ICODE = 3
ICNWD = LU + 500B
IRP = (14400/IBAUD) - 1
REG = EXEC(ICODE,ICNWD,IRP)
```

Where the value of IBAUD is the desired baud rate, and LU contains the LU number of the port.

To set receive speed from the File Manager use the CN command. For example, to set LU 10 to 2400 baud (IRP = 5):

```
:CN,10,5B,5
```

SET SEND SPEED

In a manner similar to setting the receive speed, the user can set the baud rate at which the Multiplexer sends data to a terminal. The terminal must be physically set to receive at the desired baud rate.

The function code is octal 16, and the required parameter is the same rate parameter (IRP) as is used to set the receive speed. The send and receive speeds do not have to be set to the same baud rate.

A typical calling sequence to set the send speed is:

```
ICODE = 3
ICNWD = LU + 1600B
IRP = (14400/IBAUD) - 1
REG = EXEC(ICODE,ICNWD,IRP)
```

Where the value of IBAUD is the desired baud rate, and LU contains the LU number of the port.

To set send speed from the File Manager use the CN command. For example, to set LU 10 to 2400 baud (IRP = 5):

```
:CN,10,16B,5
```

## ATTACH LOGICAL DRIVER

The logical driver, or default TTY driver, that is associated with a specific LU number will generally remain static once the Multiplexer terminals have been initialized. It is therefore advisable to initialize each Multiplexer terminal at boot-up and attach the proper logical driver at that time. This does not preclude changing the logical driver later.

It is not recommended that logical drivers be attached using the File Manager CN command due to the possibilities for errors; however, the default TTY driver can be specified using that method.

When a logical driver is attached to DVS00, the address of the logical driver must be specified. To get the address of logical driver LDVR5, the driver must be called as a subroutine, and passed a parameter in which to return the address. This parameter is then passed to DVS00 via a control request with a function code of octal 25.

For example to attach LDVR5 to the physical driver:

```
CALL LDVR5(IADDR)
ICODE = 3
ICNWD = LU + 2500B
REG = EXEC(ICODE,ICNWD,IADDR)
```

The first call to the driver after LDVR5 is attached will change the driver type to 05. Note that the type number does not completely describe the characteristics of a driver. For proper operation, always consult the reference manual for the particular driver in question.

The same calling sequence in Assembly language:

```
          EXT EXEC,LDVR5   EXTERNAL REFERENCES
           .
           .
          JSB LDVR5        CALL LDVR5
          DEF *+2          DEFINE RETURN ADDRESS
          DEF IADDR        DEFINE STORAGE FOR DRIVER ADDRESS
          LDA LU           FORM
          ADA =B2500       CONTROL WORD
          STA ICNWD        AND SAVE
          JSB EXEC         TRANSFER CONTROL TO SYSTEM
          DEF *+4          DEFINE RETURN ADDRESS
          DEF ICODE        DEFINE EXEC CODE LOCATION
          DEF ICNWD        DEFINE CONTROL WORD LOCATION
          DEF IADDR        DEFINE LOGICAL DRIVER ADDRESS LOCATION
           .
           .
    IADDR BSS 1            STORAGE FOR LOGICAL DRIVER ADDRESS
    ICNWD BSS 1            STORAGE FOR CONTROL WORD
    ICODE DEC 3            EXEC CODE 3
    LU    BSS 1            STORAGE FOR LU NUMBER
           .
           .
```

NOTE

Programs that call LDVR5 by name must be loaded with access to the SSGA area by using the Loader OP,SS command or by declaring during system generation that the program type is 18 or 19. Types 18 and 19 are types 2 and 3 with 16 added to the type number to specify SSGA access.

The first call to the driver after LDVR5 is attached will change the driver type to 05.

To reassign the default TTY driver, simply pass a 0 to DVS00 in the address parameter:

```
IADDR = 0
REG = EXEC(ICODE,ICNWD,IADDR)
```

The default TTY driver also may be reassigned using the File Manager CN command:

```
:CN,lu,25B,0
```

**Reassigning the default TTY driver returns the driver type to 00.**

NOTE

Request an Update Terminal Status whenever you attach LDVR5, to ensure that the logical driver is aware of the current terminal strapping configuration.

UPDATE TERMINAL STATUS

An Update Terminal Status call is a control request with a function code of octal 32, updating the driver with respect to the current status of the terminal strap settings. No parameters are required; the driver obtains this information directly from the terminal when this call is made.

Make a Update Terminal Status call whenever the BLOCK MODE switch or terminal internal straps are changed and whenever the logical driver LDVR5 is attached to the physical driver for a specific port.

For example, to update the status of the driver for LU number 41, the following code could be used:

```
     .
     .
     .
ICODE = 3
ICNWD = 3200B + 41
REG = EXEC(ICODE,ICNWD)
     .
     .
     .
```

Or, using a File Manager CN command to make the same status update:

```
:CN,41,32B
```

ENABLE TERMINAL REQUESTS

The Enable Terminal control requests set six terminal conditions and initialize the 12920B Multiplexer hardware to communicate with a terminal. The function code is octal 20. One parameter word (IPRAM) specifies the six conditions. The functions of the bits in the parameter word are specified in Table 2-2. The meanings of the various conditions are detailed in the following sections.

Table 2-2. Enable Terminal Parameter Word

| Bits | Meaning |
|---|---|
| 15 | Must be a 1 |
| 14 | Must be a 0 |
| 13 | Action on time-out<br>0 = Down device<br>1 = Do not down device |
| 12 | Backspace mode (default TTY driver only)<br>0 = Echo underscore on backspace<br>1 = Do not echo underscore on backspace |
| 11 - 9 | Not used |
| 8 | Stop bits<br>0 = 1 Stop bit<br>1 = 2 Stop bits |
| 7 | Echo mode<br>0 = Echo off<br>1 = Echo on/enabled |
| 5 and 4 | Parity<br>0 = Parity off<br>1 = Parity odd<br>2 = Parity even |
| 3 - 0 | Character size; size not including stop and parity bits |

The default parameter word is octal 100210, which specifies:

| | |
|---|---|
| Backspace Mode: | Echo underscore on backspace |
| Time-out Action: | Down device |
| Stop Bits: | One |
| Echo: | On |
| Parity: | None |
| Character Size: | 8 bits |

The sequence required to enable a terminal with a parameter word of, for example, 130210 octal is as follows:

```
ICODE = 3
ICNWD = LU + 2000B
IPRAM = 130210B
REG = EXEC(ICODE,ICNWD,IPRAM)
```

The File Manager CN command can be used to initialize a terminal in the following manner:

:CN,lu,20B,130210B

Specifying a parameter word value of 0 will invoke the default conditions if these conditions have not been changed since boot-up. If these conditions have been changed, a parameter word value of 0 will enable the terminal with the conditions that were established by the previous Enable Terminal request.

## Time-Out Action

A time-out occurs when a device takes longer than a predetermined time to complete a request (read, write, or control), or some error occurs and is reported as a time-out (all errors cause time-outs). The time-out period may be set by the user, as described later for the "Set Time-Out" control request for the LDVR5 only.

The action that occurs when there is a time-out is determined by bit 13 of the Enable Terminal parameter word. If bit 13 was "0" when the terminal was enabled, the device LU is downed by the system whenever a time-out occurs. If bit 13 was "1" the device LU is not downed on a time-out.

If the device is not set down, control is returned to the calling program and bit 4 of EQT5 is set. This bit can be checked by a status call, as described under "I/O Status" in the section on Status and Error Handling.

A downed device LU must be set UP using the system UP command on the EQT number associated with the LU. See your RTE Operating System Reference Manual for details.

Backspace Mode

Bit 12 in the Enable Terminal parameter word controls the way the default TTY driver echoes backspace (CNTL-A, CNTL-H, or CNTL-Y) to a terminal. Response to backspace also depends on how you set "Echo" mode, described later.

If bit 12 in the Enable Terminal parameter word is set to "1" and echo is on, the driver echoes only a backspace. Echoing the backspace moves the cursor one space to the left but does not erase the character at the new cursor position from the screen. Most hardcopy terminals backspace the print head one position.

If bit 12 is "0", the driver echoes the backspace followed by an underscore (octal 137, left arrow on some terminals). This replaces the last character printed (writes over on hard copy terminals) with the underscore. The cursor returns to its original position, one space to the right of the underscore. If echo is off, the backspace is not echoed; however, the underscore may be sent as controlled by bit 12.

Repeated backspaces in the backspace-underscore mode will have no further effect on the display. In the backspace only mode, repeated backspaces will move the cursor one space to the left for each backspace. Thus, setting bit 12 to "0" is useful on terminals that do not reposition the cursor or print head in response to the backspace. For such terminals, the number of underscores or back-arrows gives visual indication of the number of backspaces entered.

If you delete the entire buffer, the default TTY driver sends a slash, LDVR5 sends a backslash CR/LF to the terminal.

In either mode each backspace typed deletes the last character from the users buffer. This means that to correct an error at the beginning of a line everything after the error must be retyped, even though the text may still be intact on the terminal screen.

The backspace with underscore option is available only with the default TTY driver. LDVR5 forces backspace without underscore regardless of bit 12.

Stop Bits

Bit 8 in the Enable Terminal parameter word specifies the minimum number of stop bits inserted between characters on a data line. There is no maximum number, as all bits encountered prior to a start bit are considered stop bits. Figure 2-1 illustrates the sequence of data bits for one character.

```
+--------------------------------------------------------------------------+
|                                                                          |
|            ---     ------------------------     ---                      |
|           |   |   |                       |   |   |                      |
|    1    1 | 0 | 1 | 0    0    0    0    0 | 1 | 0 | 1    1    1         |
|           |   |   |                       |   |   |                      |
|   -------     ---                          ---     ---------------       |
|          Start LSB-----Character-----MSB Parity   Stop Bits              |
|           Bit                                                            |
|                                                                          |
|            (Note timing resynchronized by Start Bit)                     |
|                                                                          |
+--------------------------------------------------------------------------+
```

Figure 2-1. Character Sequence for Letter "A"

A "0" in bit 8 specifies one stop bit. A "1" in bit 8 specifies two stop bits. If the baud rate is 110, two stop bits are used and bit 8 is ignored.

Echo Mode

Echo is the feature of the driver that causes characters to be printed on the screen as they are typed on the keyboard of a terminal. For certain inputs, such as access passwords, users may want to turn echo off.

When bit 7 in the Enable Terminal parameter word is "0", echo is turned off; when it is "1", echo is turned on. When using LDVR5, and bit 7 is set to "1", echo is controlled by bit 8 of the read request control word for each individual read request. If echo is off, bit 8 is ignored and assumed to be zero.

NOTE

> This is a hardware bit-for-bit echo; the echoed bits are at the received rate. If transmit and receive rates are different, turn echo off.

## Parity Checking

When parity is specified, an extra bit is sent along with each character. This bit is set to a "1" or a "0" to make the total number of 1's in the binary character odd (or even). When the character is received, if the total number of 1's is not odd (or even) it can be assumed that a bit error has occurred in transmission.

Bits 5 and 4 of the Enable Terminal parameter word specify the parity. A 1 (binary 01) specifies odd parity will be used; a 2 (binary 10) specifies even parity will be used. A 0 (binary 00) specifies no parity; in this case no parity bit is sent or expected.

On an HP terminal when no parity is selected at the keyboard, a "0" is always sent as the parity bit; therefore, when using an HP terminal with no parity, the character size should be set at 8 bits (see "Character Size") even though the terminal is sending a 7-bit character plus 1 bit of parity that is always a "0". When using even or odd parity, specify proper parity and a character size of 7 bits.

NOTE

> Odd, or even parity checking is available for ASCII reads and writes only. Parity checking and parity generation must not be used for binary reads or writes.

## Character Size

The value in bits 3 through 0 of the Enable Terminal parameter word specifies the number of bits in each character, not including start, stop, or parity bits.

## Enable Terminal Example

Figure 2-2 shows a program example that could be used to initialize a terminal for use with the Multiplexer. Run program INIT from the WELCOM file once for each terminal on the Multiplexer. For example, four terminals on LU's 22 through 25:

```
:RU,INIT,22
:RU,INIT,23
:RU,INIT,24
:RU,INIT,25
```

```
FTN4,L
      PROGRAM INIT (19), ***PROGRAM TYPE SPECIFIES SSGA USE
C
C THIS PROGRAM INITIALIZES A MULTIPLEXER TERMINAL.
C LU NUMBER OF TERMINAL IS PASSED IN FIRST PARAMETER.
C
      DIMENSION LU(5)
      DATA IBAUD/5B/,IRS/500B/,ITS/1600B/,LDVR/2500B/
      DATA IENAB/2000B/,IPRAM/130210B/,ICODE/3/,IUPST/3200B/
C
C GET LU NUMBER
C
      CALL RMPAR(LU)
C
C SET RECEIVE BAUD RATE AT 2400 BAUD
C
      ICNWD = LU + IRS
      REG = EXEC(ICODE,ICNWD,IBAUD)
C
C SET TRANSMIT BAUD RATE AT 2400 BAUD
C
      ICNWD = LU + ITS
      REG = EXEC(ICODE,ICNWD,IBAUD)
C
C GET LOGICAL DRIVER ADDRESS AND PASS TO PHYSICAL DRIVER
C
      CALL LDVP5(IADDR)
      ICNWD = LU + LDVR
      REG = EXEC(ICODE,ICNWD,IADDR)
C
C UPDATE TERMINAL STATUS
C
      ICNWD = IUPST + LU
      REG = EXEC(ICODE,ICNWD)
C
C ENABLE TERMINAL WITH BACKSPACE, NO DOWN ON TIME-OUT, ONE STOP
C BIT, NO PARITY, AND 8 BIT CHARACTERS
C
      ICNWD = LU + IENAB
      REG = EXEC(ICODE,ICNWD,IPRAM)
      END
      END$
```

Figure 2-2. Enable Terminal Example: Program INIT

## I/O OPERATIONS

Once an LU has been initialized, it is ready to perform its principal function, input and output. Most input and output through the Multiplexer is initiated by high-level language READ and WRITE statements.

These I/O statements generate calls to library modules that make EXEC calls to RTE. When using high-level language I/O statements, you may be communicating with a DVS00, LDVP5, or a non-Multiplexer terminal. The specific calls to different types of devices are handled by the library modules and the RTE operating system.

There are, however, times when you need to control the driver directly in order to handle a special case data transfer that would not be possible with the more general high level language I/O statement.

This section tells you how to use EXEC calls to communicate directly with the driver.

### USING THE DEFAULT TTY LOGICAL DRIVER

Read or write requests using the default TTY driver are in the form of EXEC calls with a request code (ICODE) of 1 for read and 2 for write. The control word (ICNWD) must contain the LU number of the Multiplexer device. The default TTY driver ignores all other bits in the control word.

For both reads and writes two more parameters are required. Buffer and buffer length parameters must be passed to the driver by the EXEC call.

The buffer length must in all cases be less than or equal to the storage reserved for the buffer. For a write request the buffer must be less than the line length of the terminal, unless the terminal has a wrap-around mode.

The buffer length parameter may be positive or negative. If positive, the length specified is the number of words. If negative, the length is the number of characters.

As an example, the following sequence writes seven characters from a buffer called IOUT to LU 41, then reads five words from the same LU to a buffer named INBUF:

```
      .
      .
      .
ICODE = 2
ICNWD = 41
IBUFL = -7
REG = EXEC(ICODE,ICNWD,IOUT,IBUFL)
ICODE = 1
IBUFL = 5
REG = EXEC(ICODE,ICNWD,INBUF,IBUFL)
      .
      .
      .
```

Some points to consider when making read and write requests to default TTY driver DVS00:

- o Input and Output are ASCII only; binary mode not permitted.
- o On read, a carriage return (CR) is interpreted as an end of record (EOR) and a carriage-return line-feed (CR/LF) is echoed unless echo is turned off.
- o On write, the driver adds CR/LF to each record unless the last character is an underscore. If the buffer length is specified by a positive integer, an even number of characters is sent to the terminal. Take care that the underscore is the last character in the string if CR/LF suppression is desired.
- o The driver ignores line feeds and leading nulls on reads.
- o The driver will accept zero-length records from the terminal.
- o There is no transparent mode (inhibit special character processing) for the TTY driver.

## LDVR5 TO KEYBOARD/DISPLAY

Read/write requests to a keyboard/display cover three types of data transmission: writing to a display (request code 2), and two types of reads from a keyboard (request code 1). The two types of reads are character mode read and block mode read.

In character mode read with echo on, the driver enters each character into the buffer and the hardware echoes the character back to the display as it is entered at the keyboard. In block mode read, the terminal saves characters in display memory until either the ENTER key is pressed or the program initiates a block read. The block can consist of one line or one page of characters, depending on terminal strapping.

## Function Code

A read or write request to a keyboard/display using LDVR5 is the same as a read or write request using the default TTY driver except that the function code, bits 10 through 6 in the control word (ICNWD) are set by the user to control the driver and terminal. The functions of the control word bits are shown in Table 2-3.

Table 2-3. Keyboard/Display Read/Write Request Control Word for LDVR5

| Bits | Function |
|---|---|
| 15-11 | Not used. |
| 10-9 | 00 = Non-transparent character mode; terminal can initiate block or character reads.<br>01 = Not used.<br>10 = Transparent character mode; terminal can initiate character or block reads.<br>11 = Program-initiated block reads. |
| 8 | 1 = Echo on if enabled.<br>0 = Echo disabled. |
| 7 | Not used. |
| 6 | 1 = Binary transfer.<br>0 = ASCII transfer. |
| 5-0 | LU number of terminal. |

Since bits 10-9 and bits 8-6 have distinct meanings, the function code can be described in terms of its first and second octal digits:

First digit of function code:

    0 = Non-transparent character mode
    2 = Transparent character mode
    3 = Program-initiated block read

Second digit of function code:

    0 = ASCII transfer without echo
    1 = Binary transfer without echo
    4 = ASCII transfer with echo
    5 = Binary transfer with echo

As an example, the following sequence writes seven characters from a buffer called IOUT to LU 41, then does a read of five words from the same LU to a buffer named INBUF. The "normal" mode (ASCII data, non-transparent mode, function code = 0) is used for the write. The read uses ASCII data with echo on, function code = 4):

```
          .
          .
          .
     ICODE = 2
     ICNWD = 41
     IBUFL = -7
     REG = EXEC(ICODE,ICNWD,IOUT,IBUFL)
     ICODE = 1
     IBUFL = 5
     ICNWD = 41 + 400B
     REG = EXEC(ICODE,ICNWD,INBUF,IBUFL)
          .
          .
          .
```

## Writing to a Display

Programming write requests to a display requires consideration of several points. The output is a string of ASCII characters from a buffer in which each word contains two characters. The buffer length must be specified by a positive integer if you count the words, or by a negative integer if you count the characters. In either case, the driver terminates the character string by supplying a carriage return and line feed.

The length of the buffer must be limited to terminal display width unless the terminal has an End-of-Line Wrap Around mode enabled.

The CR/LF at the end of the character string can be suppressed by using the underscore character (137 octal). In the non-transparent (normal) mode the underscore must be the last character in the buffer.

In non-transparent mode, if the underscore is the last character in the string, the underscore CR and LF are not sent to the terminal. In transparent mode, an underscore anywhere in the text string will suppress the CR/LF, but the underscore will be sent to the terminal.

NOTE

> If the buffer length is specified by a positive integer, an even number of characters is sent to the terminal, appending a space to the text if necessary. Take care to ensure that the underscore is the last character in the string and not followed by a space if suppression of CR/LF is desired.

Write requests can pass escape and control sequences to the terminal to control various terminal functions. Consult the terminal User's Manual for a description of these functions.

When writing in the binary mode (bit 6 = 1) all ESC characters are stripped from the character string except when the EQT subchannel is 3. This is useful when using the Graphics features of a 2648A terminal.

## Block Mode Keyboard Read

For keyboard read requests, the driver maintains a record of the terminal status: character, block line, or block page mode. Keep this status current by requesting "Update Terminal Status" whenever the status is changed or when the logical driver LDVR5 is attached to the physical driver for a specific port. For insurance, your programs can request Terminal Status Update at any time.

In block-mode keyboard reads, terminal transmissions are either line-by-line or page-by-page, depending on the terminal strapping. With page strapping, line separators (CR/LF) are passed to the user's buffer but the data terminator (RS) is not. With line strapping, embedded RS's are passed to the user's buffer but the data terminator (CR) is not. In both cases the unit separator (US) is passed to the user's buffer.

For terminal-enabled block read (function code first digit = 0 or 2), the user composes a line or page on the terminal display then presses the ENTER key on the keyboard. For program-enabled block read (first digit of function code = 3), the user also composes the line or page but the read completes when the program requests, without waiting for the ENTER key to be pressed.

For program-enabled block reads, the program must position the cursor properly for the read. When the program requests a block read, the read progresses from the current cursor position to the end of the current line or page, depending on terminal strapping. For terminal-enabled block reads, the ENTER key positions the cursor.

In terminal-enabled block mode, all prompts before input should have ESC, underscore as the last two characters printed. Transparent mode prints the underscore, but ESC suppresses the actual display of the underscore. The ESC-underscore places a non-displayed terminator after the prompt, preventing it from being transmitted along with the user input when the ENTER key is pressed.

In block mode reads, the text that is on the display (in the terminal's display memory) is what is sent to the computer. Control codes and escape sequences that do not print on the display will not be sent unless they are made visible when entered by using the DISPLAY FUNCTIONS key.

## Character Mode Keyboard Read

In character-mode keyboard-read requests, the terminal transmits one character each time a key is pressed. The record terminator is a CR and must be entered to complete the request (the terminator is not sent to the user's buffer). At reception of the CR the driver echoes CR/LF and completes the request, passing the transmission log (number of characters or words sent) to the B Register.

In non-transparent character mode, the driver processes the following special characters:

- o DEL (RUBOUT, 177 octal). Entering DEL (shift-underscore) deletes the current record and outputs a backslash (\), CR, and LF; this deletes the line and starts a new line.

- o BACKSPACE (10 octal). Pressing the BACKSPACE key deletes the last character; the cursor moves back one position. The BS is not sent to the user's buffer.

- o LINE FEED (12 octal). A line feed (LF) is echoed back to the display but is not sent to the user's buffer.

- o CONTROL D (4 octal). Entering CNTL-D (CNTL and D keys) terminates data transmission and sets status word (EOT5) bit 5 to a "1" with all zeros in the B-Register.

In the transparent mode, the special characters listed above are not processed by the driver but are passed to the user's buffer.

The echo bit (bit 8) in the control word turns on echo mode. If the terminal was enabled with echo on, echo remains off only for the read request in which bit 8 in the control word is "0". Echo has no meaning for write requests. If echo was specified as off when the terminal was enabled, the echo bit in a read request control word has no effect, and is assumed to be zero.

LDVR5 TO A CARTRIDGE TAPE UNIT

Read and write requests to the Cartridge Tape Unit (CTU) in a terminal are handled in the same manner as read and write requests to the keyboard/display. The request code (ICODE) value must be a 1 for read and a 2 for write. The control word (ICNWD) must contain the LU number of the CTU in bits 5 through 0. The keyboard/display, left CTU, and right CTU on one terminal all have different LU numbers. The LU number of the CTU is associated with EQT subchannel 1 for left, 2 for right.

Of the remaining bits in the control word, only bit 6 is recognized; "0" specifies an ASCII data transfer, "1" specifies a binary read or write. The function code, therefore, is either 1 or 0.

A buffer and buffer length must be specified. The buffer length is a positive number for words or a negative number for characters. The buffer length is limited to 128 words for binary writes and 127 words for ASCII writes. CTU writes of greater than these specified lengths are rejected; the driver assumes records that exceed the specified length can not be written, so read requests with longer buffer length are not rejected.

For example, to do a binary read of a 128-word record from a CTU, LU number 45, into a buffer named IBUFR, the following code could be used:

```
          .
          .
          .
     ICODE = 1
     ICNWD = 100B + 45
     IBUFL = 128
     REG = EXEC(ICODE,ICNWD,IBUFR,IBUFL)
          .
          .
          .
```

Note the following points pertaining to the use of LDVR5 with CTU's:

- o If the terminal is strapped for page-mode block reads, the BLOCK MODE switch must be latched out for CTU operation.
- o A binary read request reads (via the driver) a word (or character) string into a buffer of specific length. If the buffer is filled before an End-of-Record (EOF) is read, the driver ignores the remaining data and stops the CTU at the first EOR it reads. If an EOR is read before the specified length is filled, the CTU stops at the EOR. The CTU skips one record if the buffer length is specified as zero.

- A binary write request writes a word (or character) string from a buffer of specified length. The driver rejects requests for zero length, or for greater than the maximum length (128 words).

- Parity is not permitted with binary reads or writes.

- An ASCII read request reads a word (or character) string terminated by a carriage return. If the specified buffer length is filled before a CR is read, the driver ignores the remaining characters; however, a CR must still be read in order to complete the request.

- An ASCII write request writes a word (or character) string from a buffer of specified length (the driver supplies a CR/LF). The driver terminates the request if the string includes a CR, LF, or RS. The driver uses CR as the record terminator on input and the terminal uses LF as the terminator on output; an RS is passed to the driver when the CTU encounters a file gap.

- If the End-of-Tape (EOT) point is sensed during a write operation, an End-of-Data (EOD) mark is recorded automatically; the driver completes the current record. Further attempts to write will cause LDVR5 to reject the request. The occurrence of an EOT on write is reported to the user as an error (time-out); a dynamic status request should be used to determine the exact cause.

- Similarly, the driver reads the current record if an EOD is sensed during a read operation. Either condition (EOT or EOD) is reported in the status word (EQT5). Further attempts to read past EOT will cause LDVR5 to reject the request.

- Read requests are rejected if the tape is at EOD. However, the EOD can be overwritten (using a write request) with data or a file mark unless the tape is at EOT.

- Continuously moving the tape back and forth (re-reading) more than 100 times over a short section of the tape may permanently damage the tape cartridge. Periodically "conditioning" the tape prevents failures due to such back and forth motions. Refer to "Tape Conditioning Procedure" in your terminal User's Manual.

LDVR5 TO AN AUXILIARY PRINTER

Write requests to an Auxiliary Printer connected to a terminal are handled in the same manner as write requests to the keyboard/display. The request code (ICODE) value must be a 2 for write. The control word (ICNWD) must contain the LU number of the printer in bits 5 through 0. The remaining bits of the control word are ignored; only ASCII, non-transparent writes are allowed.

The keyboard/display and Auxiliary Printer connected to the terminal will have different LU numbers. The LU number of the printer is associated with EQT subchannel 4.

A buffer and buffer length must be specified. The buffer length can be positive, for words, or negative, for characters. The buffer length must not exceed the maximum line length of the printer. Refer to the printer User's Manual for line lengths.

The driver supplies the printer with a CR/LF at the end of each line (record). If the buffer contains a CR, LF, or RS, the driver terminates the request.

LDVR5 does not support the Reverse Line Feed and Plotting capabilities of the HP 9871A Printer; however, escape sequences to perform various functions on the 9871A can be passed.

For example, to print a line of text (80 characters) on an Auxiliary Printer on LU number 41, the following code could be used:

```
          .
          .
          .
    ICODE = 2
    ICNWD = 41
    IBUFL = -80
    REG = EXEC(ICODE,ICNWD,IBUFR,IBUFL)
          .
          .
          .
```

where IBUFR contains the 80 characters.

## RETURNS

Two information words are available when a read or write request completes. The A-Register contains the device status word and the B-Register contains the transmission log, equal to the length of the data in the user's buffer.

On write requests, however, the information is meaningless if the device is buffered. Device buffering refers to a buffer in addition to user's buffer IBUFR. Device buffering is established at system generation time or on-line by the system EQ command.

When a write request is issued to a buffered device, the user's buffer is transferred into a buffer in System Available Memory (SAM) and the output processing started. Control is returned to the user program before the actual output is finished, so the device status and transmission log are meaningless.

After read requests, and after write requests when the I/O device is unbuffered, the A- and B-Registers contain the completion status.

### Getting Completion Status

To get completion status from the A- and B-Registers in a FORTRAN program, call EXEC as a function. The following code shows how to use this method:

```
      DIMENSION IREG(2)
      EQUIVALENCE (REG,IREG),(IREG,IA),(IREG(2),IB)
          .
          .
          .
      ICODE = 1
      ICNWD = LU + 400B
      IBUFL = -80
      REG = EXEC(ICODE,ICNWD,IBUFR,IBUFL)
          .
          .
          .
```

The variables IA and IB now contain the A- and B-Register contents, device status word and transmission log respectively.

An alternative method is to use the library subroutine ABREG:

```
              .
              .
              .
        CALL EXEC(ICODE,ICNWD,IBUFR,IBUFL)
        CALL ABREG(IA,IB)
              .
              .
              .
```

IA and IB contain the A- and B-Register contents, provided ABREG is called immediately after the EXEC call.

## Status Word

The status word in the A-Register indicates the end-of-operation status as defined by the driver completion code in bits 15 and 14 of the word. This is either "00" (up) or "01" (down). The equipment type code, in bits 13 through 8, will be 00 octal. Bits 7 through 0 contain the physical status of the device or port.

The status word is word 5 of the Equipment Table Entry for the device. Appendix A describes the Equipment Table Entry. The meaning of word 5 (EQT5) is described for ISTA1 under "I/O Status" in this Section.

## Transmission Log

The transmission log is a positive integer equal to the character or word count of the data in the user's buffer. If the value of buffer length parameter IBUFL in the preceding EXEC call was positive, the transmission log in the B-Register is the word count. If the value of IBUFL was negative, the transmission log is the character count.

## LOGICAL AND PHYSICAL DRIVER CONTROL REQUESTS

Control requests that apply to both the logical and physical drivers include setting the EOT bit in a device status word, setting speed sense mode to enable the Multiplexer to detect the baud rate of a terminal, and setting a port to the buffer flush condition. Also in this category is the disable terminal request.

This subsection describes the above requests, and also describes the CR/LF delay request, applicable only to the default TTY driver. These are control requests, performed by calling EXEC with the first parameter (ICODE) set to 3 and the second parameter (ICNWD) containing the function code in bits 10 through 6 and the logical unit number (LU) in bits 5 through 0, as described under "Control Word" at the beginning of this Section.

### SET EOT STATUS

Issuing a control request with a function code of 7 octal sets the End-of-Tape status bit in the device status word (EQT5 bit 5). This has the same effect as reading an End-of-Tape (EOT) from a CTU, or entering a CNTL-D from the keyboard in the non-transparent mode. The following code is an example of setting EOT status on LU 41:

```
          .
          .
          .
     ICODE = 3
     ICNWD = 700B + 41
     REC = EXEC(ICODE,ICNWD)
          .
          .
          .

          -- or --

               :CN,41,7B
```

Set EOT status is supported by both the default TTY driver and LDVR5.

### SET CR/LF DELAYS

Carriage return and line feed delays are required by some printers to allow time for the carriage or print head to return to the left margin before starting to print the next line. Also, some terminals require a delay to avoid overflowing the internal buffer.

A symptom of too short a CR/LF delay on a printing terminal is printing a character between the normal lines of text on a carriage return. HP 2640-series terminals display a half-bright rectangular "DEL" character (octal 177) if the internal buffer overflows. Consult the device User's Manual for CR/LF requirements.

One CR/LF delay is always sent. Additional delays can be set in increments of one character time (the transmission time for one character at the baud rate of the terminal) by specifying a number of stall characters for carriage return and line feed.

To set the delays, use a control request with a function code of 12 octal. One additional parameter is required (IDELY) in which bits 7 through 4 contain the number of stall characters for CR and bits 3 through 0 contain the number of stall characters for LF.

For example, to set a CR delay of 4 and an LF delay of 3 on LU 41, the following code could be used:

```
              .
              .
              .
      ICODE = 3
      ICNWD = 1200B + 41
      ICR = 16 * 4
C
C  MULTIPLY BY 16 = SHIFT LEFT 4
C
      ILF = 3
      IDELY = IOR(ICR,ILF)
      REG = EXEC(ICODE,ICNWD,IDELY)
              .
              .
              .

            -- or --

                 :CN,41,12B,103B
```

Setting CR/LF delays is only supported by the default TTY driver and is not used when LDVR5 is attached to DVS00. For this purpose, LDVR5 uses an enquiry/acknowledge handshake.

## SET SPEED SENSE MODE

Using a control request with a function code of 17 octal, it is possible to have the Multiplexer hardware detect the operating baud rate of a terminal and report that baud rate to the driver. The reported baud rate is established as both the transmit and receive baud rates by the driver.

One additional parameter is required (ICHR), to contain a "search for" character, right justified. The user must type the "search for" character in from the keyboard within one minute after issuing the request (other characters are ignored).

On receipt of the request, the driver puts the selected port into the Speed Sense (diagnostic) mode for one minute; if the correct character is not input within that amount of time, the driver times out, setting bit 4 of the status word to "1" to signify time-out, and setting the baud rate to 2400.

If the correct character is input from the port, the sensed speed is set as both the transmit and receive speeds, and the driver makes a normal completion. The speeds sensed are 110, 150, 300, 600, 1200, and 2400 baud. Since both the send and the receive rates are set, split-speed operation is not possible, unless the transmit rate is reset upon completion.

For example, to sense the baud rate on LU 41, using the CR character (user must press RETURN) the following code could be used:

```
               .
               .
               .
          ICNWD = 1700B + 41
          ICHR = 15B
     C  15 OCTAL IS CARRIAGE RETURN, USER MUST HIT RETURN
     C  WITHIN ONE MINUTE
          REG = EXEC(ICODE,ICNWD,ICHR)
               .
               .
               .

               -- or --

                    :CN,41,17B,15B
```

The speed sense mode is supported by both LDVR5 and the default TTY driver.

DISABLE TERMINAL

A control request with a function code of 21 octal will disable the terminal assigned to the addressed LU number. When a terminal is disabled, bit 0 of the status word is cleared. This means that striking a key on the keyboard will not schedule the program that was specified at generation. Read, write, and control requests to the terminal will still function normally. To re-enable the terminal an "Enable Terminal" request must be issued.

For example, to disable LU 41:

```
        .
        .
        .
   ICODE = 3
   ICNWD = 2100B + 41
   REG = EXEC(ICODE,ICNWD)
        .
        .
        .

        -- or --

             :CN,41,21B
```

Disable terminal may be used with either the default TTY driver or LDVR5.

## BUFFER FLUSH

Issuing a control request with a function code of 23 octal will put a port in the buffer flush condition. In this condition, the driver ignores all write requests and control requests for the designated port. Read requests only are honored.

To establish the buffer flush condition, the driver sets bit 7 of the device status word. Bit 7 is defined for both the default TTY driver and for LDVR5 to indicate the buffer flush condition. See "I/O Status" for a description of related status conditions.

The buffer flush condition is removed automatically when a read request is processed or the queue (a list of programs with requests pending against the EQT) is empty. The buffer flush condition may also be removed by issuing the "Buffer Unflush" control request.

NOTE

> A buffer flush condition may also be established by a communication line failure.

To establish a buffer flush condition the following example could be used:

```
        .
        .
        .
ICODE = 3
ICNWD = 2300B + 41
REG = EXEC(ICODE,ICNWD)
        .
        .
        .

        -- or --

            :CN,41,23B
```

A buffer flush condition can be established while using either the default TTY driver or LDVR5.

BUFFER UNFLUSH

To remove a buffer flush condition, issue a control request with a function code of 24 octal. Buffer unflush restores the use of write requests and control requests after a buffer flush, and clears the buffer flush bit (bit 7) in the status word. It also clears the bad communications line, time-out, and EOT bits (bits 2, 4, and 5, as listed in the "I/O Status" subsection).

A buffer unflush should be used in the recovery from a communication line failure. Either a read request or queue empty initiates an automatic buffer unflush, performing the same function as a buffer unflush request.

To execute a buffer unflush, the following example could be used:

```
          .
          .
          .
     ICODE = 3
     ICNWD = 2400B + 41
     REG = EXEC(ICODE,ICNWD)
          .
          .
          .

          -- or --

               :CN,41,24B
```

A buffer unflush can be used under either the default TTY driver or LDVR5.

NOTE

> If the communications line is still bad, the next request to that port may set the port down again.

## LDVR5 CONTROL REQUESTS

The control requests described in this subsection are used with LDVR5 only. Since they pertain to special functions of LDVR5, they are not recognized by the default TTY driver.

### SET TIME-OUT

To set the RTE device time-out value to something other than the value that was established at generation, use a control request with a function code of 22 octal. This value can be set in 100-millisecond intervals by the integer provided in an additional parameter.

Time-out values can also be set by using TO operator command. The RTE Programmer's Reference Manual for your system describes how to use the TO command. When you use a control request or the File Manager CN command, specify the port, or device, by LU number; when you use the TO operator command, specify the port by EQT number.

If the default TTY driver is used, the only method of setting the time-out value is using the RTE TO command. Time-outs set by LDVR5 remain in effect if the default driver is reenabled.

To set the time-out on LU 41 to 25 seconds, the following example can be used, if LDVR5 is attached to the physical driver:

```
        .
        .
        .
   ICODE = 3
   ICNWD = 2200B + 41
   ITO = 10 * 25
   REG = EXEC(ICODE,ICNWD,ITO)
        .
        .
        .

        -- or --

            :CN,41,22B,250
```

## CTU CONTROL

There are nine functions that are controllable on each Cartridge Tape Unit (CTU) of an HP terminal. These functions can be performed by issuing a control request in which the control word (ICNWD) contains the LU number of the CTU and a function code from the list given in Table 2-4.

Table 2-4. CTU Control-Request Function Codes

| Function Code | Description |
|---|---|
| 01 | Write End-of-File (EOF) |
| 02 | Backspace one record |
| 03 | Forward space one record |
| 04 | Rewind |
| 10 | Write End-of-File (EOF) if an EOF was not just previously written on this CTU |
| 13 | Forward space one file |
| 14 | Backspace one file |
| 26 | Write End-of-Data |
| 27 | Locate absolute file: parameter required, IFILE $0 \le IFILE \le 256$ |

A rewind, backspace one record, or backspace one file request will not cause any action if the CTU is at the load point. The load point status can be reported in a dynamic status word, as described for the "Dynamic Status" request. A request for a file number greater than any existing on the tape will position the tape to the last file that exists. If the terminal is strapped for page mode operation, the BLOCK MODE switch must be in the out position.

As an example, the following code could position the CTU assigned to LU 42 to the start of the first record in file 6:

```
     .
     .
     .
ICODE = 3
ICNWD = 2700B + 42
IFILE = 6
REG = EXEC(ICODE,ICNWD,IFILE)
     .
     .

     -- or --

          :CN,42,27B,6
```

PRINTER CONTROL

A control request with a function code of 11 octal issued to an LU that is assigned to an Auxiliary Printer (subchannel 4) connected to an HP terminal causes the printer to advance the paper a specified number of lines.

The number of lines must be specified in an additional parameter, with a value of less than 256. A negative number in the additional parameter will cause some printers to go to top-of-form, other printers will ignore negative parameters. Consult your printer manual for reaction to negative spacing parameters.

As an example, the following code would execute a top-of-form on a printer referenced by LU number 44:

```
     .
     .
     .
ICODE = 3
ICNWD = 1100B + 44
ILINE = -1
REG = EXEC(ICODE,ICNWD,ILINE)
     .
     .

     -- or --

          :CN,44,11B,-1
```

The above commands can be used to space a display, with a maximum limit of 55 lines, by using the display LU.

## MODEM CONTROL

When communicating with a terminal over a full-duplex modem link, control of the modem is relatively simple. To establish communication, the Multiplexer must issue a "Data Terminal Ready" to the modem and then wait for the modem to respond.

The modem uses two signal lines to signal the Multiplexer that the link has been established. "Data Set Ready" is on whenever the modem is turned on and connected to the line, and "Clear To Send" goes on when communication has been established with the modem at the terminal end. The corresponding communications at the terminal end are taken care of automatically by the modem and the terminal.

### OPEN LINE

A control request with a function code of 31 octal initiates a modem connect sequence on one port. The Multiplexer issues the "Data Terminal Ready" and the call does not complete until the Multiplexer receives "Data Set Ready" and "Clear To Send", or the port times out.

An additional parameter is needed to set the time-out for the Open Line call. This parameter must be specified as the negative of the number of one-tenth second intervals for time-out. A zero or a positive value specifies that the port will not time out while waiting for a modem line connection.

The time-out value must be long enough for the connect to complete. This time could be anywhere from a few seconds if the terminal end is tied in and ready to go, to a maximum time of 54 minutes if the terminal end must be powered up, phone dialed in, and modems connected. The time-out specified replaces the system assigned time-out described under "Set Time-Out" for the Open Line request only.

## MODEM CONTROL

The following code is an example of opening a modem line on LU 41 with a 10 minute wait for a line connect:

```
          .
          .
          .
     ICODE = 3
     ICNWD = 3100B + 41
     ITO = -6000
     REG = EXEC(ICODE,ICNWD,ITO)
          .
          .
          .

          -- or --

              :CN,41,31B,-6000
```

An Open Line control request can be used with either the default TTY driver or with LDVR5. **The open line request is always executed, regardless of any buffer flush condition.**

## CLOSE LINE

To provide an orderly termination of modem line communications, use the Close Line control request, which specifies a function code of 30 octal. The Close Line request sets the "Data Terminal Ready" line down from the Multiplexer to the modem, causing the modem to initiate a disconnect from the phone line. The request waits five seconds to give the phone line time to break connection, then returns to the user program. No additional parameters are required.

The following code is an example of a disconnect on LU 41:

```
          .
          .
          .
     ICODE = 3
     ICNWD = 3000B + 41
     REG = EXEC(ICODE,ICNWD)
          .
          .
          .

          -- or --

              :CN,41,30B
```

The Close Line control request can be used with either the default TTY driver or LDVR5.

## STATUS AND ERROR HANDLING

There are two status calls available to the user to check on the status of a Multiplexer port for normal processing and error diagnostics. I/O Status can provide the user with two of the device status words (EQT5 and EQT4) and an LU status word. Dynamic Status provides the user with the status of the CTU's or Auxiliary Printer on a terminal.

### I/O STATUS

The I/O Status request, using a request code (ICODE) of 13, calls the RTE operating system to provide information contained in system tables. The LU number of the port must be specified in the control word (ICNWD). One additional parameter is required and two more are optional. One, two, or three words are returned to the user's program in the parameters passed.

A sample calling sequence for LU 41 is shown below:

```
          .
          .
          .
     ICODE = 13
     ICNWD = 41
     CALL EXEC(ICODE,ICNWD,ISTA1,ISTA2,ISTA3)
          .
          .
          .
```

When the call completes the variables ISTA1, ISTA2, and ISTA3 contain the I/O status as shown in Table 2-5. ISTA1 is the status word (EQT5).

Table 2-5. I/O Status-Request Returns

| Word | Bits | Description |
|---|---|---|
| ISTA1 | 15-14 | I/O controller availability indicator:<br>00 = Available for use<br>01 = EQT disabled (down)<br>10 = Device busy |
| | 13-8 | Equipment Type Code = 00 octal |
| | 7-0 | Status: 1 = |
| | 7 | Buffer Flush condition |
| | 6 | Break key hit |
| | 5 | Control D entered (EOT) |
| | 4 | Time-Out occurred |
| | 3 | In Speed-Sense mode |
| | 2 | Bad communications line (modem failure) |
| | 1 | Pause mode (key struck during output) |
| | 0 | Terminal Enabled |
| ISTA2 | 15 | Not used: always zero |
| | 14 | 1 = Automatic output buffering enabled |
| | 13 | 1 = Driver to process power fail (always 0) |
| | 12 | 1 = Driver to process time-out (always 1) |
| | 11 | 1 = System communication flag |
| | 10-6 | Last subchannel addressed |
| | 5-0 | Dummy select code assigned to Multiplexer port |
| ISTA3 | | Logical Unit Status: |
| | 15 | 1 = LU down, 0 = LU up |
| | 14-5 | Not used |
| | 4-0 | EQT subchannel associated with the LU number |

## DYNAMIC STATUS REQUEST

The dynamic status request to a CTU or Auxiliary Printer LU number is actually a read request (ICODE = 1) with a function code of 37 octal. As with other read requests, the control word (ICNWD) includes the function code and the LU number of the CTU or printer. The buffer need only be one word long and the buffer length must be specified as one.

An example dynamic status request on LU 42 is coded as follows:

```
     .
     .
     .
ICODE = 1
ICNWD = 3700B + 42
IBUFL = 1
REC = EXEC(ICODE,ICNWD,IBUFR,IBUFL)
     .
     .
     .
```

On return, the first word of the buffer (IBUFR) contains the dynamic status for the CTU or printer as specified in Table 2-6.

NOTE

> Because the dynamic status request is a call to the driver, it will wait until any outstanding request on that EQT is completed.

Table 2-6. Dynamic Status Request Response Lists

| Bit Set | Description |
|---|---|
| 15 - 8 | Not Used |
| 7 | End of File (EOF) |
| 6 | Tape at load point |
| 5 | End of Tape (EOT) |
| 4 | Hard read error or write error |
| 3 | Last command aborted, check other bits for cause; bit 3 significant for printers only |
| 2 | Cartridge write protected |
| 1 | End of data (EOD) |
| 0 | Cartridge not inserted or unit busy |

## FAILURE ANALYSIS

For failure analysis it is important to note that all errors cause time-outs. If specified in the "Enable Terminal" request, the device will also be downed, and an error message produced at the system console:

I/O TO L #x F #y S #z

where x, y, and z are the Logical Unit number, EQT number, and subchannel number respectively. If a device is down, the request will wait for the operator to UP the EQT number.

It may be that only a simple time-out has occurred, meaning that the operator was too slow in responding to a read request, or a modem link was not established in time. In either case, retry the request a **few times**.

To recover, always perform a Buffer Unflush, and re-enable the terminal before proceeding with operations.

If the device is not downed, control returns to the user program on error (time-out) and the user program should be structured to check for errors and process accordingly. You can use the "I/O Status" request to get the status to test for various conditions.

## Modem Errors

If a port has been using a modem link, and the communications line fails (disconnects), the driver automatically executes a line-close operation and puts the port in the buffer-flush state. When in this state, the port ignores all pending requests except buffer-unflush and line-open.

Unlike a programmatic buffer flush, the buffer-flush state caused by a line failure cannot be removed by a read request; and, when the flush is removed by the driver, any subsequent I/O request will return the port to the buffer-flush state.

The recovery procedure after a line failure has placed a port in the buffer-flush state depends on whether the port is down or up. If the port is down, either of two procedures can be followed.

If the port is down and all pending requests can be removed, proceed as follows:

a. Remove all requests pending against the EQT.

b. UP the EQT.

c. Issue an open line request.

d. Wait for the communications line to be reestablished.

e. Restart any aborted process.

If the port is down and a priority higher than that of any pending request is available, proceed as follows:

a. Issue an open line request at higher priority than any pending request.

b. UP the EQT.

c. Wait for the communications line to be reestablished.

If the port is not down:

a. Issue an open line request.

b. Wait for the communications line to be reestablished.

Note that I/O requests made while the line was down were flushed.

### CTU and Printer Errors

If the program was communicating with a CTU or Auxiliary Printer, a Dynamic Status request should be performed and the bits in the return word checked for error conditions. On the basis of these results the operator should be requested to correct the problem. If the device is down it must be UP´ed prior to performing the Dynamic Status request.

### Read Errors

If the last command issued was a read, a parity error or data overrun may have occurred. Data overrun means that the Multiplexer maximum throughput rate has been exceeded and data on the LU with the time-out bit set was lost. Neither parity error nor data overrun can be explicitly tested for by the user, so the best procedure is to retry the failed request a finite number of times.

### Program Response to Line Failure

When a line failure occurs on a port initiated not to be set down on errors, the driver responds to read requests with an EOT. Therefore, if a port does not set down on error, user programs should be written to handle EOT signals in a reasonable manner. For example, interactive programs might attempt a finite number of retries, then terminate in an orderly fashion.

Since the driver performs a line close operation when the line is broken, the line must be reopened before another call can be answered. This can be done manually, from another console, as described above, or it can be done programmatically.

To open closed lines programmatically, a line monitor program can be put in the time list to perform an open line request periodically on all modem ports. This program needs a priority high enough to ensure that it is not locked out by other programs. Figure 2-3 shows an example of such a line monitor program.

```
FTN4,L
      PROGRAM LINUP(3,41), * MUX LINE MAINTENANCE PROGRAM *
      DIMENSION IPORT(32)
C
C  THIS PROGRAM SCANS ALL MULTIPLEXER PORTS USING MODEMS
C  (AS DEFINED IN IPORT ARRAY) AND ISSUES AN OPEN LINE REQUEST
C  IF THE PORT IS NOT BUSY. IF THE PORT IS BUSY OR DOWN NO ACTION
C  IS TAKEN. THIS PREVENTS THIS PROGRAM FROM BEING SUSPENDED.
C  NO CHECK IS NEEDED FOR WHETHER THE LINE IS UP OR DOWN.  THE
C  DRIVER IGNORES OPEN REQUESTS TO PORTS WHOSE LINES ARE UP.
C
C  A CHECK IS ALSO MADE FOR TERMINALS LEFT DISABLED.  DISABLED
C  PORTS ARE GIVEN A BUFFER-UNFLUSH AND THEN ENABLED SO THAT
C  THE NEXT USER DIALING IN CAN SCHEDULE PRMPT (OR WHATEVER WAS
C  SPECIFIED AT GEN TIME).  IF THIS FUNCTION IS NOT DESIRED, THE
C  INDICATED LINES SHOULD BE REMOVED.
C
C  !!!! NOTE:  ALL PORTS MUST BE BUFFERED !!!!
C
C  THIS PROGRAM SHOULD BE PUT IN THE TIME LIST FOR SCHEDULING
C  EVERY 10-20 SECONDS.
C
C  DEFINE LU OF PORTS TO BE MONITORED;
C   CHANGE IPORT DATA TO MATCH YOUR SYSTEM'S LU'S;
C   CHANGE NUMPT TO THE NUMBER OF ELEMENTS IN IPORT:
C
      DATA IPORT/14,15,16,17,18,19,20,21/
      DATA NUMPT/8/
C
C  SCAN PORTS:
      DO 10 I=1,NUMPT
      LU = IPORT(I)
C
C  CHECK IF PORT IS BUSY OR DOWN:
      CALL EXEC(13,LU,ISTAT,IQ)
      IF(IAND(ISTAT,140000B) .NE. 0) GOTO 10
C
C  IF PORT IS NOT BUSY, CHECK TERMINAL ENABLED;  ENABLE IF
C  NOT ALREADY ENABLED;  OMIT THIS CODE IF TERMINAL ENABLE
C  IS CONTROLLED BY APPLICATION PROGRAM:
      IF(IAND(ISTAT,1) .NE. 0) GOTO 20
      CALL EXEC(3,2400B+LU)
      CALL EXEC(3,2000B+LU,0)
C
C  ISSUE LINE OPEN FOR 10 MINUTES:
20    CALL EXEC(3,3100B+LU,-6000)
C
C  SCAN TO NEXT PORT:
10    CONTINUE
      END
END$
```

Figure 2-3. Line Monitor Program Example

# DEVICE EQUIPMENT TABLE

The HP 91731A Multiplexer Subsystem requires a device Equipment Table (EQT) Entry for each port on the Multiplexer. The entry consists of 15 words plus an extension of 11 words, or a total of 26 words. The EQT Entry is configured into the RTE Operating System at system generation time.

During system operation, the logical and physical drivers get the port configuration instructions and pass information to each other through the EQT Entry for the port. Table A-1 gives the function of each word in the Equipment Table Entry.

Table A-1. Equipment Table Entry (part 1)

| EQT Word Numbers | Use |
|---|---|
| 1 - 10 | Described in the RTE Programmer's Reference Manual for your system; word 5 is the status word. |
| 11 | Used by DVS00 for temporary storage. |
| 12 | Has the number of extension words requested at system generation time. After initialization this word contains the count for the logical driver time-out counter. |
| 13 | Extension starting address. |
| 14 | System time-out value; set at generation, or by system TO command, or, when using LDVR5, by a control request with a function code of 22 octal. |
| 15 | System time-out counter, set by the physical driver when either or both the physical or logical driver needs a time-out. |
| 16 | Count of the physical time-out counter. |
| 17 | The address of the logical driver. Initially set to the default TTY logical driver incorporated within physical driver DVS00. A control EXEC call subfunction 25B will set EQT17 to a new address. |

Table A-1. Equipment Table Entry (part 2)

| EQT Word Number | Use |
|---|---|
| 18 | If, at generation time, the port assigned to this EQT is specified to schedule a program on interrupt, EQT18 contains the address of the ID segment of that program. If not, EQT18 contains a negative one (-1). This specification is made in the Interrupt Table portion at generation time. |
| 19 | Carriage return, line feed delays, and extended status are defined as follows:<br><br>Bits — Meaning (Set)<br>15 - 12 — Value of character delays (stalls) for carriage return.<br>11 - 8 — Value of character delays (stalls) for line feed.<br>7 — Read/Write stall flag.<br>6 — Not used.<br>5 — Time-out caused by good completion<br>4 — 1 = Read, 0 = Write Request.<br>3 — Modems enabled<br>2 — Backspace mode.<br>1 — Line Feed Delay mode.<br>0 — Carriage Return Delay mode. |
| 20 | Stop bits for padding the output character = 43400B for no parity or even parity and = 43600B for odd parity. |
| 21 | Line state and output character; the bits are defined as follows:<br><br>Bits — Meaning (Set)<br>15 — Logical time-out.<br>14 — Line error.<br>13 — Data error.<br>12 — Break condition.<br>11 - 8 — Not used.<br>7 - 0 — Next character to be output. |

Table A-1. Equipment Table Entry (part 3)

| EQT Word Number | Use |
|---|---|
| 22 | Use defined as follows:<br><br>Bits — Meaning<br>15 — Always 1.<br>14 — Always 0.<br>13 — 0 = Down device on time-out.*<br>1 = Do not down device on time-out.*<br>12 — 1 = Do not echo underscore on backspace.<br>0 = Echo underscore on backspace.<br>11 - 9 — Not used.<br>8 — 0 = 1 Stop bit.<br>1 = 2 Stop bits.<br>7 — 0 = Echo off.<br>1 = Echo on.<br>5 & 4 — 0 = Parity off<br>1 = Parity odd.<br>2 = Parity even.<br>3 - 0 — Character size, not including stop bits and parity bit.<br>* All errors cause a time-out |
| 23 | The states of the control board control signals for clear-to-send (CB), and data-set-ready (CC).<br><br>Bits — Meaning<br>15 - 2 — Not used.<br>1 — 0 = CB down.<br>1 = CB up.<br>0 — 0 = CC down.<br>1 = CC up. |
| 24 | Contains the input port parameters, defined as follows:<br><br>Bits — Meaning<br>15 — Always 1.<br>14 — Always 0.<br>13 — 0 = Interrupts disabled.<br>1 = Interrupts enabled.<br>12 — 0 = Echo disabled.<br>1 = Echo enabled.<br>11 — 0 = Auto-speed not requested.<br>1 = Auto-speed requested.<br>10 - 8 — Total number of bits/character, (modulo 8)-1, 3 bits allocated for character size.<br>7 - 0 — Port baud rate. |

Table A-1. Equipment Table Entry (part 4)

| EQT Word Number | Use |
|---|---|
| 25 | Contains the output port parameters. The definition of the bits is the same as for EQT24 with the exception bit 14 which is a 1, and bit 12 for parity; see the definition of EQT word 20 for even-odd parity. |
| 26 | Contains the line state, auto-speed condition, and line open and close information.<br><br>Bits — Meaning<br>15 — Set when auto-speed detect is in progress.<br>14 - 13 — Not used.<br>12 — 1 = Not operating with modems.<br>0 = Operating with modems.<br>11 - 9 — Not used.<br>8 — Auto-speed completion:<br>0 = Not completing auto-speed request.<br>1 = Completing auto-speed request.<br>7 - 2 — Not used.<br>1 - 0 — Line State:<br>0 = Line down (disconnected)<br>1 = Line up, last driver operation: receive.<br>2 = Line up, last driver operation: send. |

## READER COMMENT SHEET

**HP 91731A ASYNCHRONOUS MULTIPLEXER SUBSYSTEM**
**User's Guide**

91731-90001 DEC 1978

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

**Is this manual technically accurate?**

**Is this manual complete?**

**Is this manual easy to read and use?**

**Other comments?**

---

**FROM:**

**Name** ______________________________

**Company** ______________________________

**Address** ______________________________

______________________________

______________________________

FOLD FOLD

FIRST CLASS
PERMIT NO. 141
CUPERTINO
CALIFORNIA

**BUSINESS REPLY MAIL**

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

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

FOLD FOLD

PART NO. 91731-90001
Printed in U.S.A. 12/78

HEWLETT PACKARD

Sales and service from 172 offices in 65 countries
11000 Wolfe Road, Cupertino, California 95014