digital

EK-DPV11-TM-002

DPV11 serial synchronous interface technical manual

digital equipment corporation ● merrimack, new hampshire

1st Edition, July 1980
2nd Printing (Rev), November 1980

The material in this manual is for informational purposes and is subject to change without notice.

Digital Equipment Corporation assumes no responsibility for any errors which may appear in this manual.

Printed in U.S.A.

This document was set on DIGITAL's DECset-8000 computerized typesetting system.

CONTENTS

CONTENTS (Cont)

CONTENTS (Cont)

ILLUSTRATIONS

ILLUSTRATIONS (Cont)

TABLES

PREFACE

This manual was written to satisfy the needs of Field Service and Educational Service Training personnel.

It contains the following categories of information.

- General description including features, specifications, and configurations
- Installation
- Programming
- Technical Description
- Maintenance

The manual also contains four appendixes which include diagnostic information, integrated circuit descriptions, and programming examples.

The DPV11 Field Maintenance Print Set (MP00919) contains useful additional information.

CHAPTER 1
INTRODUCTION

1.1 SCOPE

This chapter contains introductory information about the DPV11. It includes a general description, and a brief overview of the DPV11 operation, features, general specifications, and configurations.

1.2 DPV11 GENERAL DESCRIPTION

The DPV11 is a serial synchronous line interface for connecting an LSI-11 bus to a serial synchronous modem that is compatible with EIA RS-232-C interface standards and EIA RS-423-A and RS-422-A electrical standards. EIA RS-422-A compatibility is provided for use in local communications only (timing and data leads only). The DPV11 is intended for character-oriented protocols such as BISYNC, byte count-oriented protocols such as DDCMP, or bit-oriented data communication protocols such as SDLC. The DPV11 does not provide automatic error generating and checking for BISYNC.

The DPV11 consists of one double-height module and may be connected to an EIA RS-232-C modem by a BC26L-25 (RS-232-C) cable.

The DPV11 is a bus request device only and must rely on the system software for service. Interrupt control logic generates requests for the transfer of data between the DPV11 and the LSI-11 memory by means of the LSI-11 bus. (Figure 1-1 shows the DPV11 system.)

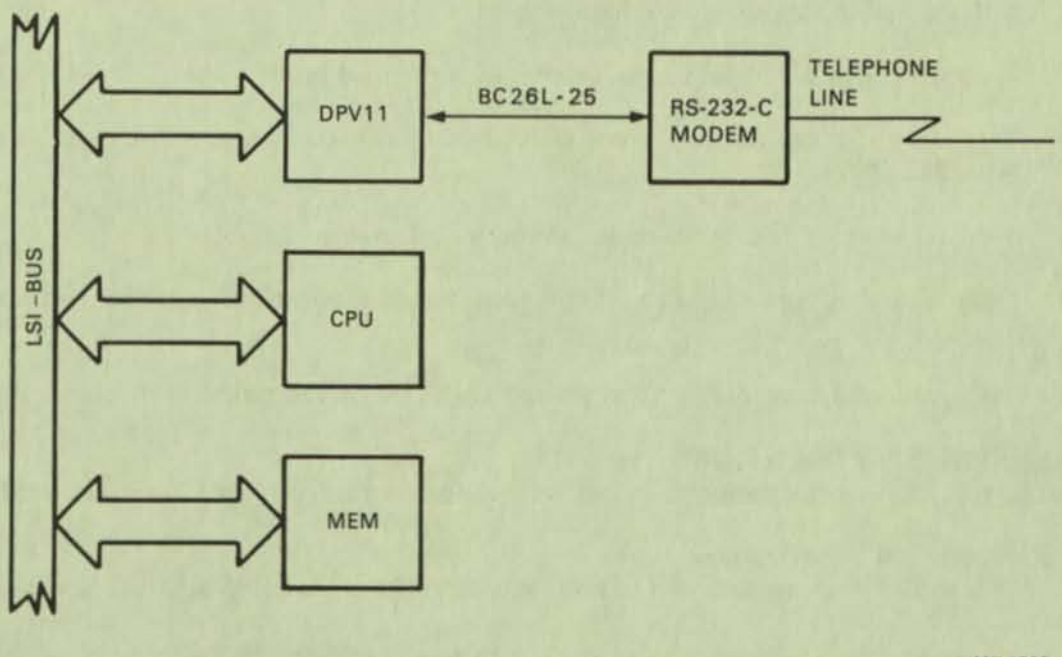


Figure 1-1 DPV11 System

1.3 DPV11 OPERATION

The DPV11 is a double-buffered program interrupt interface that provides parallel-to-serial conversion of data to be transmitted and serial-to-parallel conversion of received data. The DPV11 can operate at speeds up to 56K b/s.* It has five 16-bit registers which can be accessed in word or byte mode. These registers are assigned a block of four contiguous LSI-11 bus word addresses that start on a boundary with the low-order three bits being zeros. This block of addresses is jumper-selectable and may be located anywhere between 160000_8 and 177776_8. Two of these registers share the same address. One is accessed during a read from the address, the other during a write to the address. For a detailed description of each of the five registers, refer to Chapter 3. These registers are used for status and control information as well as data buffers for both the transmitter and receiver portions of the DPV11.

1.4 DPV11 FEATURES

Features of the DPV11 include:

- Full-duplex or half-duplex operation
- Double-buffered transmitter and receiver
- EIA RS-232-C compatibility
- All EIA RS-449 Category I modem control
- Partial Category II modem control to include incoming call, test mode, remote loopback, and local loopback
- Program interrupt on transitions of modem control signals
- Operating speeds up to 56K b/s (may be limited by software or CPU memory)
- Software-selectable diagnostic loopback
- Operation with bit-, byte count-, or character-oriented protocols
- Internal cyclic redundancy check (CRC) character generation and checking (not usable with BISYNC)
- Internal bit-stuff and detection with bit-oriented protocols.
- Programmable sync character, sync insertion, and sync stripping with byte count-oriented protocols.
- Recognition of secondary station address with bit-oriented protocols.

1.5 GENERAL SPECIFICATIONS

This paragraph contains environmental, electrical, and performance specifications for the DPV11.

1.5.1 Environmental Specifications

The DPV11 is designed to operate in a Class C environment as specified by DEC Standard 102 (extended).

Operating Temperature	5° C (41° F) to 60° C (140° F)
Relative Humidity	10% to 90% with a max. wet bulb temperature of 28° C (82° F) and a min. dew point of 2° C (36° F)

* The actual speed realized may be significantly less because of limitations imposed by the software and/or CPU memory refresh.

1.5.2 Electrical Specifications

The DPV11 requires the following voltages from the LSI-11 bus for proper operation.

+12 V at 0.30 A max. (0.15 A typical)
+5 V at 1.2 A max. (0.92 A typical)

The interface includes a charge pump to generate a negative voltage required to power the RS-423-A drivers.

The DPV11 presents 1 ac load and 1 dc load to the LSI-11 bus.

1.5.3 Performance Parameters

Performance parameters for the DPV11 are listed as follows.

Operating Mode	Full or half-duplex
Data Format	Synchronous BISYNC, DDCMP, and SDLC
Character Size	Program-selectable (5–8 bits with character-oriented protocols and 1–8 bits with bit-oriented protocols)
Max. Configuration	16 DPV11 modules per LSI-11 bus
Max. Distance	15 m (50 ft) for RS-232-C. 61 m (200 ft) for RS-423-A/RS-422-A (Distance is directly dependent on speed, and 200 ft is a suggested average. See RS-449 specification for details.)
Max. Serial Data Rates	56K b/s (May be less because of software and memory refresh limitations.)

1.6 DPV11 CONFIGURATIONS

There are two DPV11 configurations, the DA and the DB.

DPV11-DA
Unbundled version consists of:
- M8020 module
- Module Configuration Sheet (EK-DPV11-CG)

DPV11-DB
Bundled version consists of:
- M8020 module
- H3259 turn-around connector
- BC26L-25 cable
- DPV11 User Manual (EK-DPV11-UG)
- LIB kit (ZJ314-RB)
- Field Maintenance Print Set (MP00919)

Turn-around connectors, cables and documentation may be purchased separately.

1.7 EIA STANDARDS OVERVIEW (RS-449/RS-232-C)

The most common interface standard used in recent years has been the RS-232-C. However, this standard has serious limitations for use in modern data communication systems. The most critical limitations are in speed and distance.

For this reason, RS-449 standard has been developed to replace RS-232-C. It maintains a degree of compatibility with RS-232-C to accommodate an upward transition to RS-449.

The most significant difference between RS-232-C and RS-449 is in the electrical characteristics of signals used between the data communication equipment (DCE) and the data terminal equipment (DTE). The RS-232-C standard uses only unbalanced circuits, while the RS-449 uses both balanced and unbalanced electrical circuits. The specifications for the types of electrical circuits supported by RS-449 are contained in EIA standards RS-422-A for balanced circuits and RS-423-A for unbalanced circuits. These new standards permit much greater transmission speed and will allow greater distance between DTE and DCE. The maximum transmission speeds supported by RS-422-A and RS-423-A circuits vary with cable length; the normal speed limits are 20K b/s for RS-423-A and 2M b/s for RS-422-A, both at 61 m (200 feet).

Another major difference between RS-232-C and RS-449 is that additional leads are needed to support the balanced interface circuits and some new circuit functions. Two new connectors have been specified to accommodate these new leads. One connector is a 37-pin Cinch used in applications requiring secondary channel functions. Some of the new circuits added in RS-449 support local and remote loopback testing, and stand-by channel selection.

CHAPTER 2
INSTALLATION

2.1 INTRODUCTION

This chapter provides all the information necessary for a successful installation and subsequent check-out of the DPV11. Included are instructions for unpacking and inspection, pre-installation, installation and verification of operation.

2.2 UNPACKING AND INSPECTION

The DPV11 is packaged in accordance with commercial packing practices. Remove all packing material and verify that the following are present.

M8020 module
H3259 turn-around connector
BC26L-25 cable
DPV11 User Manual (EK-DPV11-UG)
LIB kit (ZJ314-RB)
Field Maintenance Print Set (MP00919)

Inspect all parts carefully for cracks, loose components or other obvious damage. Report damages or shortages to the shipper immediately, and notify the DIGITAL representative.

2.3 PRE-INSTALLATION REQUIREMENTS

Table 2-1 (Configuration Sheet) provides a convenient, quick reference for configuring jumpers.

Table 2-1 Configuration Sheet

(W1–W2) Driver Attenuation Jumper

Driver	Normal* Configuration	Alternate* Option	Description
Terminal Timing	W1 to W2	Not connected	Bypasses attenuation resistor. Jumper must be removed for certain modems to operate properly.

(W3–W11) Interface Selection Jumpers

Input Signals	Normal* Configuration	Alternate* Option	Description
SQ/TM (PCSCR-5)	W5 to W6		Signal quality
		W7 to W6	Test mode
DM (DSR) (RXCSR-9)	Not connected	W10 to W9	Data mode return for RS-422-A

*Normal configuration is typically RS-423-A compatible. Alternate option is typically RS-422-A compatible.

Table 2-1 Configuration Sheet (Cont)

(W3-W11) Interface Selection Jumpers (Cont)

Output Signals	Normal* Configuration	Alternate* Option	Description
SF/RL (RXCSR-0)	W3 to W4		Select frequency
		W5 to W3	Remote loopback
Local Loopback	W8 to W9	Not connected	Local loopback
	Not connected	W8 to W11	Local loopback (alternate pin)

(W12–W17) Receiver Termination Jumpers

Receiver	Normal* Configuration	Alternate* Option	Description
Receive Data	Not connected	W12 to W13	Connects terminating resistor for RS-422-A compatibility
Send Timing	Not connected	W14 to W15	
Receive Timing	Not connected	W16 to W17	

(W18–W23) Clock Jumpers

Function	Normal* Configuration	Alternate* Option	Description
NULL MODEM CLK	W20 to W18		Sets NULL CLK MODEM CLK to 2 kHz.
		W21 to W18	Sets NULL MODEM CLK to 50 kHz.
Clock Enable	W19 to W21 W22 to W23	W19 to W21 W22 to W23	Always installed except for factory testing.

(W24–W28) Data Set Change Jumpers

Modem Signal Name	Normal* Configuration	Alternate* Option	Description
Data Mode (DSR)	W26 to W24	Not connected	Connects the DSCNG flip-flop to the respective modem status signal for transition detection.
Clear to Send	W26 to W25	Not connected	
Incoming Call	W26 to W27	Not connected	Note: W26 is input to DSCNG flip-flop
Receiver Ready (Carrier Detect)	W26 to W28	Not connected	

*Normal configuration is typically RS-423-A compatible. Alternate option is typically RS-422-A compatible.

Table 2-1 Configuration Sheet (Cont)

Device Address Jumpers

GND	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3
W29	W31	W30	W36	W33	W32	W39	W38	W37	W34	W35

NOTE
The address to which the DPV11 is to respond is daisy-chain jumpered to W29 (GND).

Vector Address Jumpers

D8	D7	D6	D5	D4	D3	Source
W43	W42	W41	W40	W44	W45	W46

NOTE
Vector address to be asserted is daisy-chain jumpered to W46.

NOTE
Table 2-1 shows the recommended normal and alternate jumpering schemes. Any deviation from these will cause diagnostics to fail and require restrapping for full testing and verification. It is recommended that customer configurations that vary from this scheme not be contractually supported.

Prior to installing the DPV11, perform the following tasks.

1. Verify that the following modem interface wire-wrap jumpers are installed (Figure 2-1).

   W26 to W25 to W24 to W28 to W27
   W22 to W23 and W19 to W21
   W18 to W20
   W5 to W6
   W3 to W4
   W8 to W9
   W1 to W2

   This is the shipped configuration. Some of these jumpers may be changed when the module is connected to external equipment for a specific application. The RS-423-A NULL MODEM CLK is set to 2 kHz as shipped.

2. Based on the LSI-11 bus floating vector scheme or user requirements, determine the vector address for the specific DPV11 module being installed and configure W40 through W46 accordingly (Table 2-2). The floating vector ranking is 22.

3. Based on the LSI-11 bus floating address scheme or user requirements, determine the device address range for the DPV11 module and configure W30 through W39 accordingly (Table 2-3). Devices may be physically addressed starting at 160000 and continuing through 177776; however, there may be some software restrictions. The normal addressing convention is as shown in Table 2-3. The floating address ranking is 44.

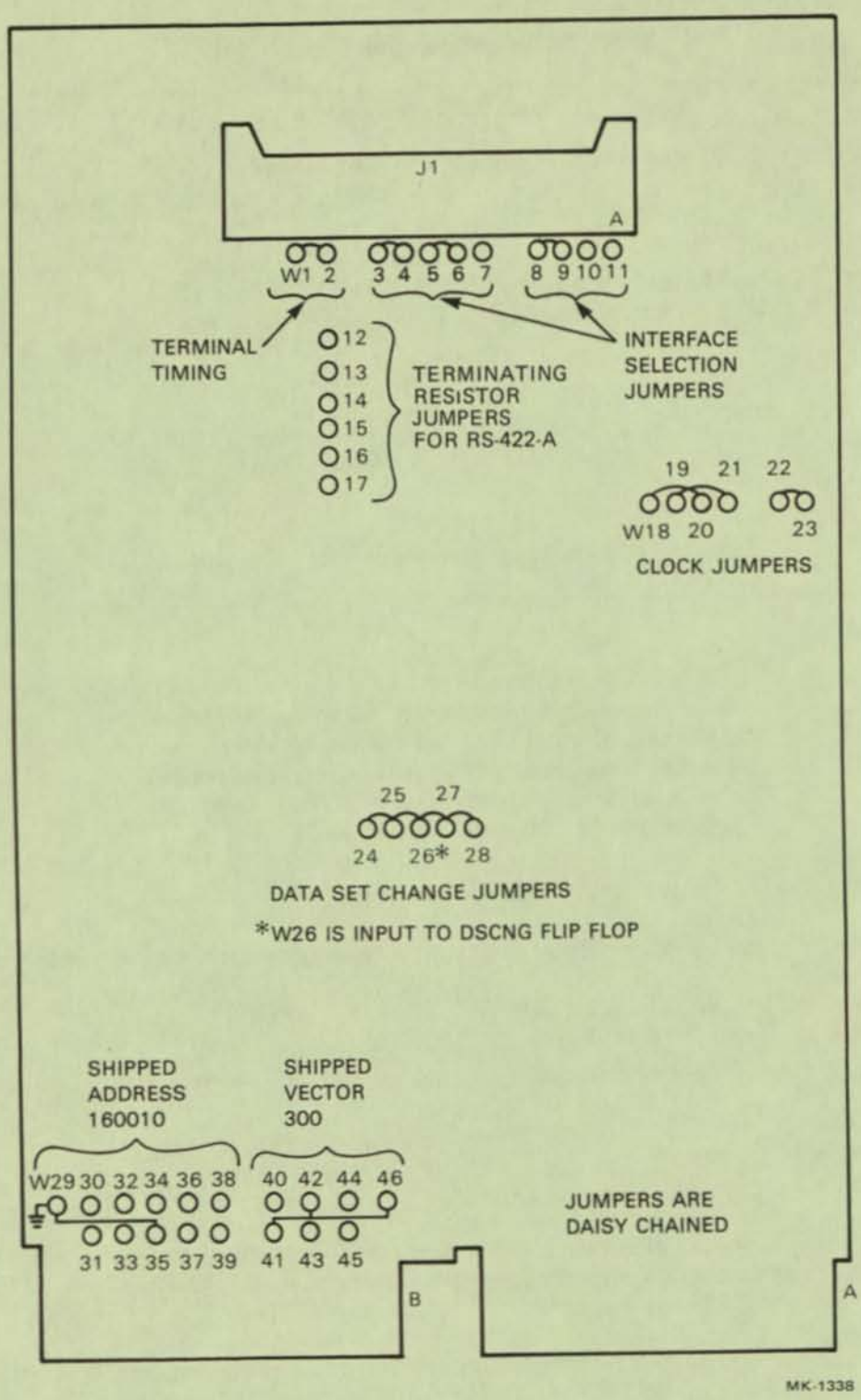


Figure 2-1 DPV11 Jumper Locations

Table 2-2 Vector Address Selection

DPV11 (M8020) VECTOR ADDRESSING

MSB LSB

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

JUMPER NUMBER	W43	W42	W41	W40	W44	W45	VECTOR ADDRESS
		X	X				300
		X	X			X	310
		X	X		X		320
		X	X		X	X	330
		X	X	X			340
		X	X	X		X	350
		X	X	X	X		360
		X	X	X	X	X	370
	X						400

	X		X				500

	X	X					600

	X	X	X				700

"X" INDICATES A CONNECTION TO W46.
W46 IS THE SOURCE JUMPER FOR THE VECTOR ADDRESS
JUMPERS ARE DAISY CHAINED.

MK-1341

Table 2-3 Device Address Selection

DPV11-XX (M8020) DEVICE ADDRESSING

MSB														LSB
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	← JUMPERS →										0	0	0

JUMPER NUMBER	W31	W30	W36	W33	W32	W39	W38	W37	W34	W35	DEVICE ADDRESS
										X	760010
									X		760020
									X	X	760030
								X			760040
								X		X	760050
								X	X		760060
								X	X	X	760070
							X				760100

						X					760200

						X	X				760300

					X						760400

					X		X				760500

					X	X					760600

					X	X	X				760700

				X							761000

			X								762000

			X	X							763000

		X									764000

"X" INDICATES A CONNECTION TO W29. W29 IS TIED TO GROUND. JUMPERS ARE DAISY CHAINED.

MK-1339

2.4 INSTALLATION

The DPV11 can be installed in any LSI-11 bus-compatible backplane such as H9270. LSI-11 configuring rules must be followed. Proceed with the installation as follows. For additional information refer to *PDP-11/03 User Manual* EK-LSI11-TM or *LSI-11 Installation Guide* EK-LSI11-IG.

1. Configure the address and vector jumpers at this time if they have not been previously done (Paragraph 2.3).

WARNING
Turn all power OFF.

2. Connect the female Berg connector on the BC26L-25 cable to J1 on the M8020 module † and plug the module into a dual LSI-11 bus slot of the backplane.

CAUTION

Insert and remove modules slowly and carefully to avoid snagging module components on the card guides.

3. Connect the H3259† turn-around connector to the EIA connection on the BC26L-25 cable. The jumper W1 on the H3259 turn-around connector must be removed.

4. Perform resistance checks from backplane pin AA2 (+5 V) to ground and from AD2 (+12 V) to ground to ensure that there are no shorts on the M8020 module or backplane.

5. Turn system power on.

6. Check the voltages to ensure that they are within the specified tolerances (Table 2-4). If voltages are not within specified tolerances, replace the associated regulator (H780 P.S.)

Table 2-4 Voltage Requirements

Voltage	Max.	Min.	Backplane Pin
+5 V	+5.25	+4.75	AA2
+12 V	12.75	+11.25	AD2

2.4.1 Verification of Hardware Operation

The M8020 module is now ready to be tested by running the CVDPV* diagnostic. Additional information on the DPV11 diagnostics is contained in Appendix A and Chapter 5. Proceed as follows.

NOTE

The * represents the revision level of the diagnostic.

1. Load and run CVDPV*. Three consecutive error-free passes of this test is the minimum requirement for a successful run. If this cannot be achieved, check the following.

   Board seating
   Jumper connections
   Cable connection
   Test connector

   If a successful run is still unachievable, corrective maintenance is required (see Chapter 5).

2. Load and run the DEC/X11 System Exerciser configured to test the number of DPV11s in the system.

   Each DEC/X11 CXDPV module will test up to eight consecutively addressed DPV11s.

   CXDPV uses a software switch register. Refer to the *DEC/X11 Cross-Reference* (AS-F055C-MC) for switch register utilization.

† If a BC26L-25 cable and H3259 turn-around connector are not available, an on-board test connector (H3260) can be ordered separately. See Paragraph 2.5.

The DEC/X11 System Exerciser is designed to achieve maximum contention with all devices that make up the system configuration. It is within this environment that the CXDPV module runs. Its intent is to isolate DPV11s which adversely affect the system operation.

For information on configuring and running the DEC/X11 System Exerciser, refer to *DEC/X11 User Manual* (AS-F0503B-MC) and *DEC-X11 Cross Reference* (AS-F055C-MC).

2.4.2 Connection to External Equipment/Link Testing

The DPV11 is now ready for connection to external equipment.

If the DPV11 is being connected to a synchronous modem, remove the H3259 connector and install the EIA connection of the BC26L-25 cable into the connector on the modem.

Configure jumpers W1–W28 in accordance with operating requirements (Table 2-1).

Load and run DCLT (CVCLH*) if a full link is available. This will check the final configuration and isolate failures to the CPU, the communications link, or the modem.

If the connection to external equipment uses RS-422-A, the user must provide the cable and test support.

2.5 TEST CONNECTORS

The only test connector provided with the DPV11 is the H3259 turn-around connector (Figure 2-2). Table 2-5 and Figure 2-3 show the relationship between pin numbers, signal names and register bits when the H3259 is connected by means of the BC26L-26 cable to the M8020 module.

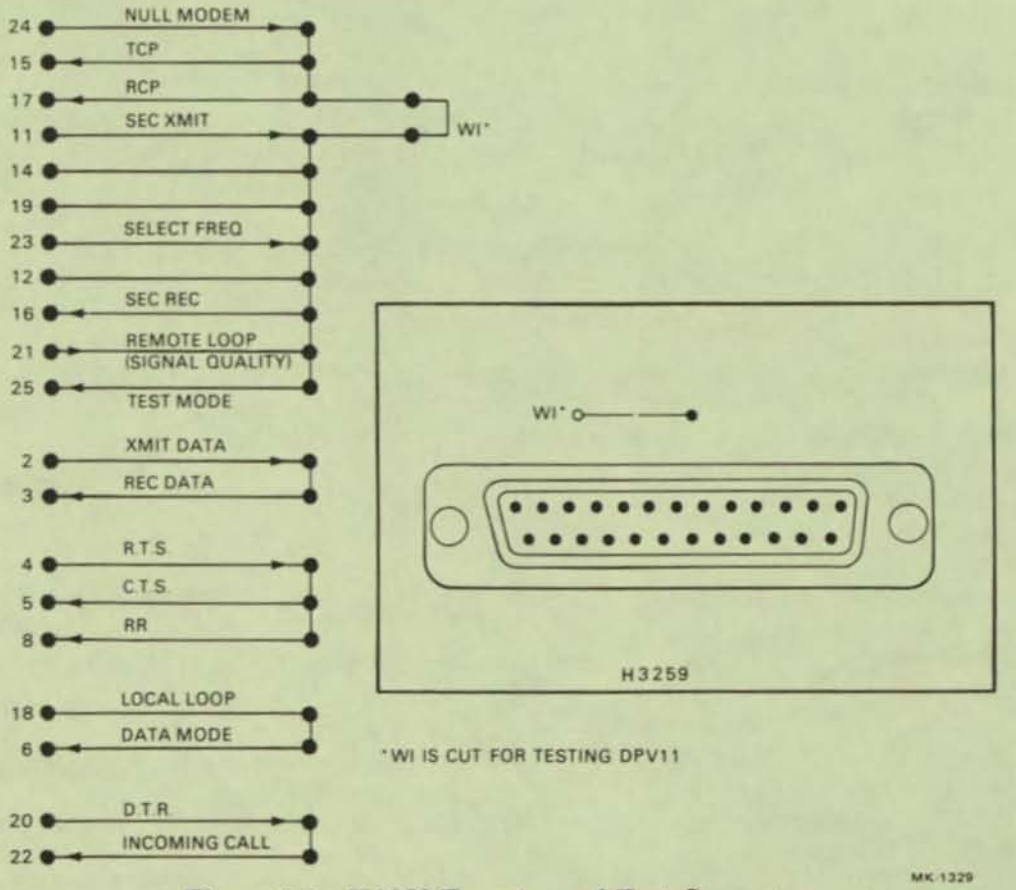


Figure 2-2 H3259 Turn-Around Test Connector

Table 2-5 H3259 Test Connections

From		To			
Signal Name	Pin No. H3259	Pin No. J1	Pin No. J1	Pin No. H3259	Signal Name
SEND DATA	2	F	J	3	RECEIVE DATA
REQUEST TO SEND (RTS) (RXCSR-2)	4	V	BB&T	5&8	CLEAR TO SEND (CTS)(RXCSR-13), RECEIVER READY (RR) (RXCSR-12)
LOCAL LOOPBACK (LL) (RXCSR-3)	18	U	Z	6	DATA MODE (DM) (RXCSR-9)
SELECT FREQ/REMOTE LOOPBACK (SF/RL) (RXCSR-0)	23/21	RR/MM	MM/C	21/25	SIGNAL QUALITY/ TEST MODE (SQ/TM) (PCSCR-5)
NULL MODEM	24	L	N&R	15&17	RCV CLOCK TX CLOCK
DATA TERMINAL READY (DTR) (RXCSR-1)	20	DD	X	22	INCOMING CALL (IC) (RXCSR-14)

The following accessories are available for interfacing and may be ordered separately.

- BC26L-X cable. Available in lengths of .3, 1.8, 2.4, 3.0, 3.6, 6.1, and 7.6 meters (1, 6, 8, 10, 12, 20 and 25 feet). When ordering, the dash number indicates the desired cable length in feet; e.g., BC26L-25 or BC26L-1.
- H3259 cable turn-around connector
- H856 Berg connector. Includes H856 Berg connector and 40 pins. Crimping tools are available from:

  Berg Electronics, Inc.
  New Cumberland, PA 17070

- H3260 on-board test connector (includes RS-422-A testing)

The H3260 on-board test connector (Figure 2-4) may be used to test the M8020 circuitry in its entirety. RS-422-A circuitry is not tested with the H3259 cable turn-around connector. The H3260 on-board test connector is shipped configured for testing RS-422-A. It may be configured to test RS-422-A or RS-423-A as follows.

RS-422-A	**RS-423-A**
W1–W2 out	W1–W2 installed
W3–W6 installed	W3–W6 out

The connector is installed into J1 with the jumper side up.

Since the H3260 on-board test connector does not test the cable, it is recommended that the DPV11 be tested with a turn-around connector at the modem end of the cable if possible.

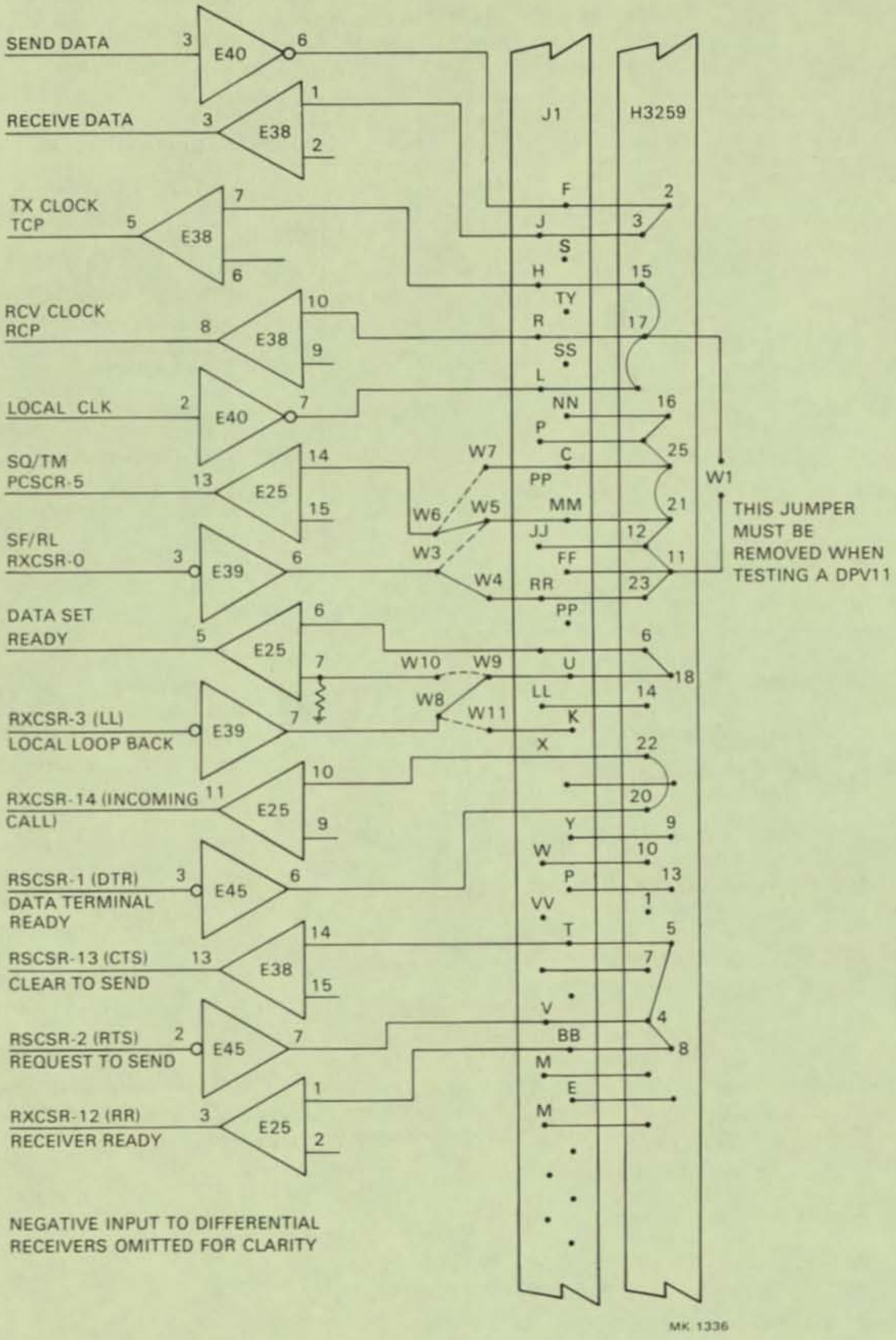


Figure 2-3 RS-423-A with H3259 Test Connector

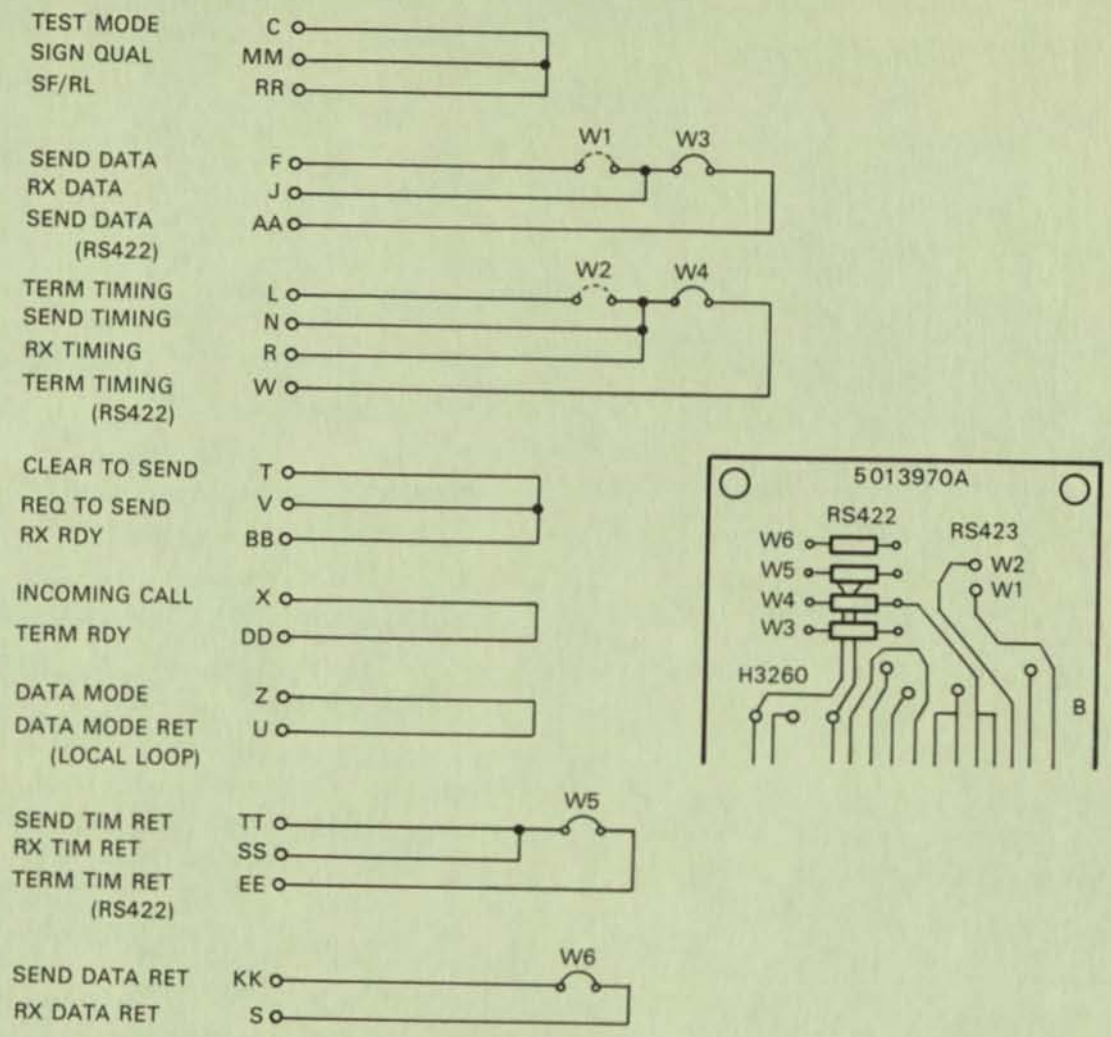


Figure 2-4 H3260 On-Board Test Connector

CHAPTER 3
REGISTER DESCRIPTIONS AND PROGRAMMING INFORMATION

3.1 INTRODUCTION

This chapter describes the bit assignments and programming considerations for the DPV11. Some typical start and receive sequences for both bit- and character-oriented protocols are included.

3.2 DPV11 REGISTERS AND DEVICE ADDRESSES

The five registers used in the DPV11 are shown in Table 3-1. Note that two of the registers (PCSAR and RDSR) have the same address. This does not constitute a conflict, however, because the PCSAR is a write-only register and the RDSR is a read-only register. These five registers occupy eight contiguous byte addresses which begin on a boundary where the low-order three bits are zero, and can be located anywhere between 160000_8 and 177776_8.

Table 3-1 DPV11 Registers

Register Name	Mnemonic	Address	Comments
Receive Control and Status	RXCSR	16xxx0	Word or byte* addressable. Read/write.
Receive Data and Status	RDSR**	16xxx2	Word or byte* addressable. Read-only.
Parameter Control Sync/Address	PCSAR**	16xxx2	Word or byte addressable. Write-only.†
Parameter Control and Character Length	PCSCR‡	16xxx4	Word or byte addressable. Read/write.
Transmit Data and Status	TDSR**	16xxx6	Word or byte addressable. Read/write.

* Reading either byte of these registers, clears data and certain status bits in other bytes. See Paragraphs 3.3.1 and 3.3.2.
** Registers contained within the USYNRT.
† It is not possible to do bit set or bit clear instructions on this register.
‡ The high byte of this register is internal to the USYNRT.

The DPV11 uses a universal-synchronous receiver/transmitter (USYNRT) chip which accounts for a large portion of the DPV11's functionality. The USYNRT provides complete serialization, deserialization and buffering of data to and from the modem.

Most of the DPV11 registers are internal to the USYNRT. Only the receiver control and status register (RXCSR) and the low byte of the parameter control and character length register (PCSCR) are external.

NOTE

When using the special space sequence function, all registers internal to the USYNRT must be written in byte mode.

3.3 REGISTER BIT ASSIGNMENTS

Bit assignments for the five DPV11 registers are shown in Figure 3-1. Paragraphs 3.3.1–3.3.5 provide a description of each register using a bit assignment illustration and an accompanying table with a detailed description of each bit.

3.3.1 Receive Control and Status Register (RXCSR) (Address 16xxx0)

Figure 3-2 shows the format for the receive control and status register (RXCSR). Table 3-2 is a detailed description of the register. This register is external to the USYNRT.

NOTE

The RXCSR can be read in either word or byte mode. However, reading either byte resets certain status bits in both bytes.

3.3.2 Receive Data and Status Register (RDSR) (Address 16xxx2)

Figure 3-3 show the format for the receive data and status register (RDSR). It is a read-only register and shares its address with the parameter control sync/address register (PCSAR) which is write-only. Table 3-3 is a detailed description of the RDSR.

NOTE

The RDSR can be read in either word or byte mode. However, reading either byte resets data and certain status bits in both bytes of this register as well as bits 7 and 10 of the RXCSR.

3.3.3 Parameter Control Sync/Address Register (PCSAR) (Address 16xxx2)

The parameter control sync/address register (PCSAR) is a write-only register which can be written in either byte or word mode. Figure 3-4 shows the format and Table 3-4 is a detailed description of the PCSAR. This register shares its address with the RDSR.

NOTE

Bit set (BIS) and bit clear (BIC) instructions cannot be executed on the PCSCR, since they execute using a read-modify-write sequence.

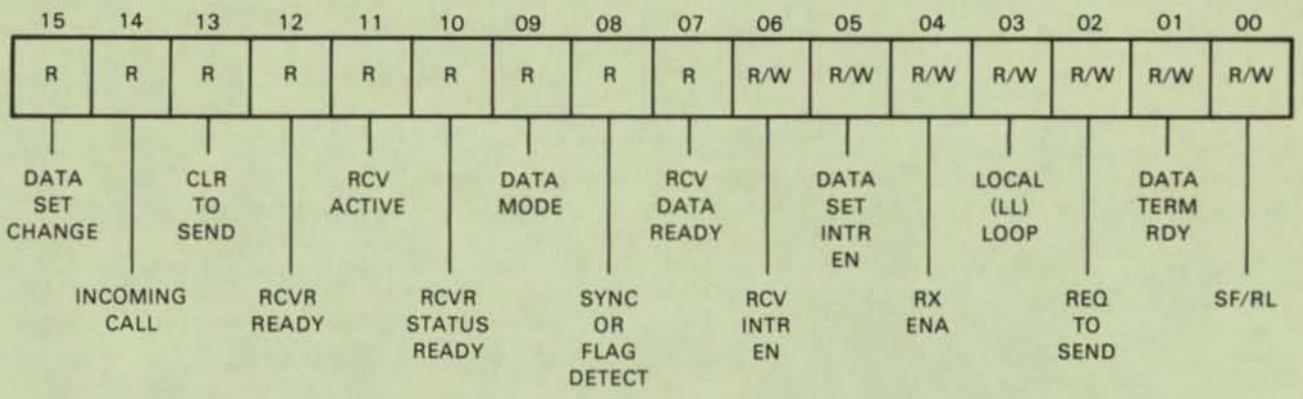
3.3.4 Parameter Control and Character Length Register (PCSCR) (Address 16xxx4)

The parameter control and character length register (PCSCR) can be read from or written into in either word or byte mode. The low byte of this register is external to the USYNRT and the high byte is internal. Figure 3-5 shows the format and Table 3-5 is a detailed description of the PCSCR.

3.3.5 Transmit Data and Status Register (TDSR) (Address 16xxx6)

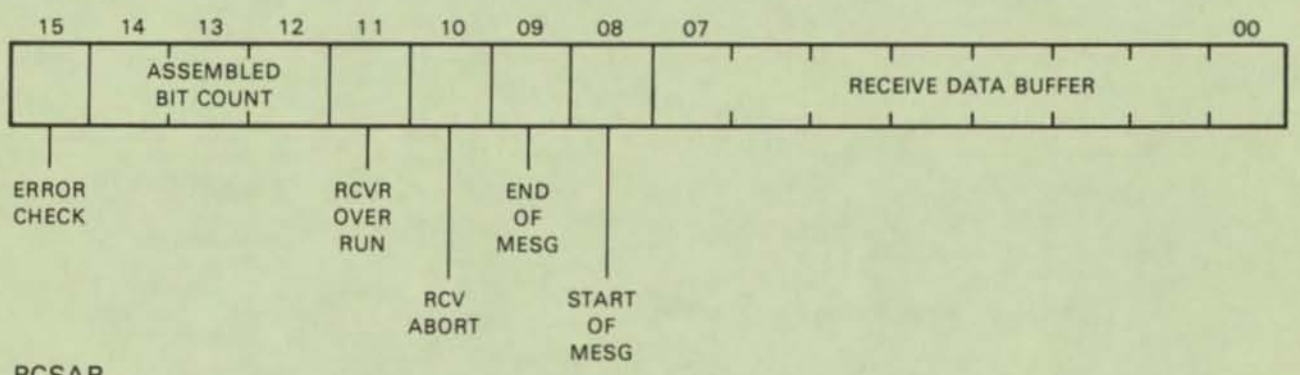
The format for the transmit data and status register (TDSR) is shown in Figure 3-6 and Table 3-6 is a detailed description. The TDSR is a read/write register which can be accessed in either word or byte mode with no restrictions. All bits can be read from or written into and are reset by Device Reset or Bus INIT except where noted.

RXCSR
16XXX0
READ/WRITE



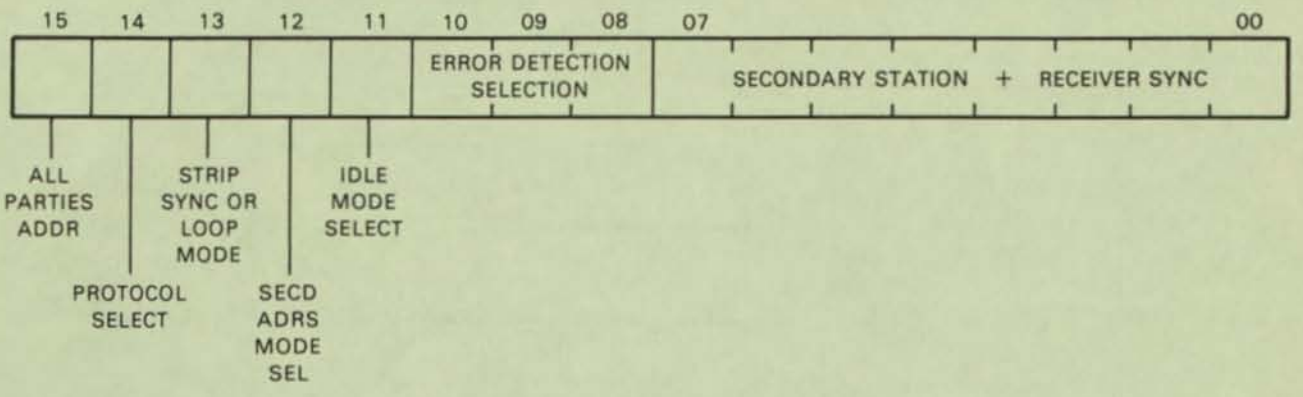
RDSR
16XXX2
READ ONLY

MK-1504



PCSAR
16XXX2
WRITE ONLY

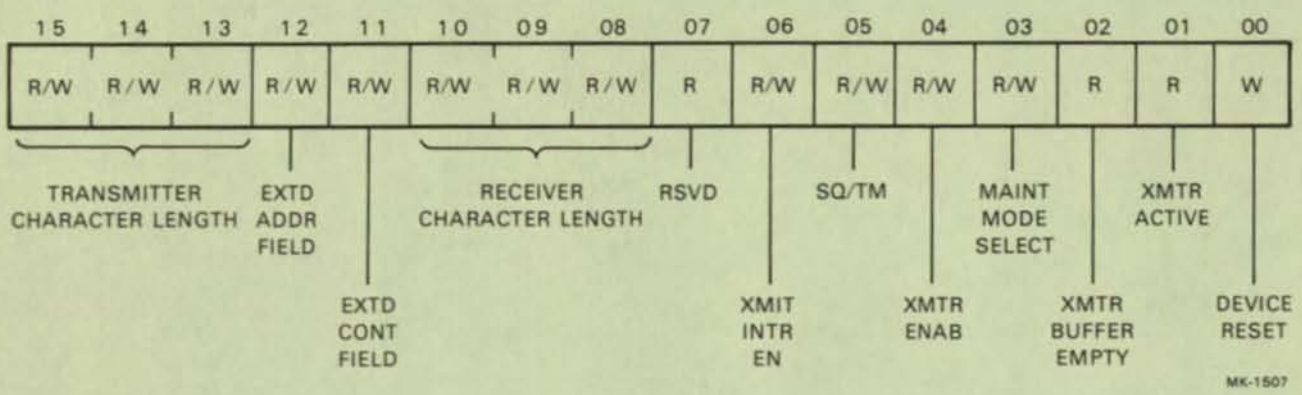
MK-1505



MK-1506

Figure 3-1 DPV11 Register Configurations and Bit Assignments (Sheet 1 of 2)

PCSAR
16XXX4
READ/WRITE



TDSR
16XXX6
READ/WRITE

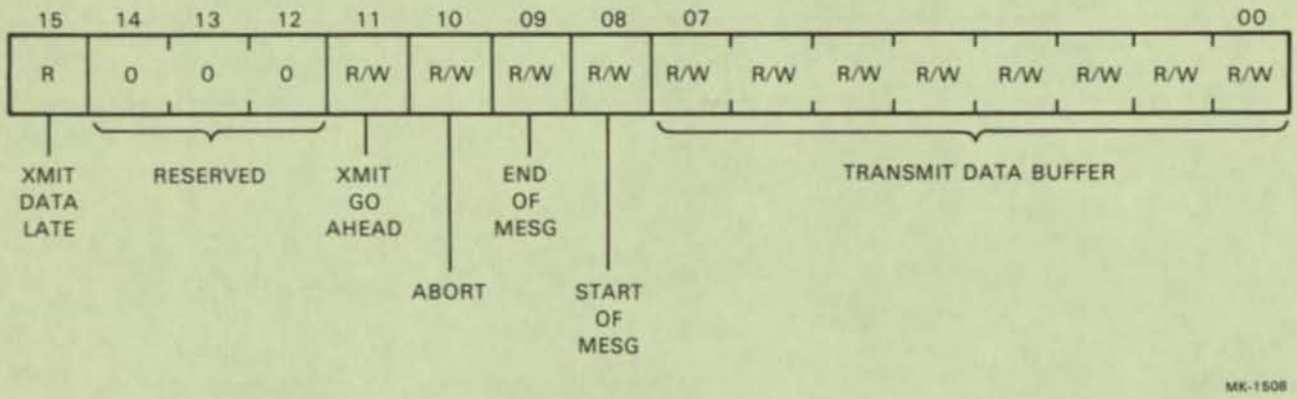


Figure 3-1 DPV11 Register Configurations and Bit Assignments (Sheet 2 of 2)

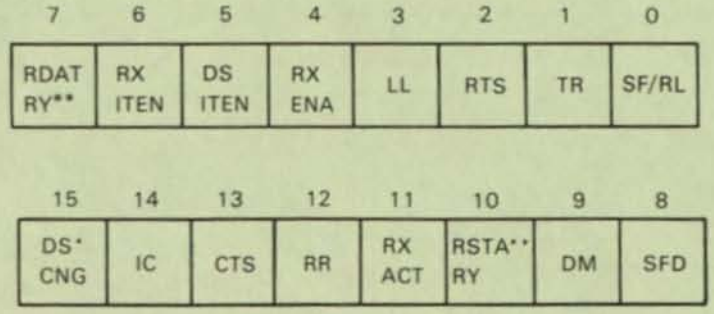


Figure 3-2 Receive Control and Status Register (RXCSR) Format

Table 3-2 Receive Control and Status Register (RXCSR) Bit Assignments

Bit	Name	Description
15	Data Set Change (DSCNG)	This bit is set when a transition occurs on any of the following modem control lines: Clear to Send Data Mode Receiver Ready Incoming Call Transition detectors for each of these four lines can be disabled by removing the associated jumper. Data Set Change is cleared by reading either byte of the RXCSR or by Device Reset or Bus INIT. Data Set Change causes a receive interrupt if DSITEN (bit 5) and RXITEN (bit 6) are both set.
14	Incoming Call (IC)	This bit reflects the state of the modem Incoming Call line. Any transition of this bit causes Data Set Change bit (bit 15) to be asserted unless the Incoming Call line is disabled by removing its jumper. This bit is read-only and cannot be cleared by software.
13	Clear to Send (CTS)	This bit reflects the state of the Clear to Send line of the modem. Any transition of this line causes Data Set Change (bit 15) to be set unless the jumper enabling the Clear to Send signal is removed. Clear to Send is a program read-only bit and cannot be cleared by software.
12	Receiver Ready (RR)	This bit is a direct reflection of modem Receiver Ready lead. It indicates that the modem is receiving a carrier signal. For external maintenance loopback, this signal must be high. If the line is open, RR is pulled high by the circuitry. Any transition of this bit causes Data Set Change (bit 15) to be asserted unless the jumper enabling the Receiver Ready signal is removed. Receiver Ready is a read-only bit and cannot be cleared by software.
11	Receiver Active (RXACT)	This bit is set when the USYNRT presents the first character of a message to the DPV11. It remains set until the receive data path of the USYNRT becomes idle. Receiver Active is cleared by any of the following conditions: a terminating control character is received in bit-oriented protocol mode; an off transition of Receiver Enable (RXENA) occurs; or Device Reset or Bus INIT is issued.

Table 3-2 Receive Control and Status Register (RXCSR) Bit Assignments (Cont)

Bit	Name	Description
		Receiver Active is a read-only bit which reflects the state of the USYNRT output pin 5.
10	Receiver Status Ready (RSTARY)	This bit indicates the availability of status information in the upper byte of the receive data and status register (RDSR). It is set when any of the following bits of the RDSR are set: Receiver End of Message (REOM); Receiver Overrun (RCV OVRUN); Receiver Abort or Go Ahead (RABORT); Error Check (ERRCHK) if VRC is selected. Receiver Status is cleared by any of the following conditions: reading either byte of the RDSR; clearing Receiver Enable (bit 4 of RXCSR); Device Reset, or Bus Init. When set, Receiver Status Ready causes a receive interrupt if Receive Interrupt Enable (bit 6) is also set. Receiver Status Ready is a read-only bit which reflects the state of USYNRT pin 7.
9	Data Mode (DM) (Data Set Ready)	This bit reflects the state of the Data Mode signal from the modem. When this bit is set it indicates that the modem is powered on and not in test, talk or dial mode. Any transition of this bit causes the Data Set Change bit (bit 15) to be asserted unless the Data Mode jumper has been removed. Data Mode is a read-only bit and cannot be cleared by software.
8	Sync or Flag Detect (SFD)	This bit is set for one clock time when a flag character is detected with bit-oriented protocols, or a sync character is detected with character-oriented protocols. SFD is a read-only bit which reflects the state of USYNRT pin 4.
7	Receive Data Ready (RDATRY)	This bit indicates that the USYNRT has assembled a data character and is ready to present it to the processor. If this bit becomes set while Receiver Interrupt Enable (bit 6) is set, a receive interrupt request will result. Receive Data Ready is reset when either byte of RDSR is read, Receiver Enable (bit 4) is cleared, or Device Reset or Bus INIT is issued. RDATRY is a read-only bit which reflectes the state of USYNRT pin 6.

Table 3-2 Receive Control and Status Register (RXCSR) Bit Assignments (Cont)

Bit	Name	Description
6	Receiver Interrupt Enable (RXITEN)	When set, this bit allows interrupt requests to be made to the receiver vector whenever RDATRY (bit 7) becomes set. The conditions which cause the interrupt request are the assertion of Receive Data Ready (bit 7), Receive Status Ready (bit 10), or Data Set Change (bit 15) if DSITEN (bit 5) is also set. RXITEN is a program read/write bit and is cleared by Device Reset or Bus INIT.
5	Data Set Interrupt Enable (DSITEN)	This bit, when set along with RXITEN, allows interrupt requests to be made to the receiver vector whenever Data Set Change (bit 15) becomes set. DSITEN is a program read/write bit and is cleared by Device Reset or Bus INIT.
4	Receiver Enable (RXENA)	This bit controls the operation of the receive section of the USYNRT. When this bit is set, the receive section of the USYNRT is enabled. When it is reset the receive section is disabled. In addition to disabling the receive section of the USYNRT, resetting bit 4 reinitializes all but two of the USYNRT receive registers. The two registers not reinitialized are the character length selection buffer and the parameter control register.
3	Local Loopback (LL)	Asserting this bit causes the modem connected to the DPV11 to establish a data loopback test condition. Clearing this bit restores normal modem operation. Local Loopback is program read/write and is cleared by Device Reset or Bus request to Send is program read/write and is cleared by Device Reset or Bus INIT.
2	Request to Send (RTS)	Setting this bit asserts the Request to Send signal at the modem interface. Request to Send is program read/write and is cleared by Device Reset or Bus INIT.
1	Terminal Ready (TR) (Data Terminal Ready)	When set, this bit asserts the Terminal Ready signal to the modem interface. For auto dial and manual call origination, it maintains the established call. For auto answer, it allows handshaking in response to a Ring signal.

Table 3-2 Receive Control and Status Register (RXCSR) Bit Assignments (Cont)

Bit	Name	Description
0	Select Frequency or Remote Loopback (SF/RL)	This bit can be wire-wrap jumpered to function as either select frequency or remote loopback. When jumpered as select frequency (W3 to W4), setting this bit selects the modem's higher frequency band for transmission to the line and the lower frequency band for reception from the line. The clear condition selects the lower frequency for transmission and the higher frequency for reception. When jumpered for remote loopback (W5 to W3), this bit, when asserted, causes the modem connected to the DPV11 to signal when a remote loopback test condition has been established in the remote modem. SF/RL is program read/write and is cleared by Device Reset or Bus INIT.

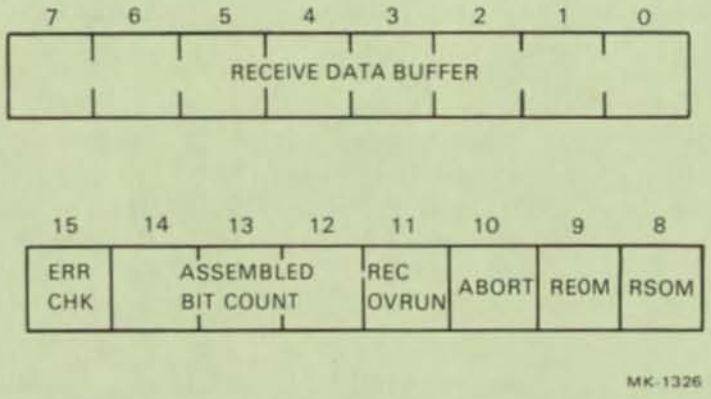


Figure 3-3 Receive Data and Status Register (RDSR) Format

Table 3-3 Receive Data and Status Register (RDSR) Bit Assignments

Bit	Name	Description
15	Error Check (ERR CHK)	This bit when set, indicates a possible error. It is used in conjunction with the error detection selection bits of the parameter control sync/address register (bits 8–10) to indicate either an error or an all zeros state of the CRC register. With bit-oriented protocols, ERR CHK indicates that a CRC error has occurred. It is set when the Receive End of Message bit (RDSR bit 9) is set. With character-oriented protocols ERR CHK is asserted with each data character if all zeros are in the CRC register. The processor must then determine if this indicates an error-free

Table 3-3 Receive Data and Status Register (RDSR) Bit Assignments (Cont)

Bit	Name	Description
		message or not. If VRC parity is selected, this bit is set for every character which has a parity error. ERR CHK is cleared by reading the RDSR, clearing RXENA (RXCSR bit 4), Device Reset or Bus INIT.
14–12	Assembled Bit Count (ABC)	Used only with bit-oriented protocols, these bits represent the number of valid bits in the last character of a message. They are all zeros unless the message ends on an unstated boundary. The bits are encoded to represent valid bits as shown below. **14 13 12 Number of Valid Bits** 0 0 0 All bits are valid 0 0 1 One valid bit 0 1 0 Two valid bits 0 1 1 Three valid bits 1 0 0 Four valid bits 1 0 1 Five valid bits 1 1 0 Six valid bits 1 1 1 Seven valid bits These bits are presented simultaneously with the last bits of data and are cleared by reading the RDSR or by resetting RXENA (bit 4 of RXCSR).
11	Receiver Overrun (RCV OVRUN)	This bit is used to indicate that an overrun situation has occurred. Overrun exists when the data buffer (bits 0–7 of RDSR) has not been serviced within one character time. As a general rule, the overrun is indicated when the last bit of the current character has been received into the shift register of the USYNRT and the data buffer is not yet available for a new character. Two factors exist which modify this general rule and apply only to bit-oriented protocols. The first factor is the number of bits inserted into the data stream for transparency. For each bit inserted during the formatting of the current character, the controller's maximum response time is increased by one clock cycle. The second factor is the result of termination of the current message. When this occurs, the data of the terminated message which is within the USYNRT is not overrunable. If an attempt is made to displace this data by the reception of a subsequent message, the data of the subsequent message is lost until the data of the prior message has been released.

Table 3-3 Receive Data and Status Register (RDSR) Bit Assignments (Cont)

Bit	Name	Description
10	Receiver Abort or Go Ahead (RABORT)	This bit is used only with bit-oriented protocols and indicates that either an abort character or a go-ahead character has been received. This is determined by the Loop Mode bit (PCSAR bit 13). If the Loop Mode bit is clear, RABORT indicates reception of an abort character. If the Loop Mode bit is set, RABORT indicates a go-ahead character has been received. The setting of RABORT causes Receiver Status Ready (bit 10 of RXCSR) to be set. RABORT is reset when the RDSR is read or when Receiver Enable (bit 4 of RXCSR) is reset. The abort character is defined to be seven or more contiguous one bits appearing in the data stream. Reception of this bit pattern when Loop Mode is clear causes the receive section of the USYNRT to stop receiving and set RSTARY (bit 10 of RXCSR). The abort character indicates abnormal termination of the current message. The go-ahead character is defined as a zero bit followed by seven consecutive one bits. This character is recognized as a normal terminating control character when the Loop Mode bit is set. If Loop Mode is cleared this character is interpreted as an abort character.
9	Receiver End of Message (REOM)	This bit is used only with bit-oriented protocols and is asserted if Receiver Active (bit 11 of RXCSR) is set and a message is terminated either normally or abnormally. When REOM becomes set, it sets RSTARY (bit 10 of RXCSR). REOM is cleared when RDSR is read or when Receive Enable (bit 4 of RXCSR) is reset.
8	Receiver Start of Message (RSOM)	Used only with bit-oriented protocols. This bit is presented to the processor along with the first data character of a message and is synchronized to the last received flag character. Setting of RSOM does not set RSTARY (RXCSR bit 10). RSOM is cleared by Device Reset, Bus INIT, resetting Receiver Enable (RXCSR bit 4), or the next transfer into the Receive Data buffer (low byte of RDSR).
7–0	Receive Data Buffer	The low byte of the RDSR is the Receive Data buffer. The serial data input to the USYNRT is assembled and transferred to the low byte of the RDSR for presentation to the processor. When the RDSR receives data, Receive Data Ready (bit 7 of RXCSR) becomes set to indicate that the RDSR has data to be picked up. If this data is not read within one character time, a data overrun occurs. The characters in the Receive Data buffer are right-justified with bit 0 being the least significant bit.

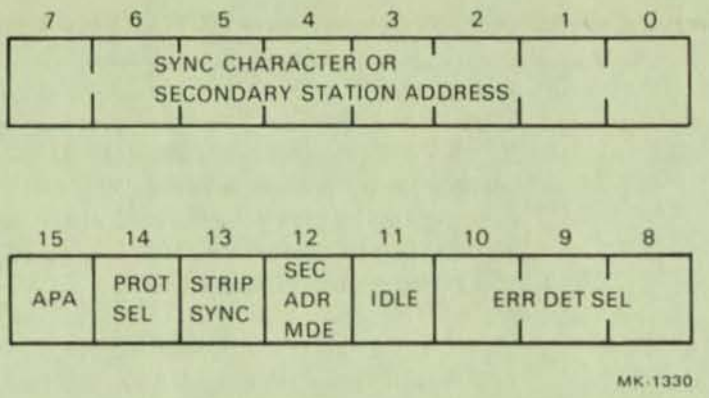


Figure 3-4 Parameter Control Sync/Address Register (PCSAR) Format

Table 3-4 Parameter Control Sync/Address Register (PCSAR) Bit Assignments

Bit	Name	Description
15	All Parties Addressed (APA)	This bit is set when automatic recognition of the All Parties Addressed character is desired. The All Parties Addressed character is eight bits of ones with necessary bit stuffing so as not to be confused with the abort character. Recognition of this character is done in the same way as the secondary station address (see bit 12 of this register) except that the broadcast address is essentially hardwired within the receive data path. The logic inspects the address character of each frame for the broadcast address. When the broadcast address is recognized, the USYNRT makes it available and sets Receiver Start of Message (bit 8 of RDSR). If the broadcast address is not recognized, one of two possible actions occurs. 1. If the Secondary Address Select mode bit (bit 12) is set, a test of the secondary station address is made. 2. If bit 12 is not set or the secondary station address is not recognized, the receive section of the USYNRT renews its search for synchronizing control characters.
14	Protocol Select (PROT SEL)	This bit is used to select between character- and byte count-oriented or bit-oriented protocols. It is set for character- and byte count-oriented protocols and reset for bit-oriented protocols.
13	Strip Sync or Loop Mode (STRIP SYNC)	This bit serves the following two functions. 1. Strip Sync (character-oriented protocols) – In character-oriented protocols, all sync characters after the initial synchronization are deleted from the message and not included in the CRC computation if this bit is set. If it is cleared, all sync characters remain in the message and are included in the CRC computation.

Table 3-4 Parameter Control Sync/Address Register (PCSAR) Bit Assignments (Cont)

Bit	Name	Description
		2. Loop Mode (bit-oriented protocols) – With bit-oriented protocols, this bit is used to control the method of termination. If it is set, either a flag or go-ahead character can cause a normal termination of a message. If it is cleared, only a flag character can cause a normal termination.
12	Secondary Address Mode (SEC ADR MDE)	This bit is used with bit-oriented protocols when automatic recognition of the secondary station address is desired. If it is set, the station address of the incoming message is compared with the address stored in the low byte of this register. Only messages prefixed with the correct secondary address are presented to the processor. If the addresses do not compare, the receive section of the USYNRT goes back to searching for flag or go-ahead characters. When SEC ADR MDE is cleared, the receive section of the USYNRT recognizes all incoming messages.
11	Idle Mode Select (IDLE)	This bit is used with both bit- and character-oriented protocols. With bit-oriented protocols, IDLE is used to select the type of control character issued when either Transmit Abort (bit 10 of TDSR) is set or a data underrun error occurs. If IDLE is set, flag characters are issued. If IDLE is clear, abort characters are issued. With character-oriented protocols, IDLE is used to control the method in which initial sync characters are transmitted and the action of the transmit section of the USYNRT when an underrun error occurs. IDLE is cleared to cause sync characters from the low byte of PCSAR to be transmitted. When IDLE is set, the transmit data output is held asserted during an underrun error and at the end of a message.
10–8	Error Detection Selection (ERR DEL SEL)	These bits are used to determine the type of error detection used on received and transmitted messages. In bit-oriented protocols, the selection is independent of character length. In character- and byte count-oriented protocols, CRC error detection is usable only with 8-bit character lengths. The maximum character length for VRC is seven. The bits are encoded as follows. **10 9 8 CRC Polynomial** 0 0 0 $x^{16}+x^{12}+x^5+1$ (CRC CCITT) (Both CRC data registers in the transmit and receive sections are set to all ones prior to the computation.) 0 0 1 $x^{16}+x^{12}+x^5+1$ (CRC CCITT) (Both CRC data registers set to all zeros.)

Table 3-4 Parameter Control Sync/Address Register (PCSAR) Bit Assignments (Cont)

Bit	Name	Description			
		0	1	0	Not used
		0	1	1	$x^{16}+x^{15}+x^2+1$ (CRC 16) (Both CRC registers set to all zeros.)
		1	0	0	Odd VRC Parity (A parity bit is attached to each transmitted character.) Should be used only in character-oriented protocols.
		1	0	1	Even VRC parity (Resembles odd VRC except that an even number of bits are generated.)
		1	1	0	Not used.
		1	1	1	All error detection is inhibited.
7–0	Sync Character or Secondary Address	The low byte of PCSAR is used as either the sync character for character-oriented protocols or as the secondary station address for bit-oriented protocols. The bits are right-justified with the least significant bit being bit 0.			

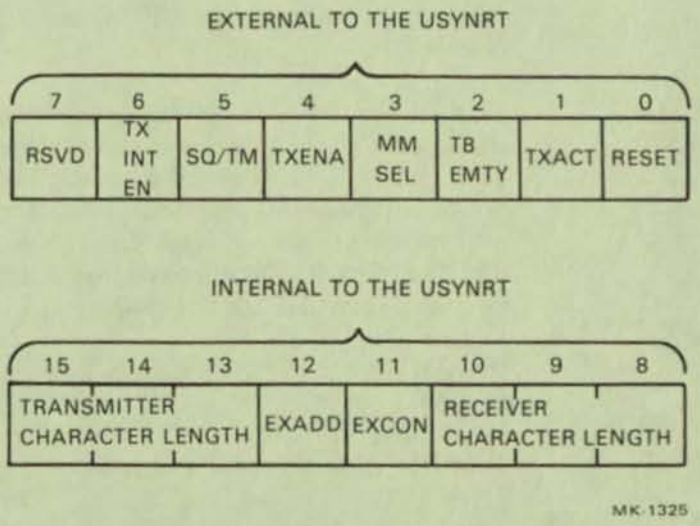


Figure 3-5 Parameter Control and Character Length Register (PCSCR) Format

Table 3-5 Parameter Control and Character Length Register (PCSCR) Bit Assignments

Bit	Name	Description
15–13	Transmitter Character Length	These bits can be read or written and are used to determine the length of the characters to be transmitted. They are encoded to set up character lengths as follows. **15 14 13 Character Length** 0 0 0 Eight bits per character 1 1 1 Seven bits per character 1 1 0 Six bits per character 1 0 1 Five bits per character (bit-oriented protocol only) 1 0 0 Four bits per character (bit-oriented protocol only) 0 1 1 Three bits per character (bit-oriented protocol only) 0 1 0 Two bits per character (bit-oriented protocol only) 0 0 1 One bit per character (bit-oriented protocol only) These bits can be changed while the transmitter is active, in which case the new character length is assumed at the completion of the current character. This field is set to a character length of eight by Device Reset or Bus INIT. When VRC error detection is selected, the default character length is eight bits plus parity.
12	Extended Address Field (EXADD)	This bit is used with bit-oriented protocols and affects the address portion of a message in receiver operations. When it is set, each address byte is tested for a one in the least significant bit position. If the least significant bit is zero, the next character is an extension of the address field. If the least significant bit is one, the current character terminates the address field and the next character is a control character. EXADD is not used with Secondary Address Mode (bit 12 of PCSAR). EXADD is read/write and is reset by Device Reset or Bus INIT.
11	Extended Control Field (EXCON)	This bit is used with bit-oriented protocols and affects the control character of a message in receiver operations. When EX-

Table 3-5 Parameter Control and Character Length Register (PCSCR) Bit Assignments (Cont)

Bit	Name	Description
		CON is set it extends the control field from one 8-bit byte to two 8-bit bytes. EXCON is not used with Secondary Address Mode (bit 12 of PCSAR) EXCON is read/write and is reset by Device Reset or Bus INIT.
10–8	Receiver Character Length	These bits are used to determine the length of the characters to be received. They are encoded to set up character lengths as follows. **10 9 8 — Character Length** 0 0 0 — Eight bits per character 1 1 1 — Seven bits per character 1 1 0 — Six bits per character 1 0 1 — Five bits per character 1 0 0 — Four bits per character (bit-oriented protocols only) 0 1 1 — Three bits per character (bit-oriented protocols only) 0 1 0 — Two bits per character (bit-oriented protocols only) 0 0 1 — One bit per character (bit-oriented protocols only)
7	Reserved	Not used by the DPV11
6	Transmit Interrupt Enable (TXINTEN)	When set, this bit allows a transmitter interrupt request to be made to the transmitter vector when Transmit Buffer Empty (TBEMTY) is asserted. Transmit Interrupt Enable (TXINTEN) is read/write and is cleared by Device Reset or Bus INIT.
5	Signal Quality or Test Mode (SQ/TM)	This bit can be wire-wrap jumpered to function as either Signal Quality or Test Mode. When jumpered for signal quality (W5 to W6), this bit reflects the state of the signal quality line from the modem. When asserted, it indicates that there is a low probability of errors in the received data. When clear it indicates that there is a high probability of errors in the received data.

Table 3-5 Parameter Control and Character Length Register (PCSCR) Bit Assignments (Cont)

Bit	Name	Description
		When jumpered for the test mode (W6 to W7), this bit indicates that the modem has been placed in a test condition when asserted. The modem test condition could be established by asserting Local Loopback (bit 3 of RXCSR), Remote Loopback (bit 0 of RXCSR) or other means external to the DPV11. When SQ/TM is clear, it indicates that the modem is not in test mode and is available for normal operation. SQ/TM is program read-only and cannot be cleared by software.
4	Transmitter Enable (TXENA)	This bit must be set to initiate the transmission of data or control information. When this bit is cleared, the transmitter will revert back to the mark state once all indicated sequences have been completed. TXENA should be cleared after the last data character has been loaded into the transmit data and status register (TDSR). Transmit End of Message (bit 9 of TDSR) should be asserted when TXENA is reset (if it is to be asserted at all) and remain asserted until the transmitter enters the idle mode. TXENA is connected directly to USYNRT pin 37. It is a read/write bit and is reset by Device Reset or Bus INIT.
3	Maintenance Mode Select (MM SEL)	When this bit is asserted, it causes the USYNRT's serial output to be internally connected to the USYNRT's serial input. The serial send data output line from the interface is asserted and the receive data serial input is disabled. Send timing and receive timing to the USYNRT are disabled and replaced with a clock signal generated on the interface. The clock rate is either 49.152K b/s or 1.9661K b/s depending on the position of a jumper on the interface board. Maintenance mode allows diagnostics to run in loopback without disconnecting the modem cable. MM SEL is a read/write bit and is cleared by Device Reset or Bus INIT. When it is cleared, the interface is set for normal operation.
2	Transmitter Buffer Empty (TBEMTY)	This bit is asserted when the transmit data and status register (TDSR) is available for new data or control information. It is also set after a Device Reset or Bus INIT. The TDSR should be loaded only in response to TBEMTY being set. When the TDSR is written into, TBEMTY is cleared. If TBEMTY becomes set while Transmit Interrupt Enable (bit 6 of PCSCR) is set, a transmit interrupt request results. TBEMTY reflects the state of USYNRT pin 35.

Table 3-5 Parameter Control and Character Length Register (PCSCR) Bit Assignments (Cont)

Bit	Name	Description
1	Transmitter Active (TXACT)	This bit indicates the state of the transmit section of the USYNRT. It becomes set when the first character of data or control information is transmitted. TXACT is cleared when the transmitter has nothing to send or when Device Reset or Bus INIT is issued. TXACT reflects the state of USYNRT pin 34.
0	Device Reset (RESET)	When a one is written to this bit all components of the interface are initialized. It performs the same function as Bus INIT with respect to this interface. Modem Status (Data Mode, Clear to Send, Receiver Ready, Incoming Call, Signal Quality or Test Mode) is not affected. RESET is write-only; it cannot be read by software.

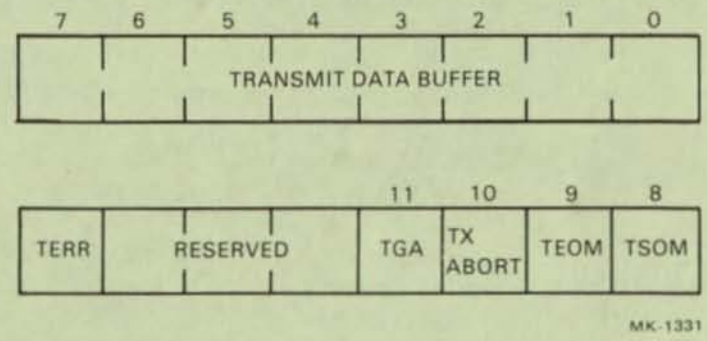


Figure 3-6 Transmit Data and Status Register (TDSR) Format

Table 3-6 Transmit Data and Status Register (TDSR) Bit Assignments

Bit	Name	Description
15	Transmitter Error (TERR)	This is a read-only bit which becomes asserted when the Transmitter Buffer Empty (TBEMTY) indication has not been serviced for more than one character time. When TERR occurs in bit-oriented protocols, the transmit section of the USYNRT generates an abort or flag character based on the state of the IDLE bit (PCSAR bit 11). If IDLE is set, a flag character is sent. If it is reset, an abort character is sent. When TERR occurs in character-oriented protocols, the state of the IDLE bit again determines the result. If IDLE is set, the transmit serial output is held in the MARK condition. If it is cleared, a sync character is transmitted.

Table 3-6 Transmit Data and Status Register (TDSR) Bit Assignments (Cont)

Bit	Name	Description
		TERR is cleared when TSOM (TDSR bit 8) becomes set or by Device Reset or Bus INIT. Clearing Transmitter Enable (PCSCR bit 4) does not clear TERR and TERR is not set with Transmit End of Message.
14–12	Reserved	Not used by the DPV11
11	Transmit Go Ahead (TGA)	This bit, when asserted, modifies the bit pattern of the control character initiated by either Transmit Start of Message (TSOM) or Transmit End of Message (TEOM). TSOM or TEOM normally causes a flag character to be sent. If TGA is set, a go-ahead character is sent in place of the flag character. TGA is only used with bit-oriented protocols.
10	Transmit Abort (TXABORT)	This bit is used only with bit-oriented protocols to abnormally terminate a message or to transmit filler information used to establish data link timing. When TXABORT is asserted, the transmitter automatically transmits either flag or abort characters depending on the state of the IDLE mode bit. If IDLE is cleared, abort characters are sent. If IDLE is set, flag characters are sent.
9	Transmit End of Message (TEOM)	This control bit is used to normally terminate a message in bit-oriented protocol. It also terminates a message in character-oriented protocols when CRC error detection is used. As a secondary function, it is used in conjunction with the Transmit Start of Message (TSOM) bit to transmit a SPACE SEQUENCE. Refer to the TSOM bit description (bit 8 of this register) for information regarding this sequence. With bit-oriented protocols, asserting this bit causes the CRC information to be transmitted, if CRC is enabled, followed by flag or go-ahead characters depending on the state of the Transmit Go Ahead (TGA) bit. See bit 11 of this register. With character-oriented protocols, asserting this bit causes CRC information, if CRC is enabled, to be transmitted followed by either sync characters or a MARK condition depending on the state of the IDLE bit. If IDLE is cleared, sync characters are transmitted. The character following the CRC information is repeated until the transmitter is disabled or the TEOM bit is cleared. A subsequent message may be initiated while the transmit section of the USYNRT is active. This is accomplished by clearing the TEOM bit and supplying new message data without setting

Table 3-6 Transmit Data and Status Register (TDSR) Bit Assignments (Cont)

Bit	Name	Description
		the Transmit Start Of Message bit. However, the CRC character for the prior message must have completed transmission.
8	Transmit Start of Message (TSOM)	This bit is used with either bit- or character-oriented protocols. As long as it remains asserted, flag characters (bit-oriented protocols) or sync characters (character-oriented protocols) are transmitted. With bit-oriented protocols, a space sequence (byte mode only) of 16 zero bits can be transmitted by asserting TSOM and TEOM simultaneously provided the transmitter is in the idle state and Transmit Enable is cleared. This should not be done during the transfer of data, and must only be done in byte mode. **NOTE** **When using the special space sequence function, all registers internal to the USYNRT must be written in byte mode.** Normally at the completion of each sync, flag, go-ahead or Abort character, the TBEMTY indication is asserted. This allows the software to count the number of transmitted characters. In certain applications, the software may elect to ignore the service of the Transmitter Buffer Empty (TBEMTY) indication. Normally during data transfers, this would cause a transmit data late error. The TSOM bit asserted suppresses this error and provides the necessary synchronization to automatically transmit another flag, go-ahead or sync character.
7–0	Transmit Data Buffer	Data from the processor to be transmitted on the serial output line is loaded into this byte of the TDSR when Transmitter Buffer Empty (TBEMTY) is asserted. If the transmitter buffer is not loaded within one character time, an underrun error occurs. The characters are right-justified, with bit 0 being the least significant bit.

3.4 DATA TRANSFERS

Paragraphs 3.4.1 and 3.4.2 discuss receive and transmit data transfers as they relate to the system software.

3.4.1 Receive Data

Serial data to be presented to the DPV11 from the modem enters the receiver circuit and is presented to the USYNRT. Recognition by the USYNRT of a control character initiates the transfer. When a transfer has been initiated, a character is assembled by the USYNRT and then placed in the low byte of the receive data and status register (RDSR) when it is available. If the RDSR is not available, the transfer is delayed until the previous character has been serviced. This must take place before the next character is fully assembled or an overrun error exists. Refer to the description of bit 11 in Table 3-3 for more details on Receiver Overrun.

Servicing of the RDSR is the responsibility of the system software in response to the Receive Data Ready (RDATRY) signal. This signal is asserted when a character has been transferred to the RDSR. The setting of RDATRY would also cause a receive interrupt request if Receive Interrupt Enable (RXITEN) is set. The software's response to RDATRY is to read the contents of the RDSR. At the completion of this operation, the new information is loaded into the RDSR and RDATRY is reasserted. This operation continues until terminated by some control character. The upper byte of the RDSR contains status and error indications which the software can also read.

The DPV11 will handle data in bit-, byte count- or character-oriented protocols.

With bit-oriented protocol, only flag characters are used to initiate the transfer of a message. Information inserted into the data stream for transparency or control is deleted before it is presented to the RDSR. This means that only data characters are available to the software. The first two characters of every message or frame are defined to be 8-bit characters and the USYNRT will handle them as such regardless of the programmed character length. All subsequent data is formatted in the selected character length. When CRC error detection is selected, the received CRC check characters are not presented to the software, but the error indication will be presented if an error has been detected.

If the secondary address mode is implemented, the first received data character must be the selected address. If this is not the case, the USYNRT will renew its search for flag or go-ahead characters. Refer to the description of bit 12 of the PCSAR in Table 3-4.

With byte count- or character-oriented protocols, two consecutive sync characters are required to synchronize the transfer of data. The sync characters used in the message must be the same as the sync character loaded by the software into the low byte of the parameter control sync/address register (PCSAR). If leading sync characters subsequent to the initial two syncs are to be deleted from the data stream, the Strip Sync bit (bit 13) must also be set in the upper byte of the PCSAR. The character length of the data to be received should also be set in bits 8, 9, and 10 of the parameter control and character length register (PCSCR). Sync characters and data must have the same character length and only characters of the selected length will be presented to the receive buffer. Sync characters following the initial two will be presented to the buffer and included in the CRC computation unless the Strip Sync bit is set. If vertical redundancy check (VRC) parity checking is selected, the parity bit itself is deleted from the character before it is presented to the buffer.

3.4.2 Transmit Data

System software loads information to be transmitted to the modem into the transmit data and status register (TDSR). This does not ordinarily include error detection or control character information. Loading of the TDSR occurs in response to the Transmitter Buffer Empty (TBEMTY) signal from the USYNRT. The character length of information to be transmitted is established by the software when it loads the transmit character length register (bits 13, 14, and 15 of the PCSCR). The default length of eight is assigned when the transmit character length register equals zero. The length of characters presented to the TDSR should not exceed the assigned character length. When the information in the TDSR is transmitted, the TBEMTY signal is again asserted to request another character. The setting of TBEMTY also causes a transmit interrupt request if Transmit Interrupt Enable is set.

Byte count- or character-oriented protocols require the transmission of synchronizing information normally referred to as sync characters. The sync characters can be transmitted when Transmit Start of Message (TDSR bit 8) is set. This happens in one of two ways depending on the state of the IDLE bit (PCSAR bit 11). When the IDLE bit is cleared, the sync character is taken directly from the common sync register (PCSAR bits 7–0). The sync register would have been previously loaded by the software. If the IDLE bit is set, the sync character must be loaded into the TDSR by the software when it is to be transmitted. If multiple sync characters are to be transmitted, the TDSR must only be loaded with the first one of the sequence. This character will be transmitted until data information is loaded into the TDSR. The TBEMTY signal is asserted at the end of each sync character but the TSOM signal allows it to be ignored without causing a data late error.

With bit-oriented protocols, the USYNRT automatically generates control characters as initiated by the software and inserts necessary information into the data stream to maintain transparency.

Typical programming examples in bit- and byte count-oriented protocols appear in Appendix D.

3.5 INTERRUPT VECTORS

The DPV11 generates two vector addresses, one for receive data and modem control and the other for transmit data.

The receive and modem control interrupt has priority over the transmit interrupt and is enabled by setting bit 6 (RXITEN) of the receiver control and status register (RXCSR).

If bit 6 of the RXCSR is set, a receiver interrupt may occur when any one of the following signals is asserted.

- Receive Data Ready (RDATRY)
- Receive Status Ready (RSTARY)
- Data Set Change (DAT SET CH)

The signal DAT SET CH only causes an interrupt if bit 5 (DSITEN) of the RXCSR is also set.

It is possible that a data set change interrupt could be pending while a receiver interrupt is being serviced, or the opposite could be true. In either case, the hardware ensures that both interrupt requests are recognized.

NOTE

The modem status change circuit interprets any pulse of two microseconds or greater duration as a data set change. This ensures that all legitimate transitions of modem status will be detected. However, on a poor line, noise may be interpreted as a data set change. Software written for the DPV11 must account for this possibility.

A transmitter interrupt request occurs if Transmit Interrupt Enable (TXINTEN) is set when Transmit Buffer Empty (TBEMTY) becomes asserted.

CHAPTER 4
TECHNICAL DESCRIPTION

4.1 INTRODUCTION

This chapter provides a 2-level discussion of the DPV11. Paragraph 4.2 includes a description of the DPV11 logic in functional groups at the block diagram level. At this level, a general operational overview is also discussed. The second level of discussion is the detailed description, which covers the complete DPV11 logic at the circuit schematic level, as shown in the DPV11 print set.

4.2 FUNCTIONAL DESCRIPTION

4.2.1 Logic Description

For discussion purposes, the DPV11 logic is divided into the ten sections shown in Figure 4-1. The sections are described in Paragraphs 4.2.1.1 through 4.2.1.10.

4.2.1.1 Bus Transceivers – The interface for data, and address on the LSI-11 bus consists of four bus transceiver chips (DC005). These function as bidirectional buffers between the LSI-11 bus and the DPV11 Logic. These transceivers provide isolation, address comparison, and vector generation.

4.2.1.2 Read/Write Control – The read/write control logic consists of a DC004 protocol chip and its associated logic. It provides the control signals for accessing registers and strobing data. It controls reading from and writing into registers in both word and byte mode, and provides the deskew delays for these operations. When data has been placed on or picked up from the LSI-11 bus or when vector information has been placed on the LSI-11 bus, the read/write control logic notifies the processor by asserting BRPLY.

4.2.1.3 USYNRT and Bidirectional Buffer – The USYNRT provides a large portion of the functionality of the DPV11. The USYNRT is installed in a socket for ease of replacement. It provides complete serialization, deserialization and buffering of data between the modem and the LSI-11 bus. The USYNRT also provides logic support, via program parameter registers, for basic protocol handling and error detection.

The tri-state bidirectional buffer provides the fan-out drive to accommodate the number of circuits the USYNRT feeds.

4.2.1.4 Receive Control And Status Register (RXCSR) – This register contains most of the control and status information pertaining to receiver operation, including the status of the lines to and from the data set. The receive and data set interrupt enable bits are also contained in this register, but the receive interrupt enable is actually generated by the interrupt logic. The high byte of the RXCSR is read-only and the low byte is read/write. RXCSR is both word- and byte-addressable.

4.2.1.5 Transmit Control And Status Register – This register is the low byte of the parameter control and character length register (PCSCR). (The high byte is internal to the USYNRT). It contains most of the control and status information pertaining to transmit operations. The maintenance mode bit is also a part of this register. The register is read/write and can be accessed separately as the low byte of the PCSCR or in word mode when the entire PCSCR is accessed.

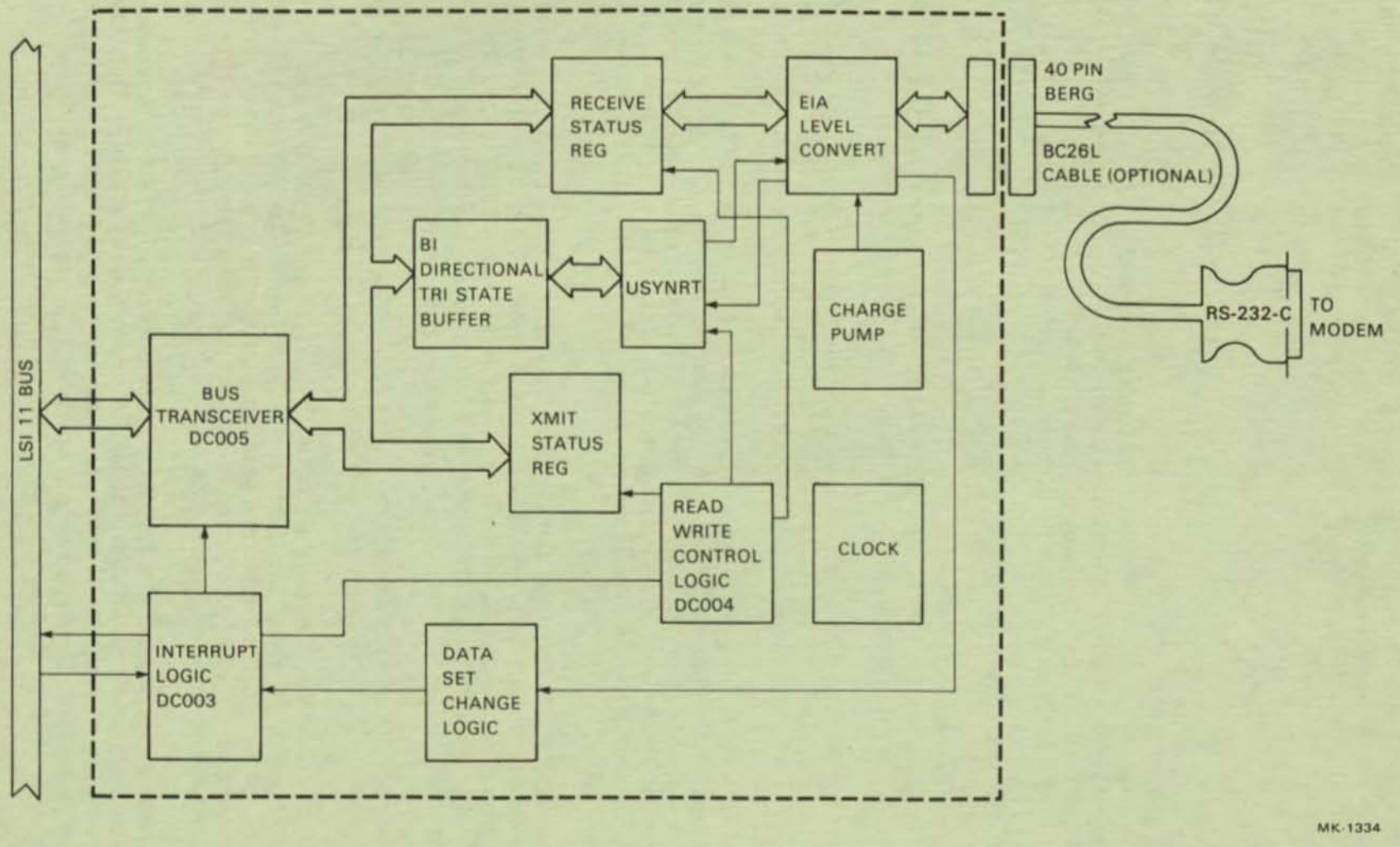


Figure 4-1 DPV11 Block Diagram

4.2.1.6 Interrupt Logic – Most of the logic for interrupts is contained in a single DC003 interrupt chip. The chip contains two interrupt channels: one for receive and one for transmit interrupts. The circuit generates a receive interrupt when the Receiver Interrupt Enable bit (RXITEN) is set and one of the following signals becomes asserted.

Receive Status Ready (RSTARY)
Receive Data Ready (RDATRY)
Modem Control Interrupt Request (MCINT)

MCINT requires that DSITEN (RXCSR bit 5) also be set.

If the Transmit Interrupt Enable bit (PCSCR bit 6) is set, a transmit interrupt is generated when the Transmit Buffer Empty signal (TBEMTY) is asserted.

Receive interrupts have priority over transmit interrupts.

4.2.1.7 Data Set Change Logic – This logic is used to determine if the modem had a change in status. Jumpers can be removed or installed to allow any or all of the following signals to set the Data Set Change bit (RXCSR bit 15).

RS-232-C	RS-449
Clear to Send (CTS)	Clear to Send (CTS)
Carrier Detect (CD)	Receiver Ready (RR)
Data Set Ready (DSR)	Data Mode (DM)
Ring Indicator (RI)	Incoming Call (IC)

If the Data Set Interrupt Enable bit and Receiver Interrupt Enable (RXCSR bits 5 and 6) are both set, Data Set Change causes the interrupt logic to generate an interrupt request.

4.2.1.8 Clock Circuit – The clock circuit consists of a 19.6608 MHz off-the-shelf oscillator and two 74LS390 dividers to provide the clock signals for the DPV11.

4.2.1.9 EIA Level Converters – These circuits contain drivers and receivers necessary for converting from TTL levels to EIA levels and from EIA levels to TTL levels. There are drivers and receivers to accommodate both RS-422-A (RS-449 compatible: limited to clock and data) and RS-423-A (RS-232-C compatible) electrical standards. Selection of RS-422-A or RS-423-A interface standard is provided by wire-wrap jumpers.

4.2.1.10 Charge Pump – This circuit converts the +12 volts to a negative voltage to power the RS-423-A drivers.

4.2.2 General Operational Overview

This discussion describes the relationships between the different sections of the block diagram from a simplified operational viewpoint. It is assumed for the purpose of this discussion that the DPV11 will be operated with the interrupts enabled. A simplified diagram which emphasizes the functions of the USYNRT (Figure 4-2) is referenced for both the receive and transmit operations. Bit-oriented protocol (BOP) and byte count-or character-oriented protocols (BCP) are not discussed in detail here.

4.2.2.1 Receive Operation – Serial data from the modem enters the EIA receiver where it is converted from EIA to TTL level. This TTL data is then presented directly to the receive serial input of

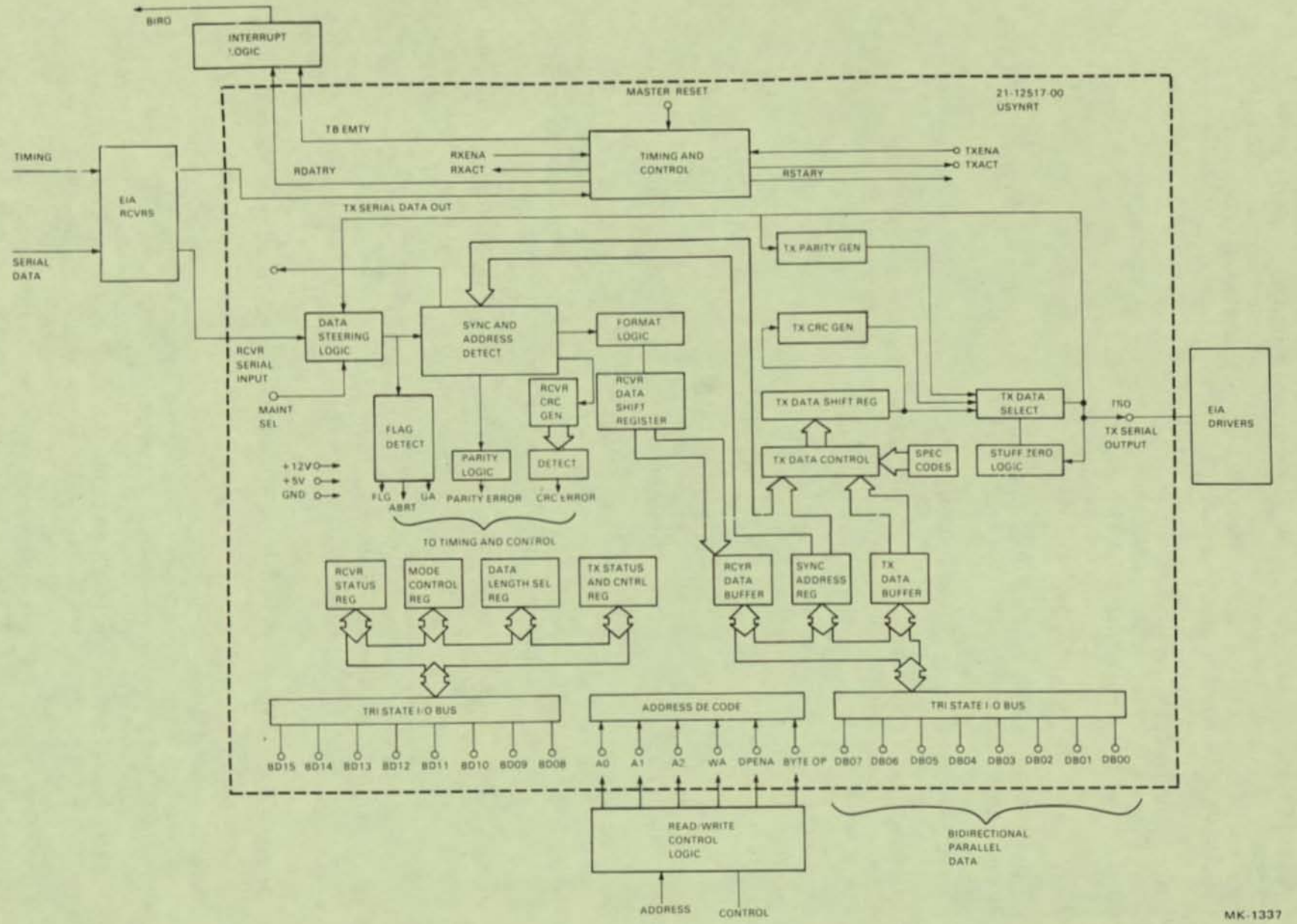


Figure 4-2 Simplified Functional Diagram

the USYNRT. At the same time the EIA receiver converts the receive timing signal from the modem to TTL level and presents it to the USYNRT. The USYNRT uses the timing signal to control the assembling of the incoming data characters. As the information enters the USYNRT, sync-detect and flag-detect circuits check for FLAG (BOP) or SYNC (BCP) until there is a match. When a match occurs, assembling of data characters begins. Error circuits check for errors while the data is being assembled. When a character is assembled in the receive data shift register, it is then transferred to the receive data buffer, and the USYNRT timing and control logic generates the signal receive data ready (RDATRY). Interrupt logic uses this signal to produce an interrupt request to the processor. When the processor responds to the interrupt request, the interrupt logic causes the bus transceiver circuits to assert the associated vector and the interrupt sequence takes place.

The processor now retrieves the data from the receive data buffer which resets the interrupt condition. To do this the processor asserts the address of the buffer and the necessary control signals on the bus. The bus transceivers recognize the address and enable the read/write control logic. The read/write control logic then generates the necessary control signals to select and read from the receive data buffer (low byte of RDSR). Data in the buffer is sent through the bidirectional tri-state buffer to the LSI-11 bus transceivers where it is enabled onto the LSI-11 bus and picked up by the processor. The USYNRT is double-buffered so that while the processor is picking up the character from the receive data buffer, the receive data shift register is already assembling a second character. This process is repeated until the entire message is received.

4.2.2.2 Transmit Operation – When the processor wishes to send data to the modem, it first places the address of the transmit buffer (low byte of TDSR) and the necessary control signals on the LSI-11 bus.

The bus transceivers recognize the address and enable the read/write control logic which selects the register. The processor then places the parallel data on the LSI-11 bus and the read/write control logic gates it through the bus transceivers and writes it into the transmit buffer. When a character is written into the transmit data buffer, the USYNRT transfers it to its transmit shift register and asserts TBEMTY. Once the character is in the shift register, the USYNRT begins to serialize and send it by means of the serial output line to the EIA drivers. Here it is converted from TTL to EIA level and sent to the modem.

TBEMTY causes the interrupt logic to generate an interrupt request to the processor. At the completion of the interrupt sequence, the processor repeats the process of addressing the transmit buffer and sending another character. This operation continues until the entire message has been sent.

4.3 DETAILED DESCRIPTION

The circuit operation is described in Paragraphs 4.3.1 through 4.3.9.

4.3.1 Bus Transceivers

Data, address and control signals move between the LSI-11 bus and the DPV11 by means of a group of bus transceivers. The bus transceivers are contained in four DC005 transceiver chips and perform the following functions.

- Address selection/decode
- Data transfers to and from the LSI-11 bus
- Vector generation

4.3.1.1 Address Selection – Each DPV11 is assigned four consecutive addresses that are decoded to generate control signals to enable five registers in the DPV11. Four addresses are able to access five registers because two of the registers (RDSR and PCSAR) share the same address. RDSR is a read-only register and PCSAR is a write-only register. Refer to Chapter 2 for address assignments.

When the software communicates with the DPV11, it does so by placing the address of the register it wishes to access and the necessary control signals on the LSI-11 bus. The DPV11 checks the address to see if it is within the range assigned to it. If so, access to the register is allowed. Paragraphs 4.3.1.2 through 4.3.1.4 describe the decoding of the address.

4.3.1.2 Address Decode – Address decoding is accomplished in the DC005 chips where a comparison is made of the BDAL03 through BDAL12 lines with the states selected by the address jumpers W29 through W39. (Refer to Chapter 2 for address selection and jumper connections). Each DC005 chip looks at three address lines and compares each of them against a corresponding jumper connection. When each address line agrees with its jumper input, the DC005 asserts pin 3 high. If all four DC005 chips have pin 3 asserted, the address on the bus is within the range assigned to this DPV11. When this condition exists, the register decode circuit is enabled to allow access to the specific register being addressed. Notice that BDAL00 through BDAL02 are not used in the address compare. Line zero is used in byte selection and lines one and two are used to select a particular register. Register selection and byte operation are discussed in Paragraph 4.3.2.

4.3.1.3 Bus Data Transfers – Once the address has been accepted and access to the selected register has been granted, data transfers can take place on the bus. The DC005 chips handle this function too. Consider first the operation in which the processor is sending data to a register in the DPV11. In this case, the DC005s would be placed in receive mode by a high on pin 4. This is a result of control signals placed on the bus by the processor. In the receive mode, data on the BDAL0 through BDAL15 lines is passed through the DC005 and made available to the register on the DA0 through DA15 lines.

When the processor is requesting information from one of the DPV11 registers, the DC005s are placed in transmit mode by a high on pin 5. In the transmit mode, data from the selected register is presented to the DC005s on the DA0 through DA15 lines. The DC005s then pass this data to the bus on the BDAL0 through BDAL15 lines.

4.3.1.4 Vector Generation – A third function of the DC005 chips is vector generation. This is accomplished by daisy-chain strapping W40 through W45 to W46 in the proper configuration for the vector address desired. Refer to Chapter 2 for information on vector assignments and jumper connections. W46 is high when the vector is to be sent to the processor. The signal VECTOR H is asserted by the interrupt logic during an interrupt sequence. W45 corresponds to BDAL3 and W43 corresponds to BDAL8.

4.3.2 Read/Write Control Logic

The read/write control logic contains circuits for controlling register decoding, USYNRT operations, and BRPLY. A description of each follows.

4.3.2.1 Register Decode (Figure 4-3) – The selection of individual registers within the DPV11 is accomplished by a DC004 protocol chip and its associated logic. This circuit is enabled by an address match from the DC005s. When enabled, the DC004 decodes address lines 1 and 2 to produce one of four select signals. These select lines, however, do not directly select the registers. Two registers share the same address, one being a read-only and the other, a write-only register. One entire register and the low byte of another are external to the USYNRT. For these reasons, additional gates are used with the select lines to properly select the one register in five to be accessed. These gates use byte and write signals to aid in the register selection. Table 4-1 shows the register selection based on the three low-order address bits.

NOTE

All registers can be accessed in either word or byte mode. However, reading either byte of the RXCSR resets certain status bits in both bytes.

Reading either byte of the RDSR resets data and certain status bits in both bytes of this register as well as bits 7 and 10 of the RXCSR.

NOTE

The address inputs to the DC004 are inverted, thereby causing a reverse order on the select lines. Pin 17 corresponds to select 0 and pin 14 corresponds to select 6. This applies also to the OUTLB (pin 13) and OUTHB (pin 12).

4.3.2.2 USYNRT Control – Most of the control signals for the USYNRT are generated by the DC004 and its associated logic. This paragraph describes the control signals and their functions.

ADR0, ADR1, and ADR2 are used to select a register within the USYNRT. They are encoded as shown in Table 4-2. ADR0 is used in conjunction with BYTE OP to select a byte.

WRITE USYNRT is used to control writing into or reading from registers within the USYNRT. When it is asserted, a write operation is indicated. When it is not asserted, a read operation is indicated. WRITE USYNRT is generated by ORing the OUTLB and OUTHB signals from the DC004. OUTLB and OUTHB are used to write data into the low byte, high byte or both bytes of a selected register. They are generated by the DC004 in response to the bus signals BWTBT, BDOUT, and BDAL0. OUTLB and OUTHB do not directly control byte selection for the USYNRT but are used to generate ADR0 and BYTE OP.

BYTE OP is used to indicate to the USYNRT that a byte operation is to be performed on the selected register. It is generated during a write operation when either OUTLB or OUTHB but not both are asserted.

DPENA (Data Port Enabled) is used to enable the tri-state data bus of the USYNRT and supply the necessary timing for transactions between the USYNRT and the external circuits. DPENA strobes the data for write or read operations. It is generated from the register select signals and the output of pin 8 of the DC004 chip which results directly from BDIN or BDOUT. Deskew delay is accomplished by using a 74LS164 serial to parallel shift register. Pin 8 of the DC004 is used as the serial input to the shift register which is clocked by a 100 ns clock. Initially the serial input is high and the shift register outputs are all high. 100 to 200 ns after the serial in goes low, DPENA becomes asserted to strobe the USYNRT. DPENA remains asserted for at least 300 ns as determined by pin 10 of the shift register. For read operations, DPENA will remain asserted until BDIN becomes not asserted. This is to ensure that the data is on the bus when the processor strobes it. For write operations DPENA will be asserted for 300 ns.

BRPLY (Bus Reply) indicates to the processor that the DPV11 has placed data on the bus or has received data from the bus. It is generated from the same circuit as DPENA and is asserted 300 ns after DPENA. BRPLY remains asserted until the processor responds by negating BDIN or BDOUT.

Figure 4-4 shows the timing for the generation of DPENA and BRPLY for a read operation. Figure 4-5 shows the timing for a write operation.

4.3.3 USYNRT, RXCSR, and PCSCR

Most of the registers used in the DPV11 are contained within the USYNRT. The receive control and status register (RXCSR) and the low byte of the parameter control and character length register (PCSCR) are external to the USYNRT. The USYNRT and the external registers are discussed in Paragraphs 4.3.3.1 through 4.3.3.3.

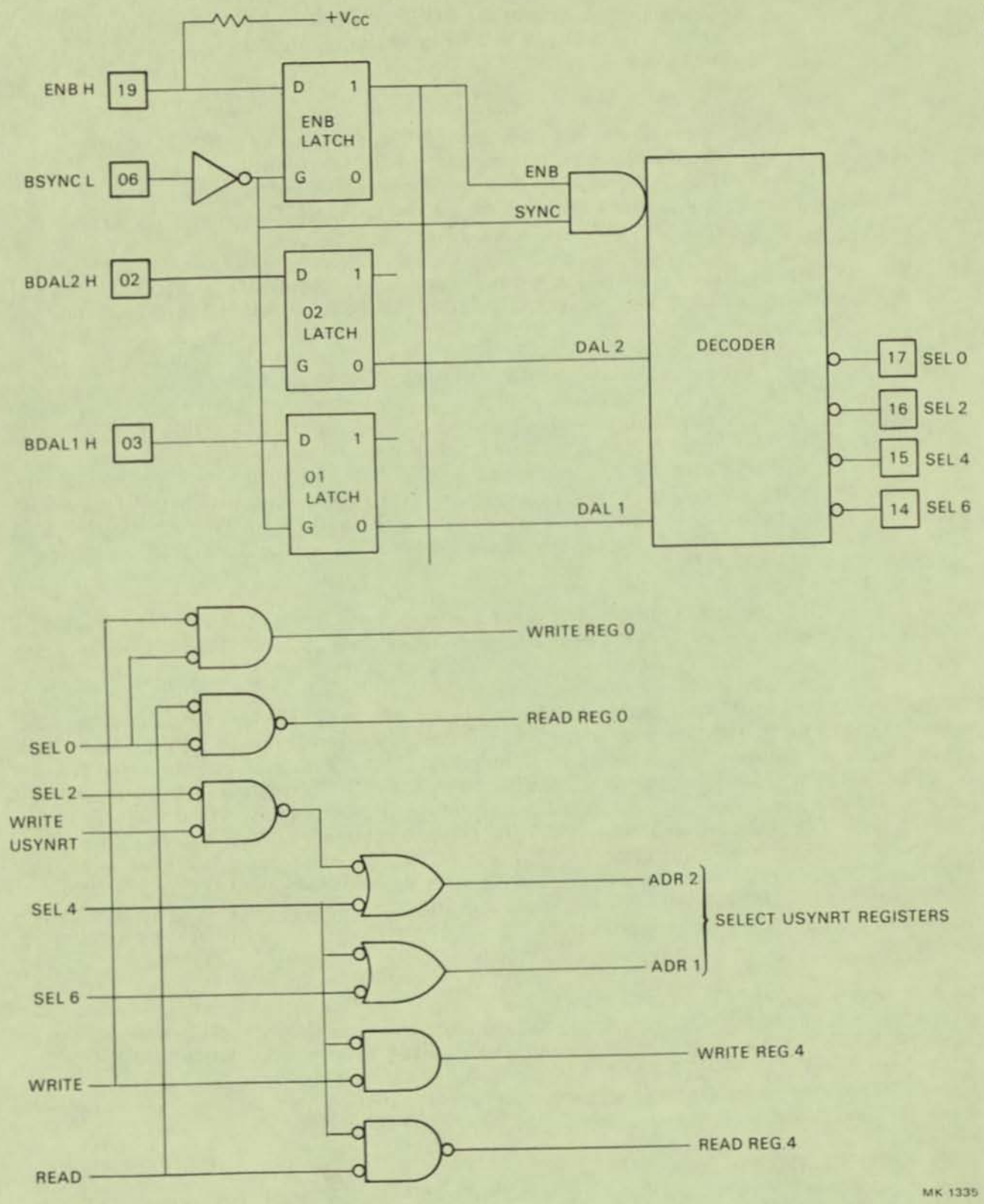


Figure 4-3 Register Decode

Table 4-1 Register Selection

A2	A1	A0	Register
0	0	0	RXCSR (word or low byte)
0	0	1	RXCSR (high byte)
0	1	0	RDSR (read) or PCSAR (write)
0	1	1	RDSR or PCSAR (high byte)
1	0	0	PCSCR (word or low byte)
1	0	1	PCSCR (high byte)
1	1	0	TDSR (word or low byte)
1	1	1	TDSR (high byte)

Table 4-2 USYNRT Register Select

ADR2	ADR1	Register
0	0	RDSR (read only)
0	1	TDSR
1	0	PCSAR (write only)
1	1	PCSCR (high byte)

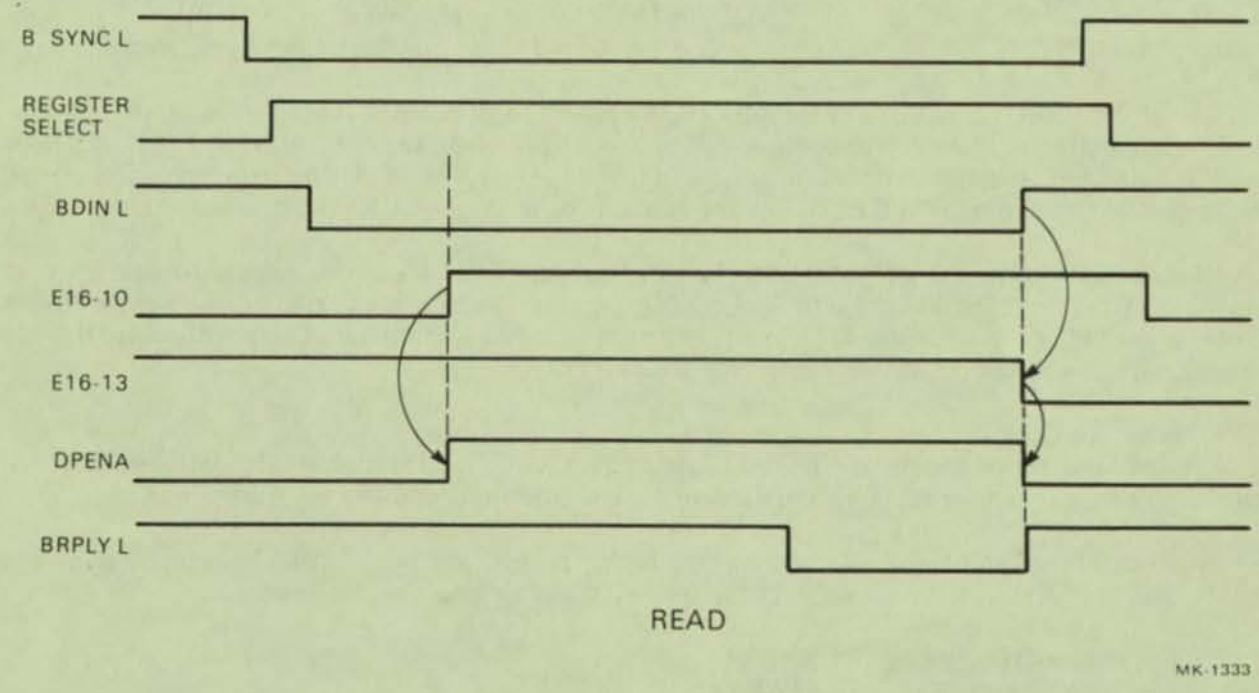


Figure 4-4 Timing for Read Operation

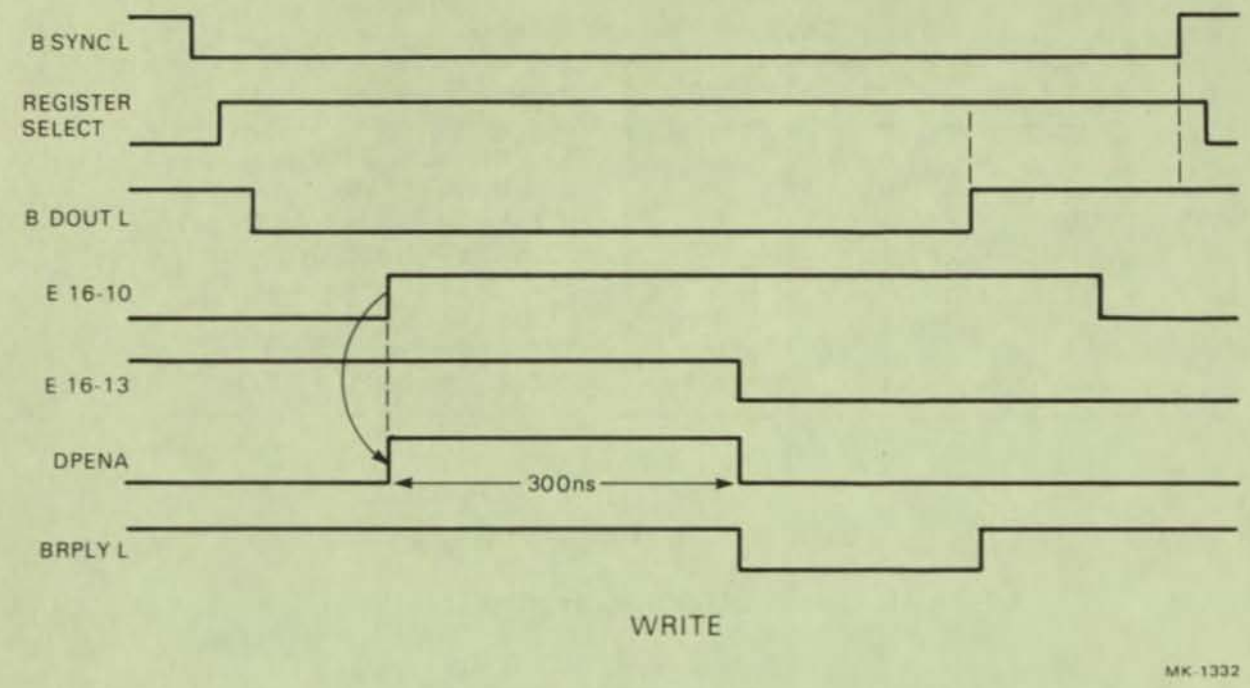


Figure 4-5 Timing for Write Operation

4.3.3.1 USYNRT – The universal synchronous receiver/transmitter (USYNRT) functions as a large scale integration (LSI) subsystem for synchronous communications. The USYNRT provides the logic support, via program parameter registers, for basic protocol handling and error detection. Protocol handling by the USYNRT conforms to standards imposed by these protocols, but is slightly different in each version of the USYNRT. The 5025 (2112517-00) is implemented in the DPV11. For more details on the USYNRT, refer to Appendix B or to A-PS-2112517-0-0 Purchase Specification.

4.3.3.2 Receive Control and Status Register (RXCSR) – The RXCSR is described in detail in Chapter 3. It is a buffer and line driver consisting of two 74LS244 chips and one 74LS174 hex D flip-flop. The low byte can be read or written into but the high byte can only be read. The write operation occurs on the positive transition of WREG0. The register can be read when RREG0 is asserted low.

4.3.3.3 Parameter Control and Character Length Register (PCSCR) – This register is described in Chapter 3. Its upper byte is internal to the USYNRT and its low byte is external. Three bits (0, 3, and 4) of the low byte of this register are directly program-writable with bit zero being write-only. Bit 6 is program writable but is a function of the interrupt circuit.

4.3.4 Interrupt Logic

Most of the logic for interrupts is contained in a single DC003 interrupt chip. The chip contains two interrupt channels: one for receiver and modem control interrupts and one for transmitter interrupts.

The receive and modem control interrupt has the higher priority and may occur when receive interrupt enable (RX INT ENA) is set and any of the following signals become asserted.

Receive Data Ready (RDATRY)
Receive Status Ready (RSTARY)
Data Set Change (DAT SET CH)

Notice that DAT SET CH requires that MC INT ENA (RXCSR bit 5) also be asserted.

When a register in the receive section (RXCSR or RDSR) is accessed; i.e., when servicing a receive interrupt request, the receive interrupt request is disabled for 600 ns by the output on pin 5 of the 74LS74 flip-flop. This is done to ensure that any modem control interrupt request that might have occurred while servicing the receive interrupt request, is recognized. When the flip-flop is reset by the 600 ns signal, a negative to positive transition is recognized on pin 17 of the DC003 if a modem control interrupt request is present.

A transmitter interrupt is generated by the DC003 if the TBEMTY signal is asserted when transmitter interrupt enable (bit 6 of PCSCR) is set.

Both the TX INT ENA and RX INT ENA bits are located physically in the DC003 interrupt chip although they are functionally part of the PCSCR and RXCSR respectively.

The bus interrupt request (BIRQ) is asserted by the DC003 for either a receive or transmit interrupt request. The processor responds to BIRQ by asserting BIAKI and BDIN. BIAKI is the interrupt acknowledge signal. It is passed down the priority chain until it reaches the section of the interrupt chip that initiated the request.

When the interrupt logic receives both BDIN and BIAKI, it asserts the signal VECTOR. VECTOR enables the assertion of the vector address by the DS005s. If the interrupt is a transmitter interrupt, the RQSTB signal would assert vector address bit 2.

4.3.5 Data Set Change Circuit (Transition Detector)

The data set change circuit consists of a 74LS273 D-register, exclusive NOR gates and two flip-flops. Setting of the Data Set Change bit (DAT SET CH) is determined by the configuration of jumpers W24 through W28. Any or all of the following modem signals can set DAT SET CH if its associated jumper is installed.

RS-232-C	RS-449
Clear to Send (CTS)	Clear to Send (CTS)
Carrier Detect (CD)	Receiver Ready (RR)
Data Set Ready (DSR)	Data Mode (DM)
Ring Indicator (RI)	Incoming Call (IC)

NOTE

The modem change circuit interprets any pulse of two microseconds or greater duration as a modem status change. This ensures that all legitimate modem status changes will be detected. However, on a poor line, noise may be interpreted as a modem status change. Software written for the DPV11 must account for this possibility.

4.3.6 Clock Circuit

The clock circuit consists of an off-the-shelf 19.6608 MHz crystal oscillator, and two 74LS390 counters. The 19.6608 MHz signal is divided by the counter circuits to produce the following four clock signals.

1. LOCAL CLK (49.152 kHz) – Normally jumpered to NULL MODEM CLK (W18 and W21) and used as the data clock.

2. DIAG CLK (1.9661 kHz) – Nonsymmetrical clock available for diagnostic purposes (not recommended for local communications). It becomes the transmit clock when the DPV11 is placed in diagnostic mode. DIAG CLK can also be jumpered to LOCAL CLK for 50 kHz operation but some of the tests must be omitted.

3. SR CLK (9.8304 MHz) – Used to clock the shift register to establish delays for DPENA and BRPLY.

4. Charge PUMP CLK (491.52 kHz) – Used by the charge pump circuit and transition detector.

4.3.7 USYNRT Timing – USYNRT timing for the transmit and receive sections originates with the modem and is gated through the AND-OR inverter to the USYNRT.

During normal receive data transfers, the 74LS51 gates receiver timing from the modem as receive clock pulse (RCP) to the USYNRT. If the modem clock stops with the last valid data bit, Receiver Ready becomes not asserted. The next positive transition of the NULL MODEM CLK causes 74LS74 pin 8 to go high, thus substituting NULL MODEM CLK for modem receive timing. In this way, the USYNRT receives the necessary 16 clock pulses to complete its operation after the modem has stopped sending.

During normal transmit data transfers, timing for the USYNRT is gated from the modem through the 74LS51 pin 6 to the USYNRT.

In maintenance mode, the signal MSEL disables the modem timing and enables the DIAG CLK as the clock for the USYNRT.

4.3.8 EIA Receivers
26LS32 quad differential line receivers are used to accept signals and data from the modem. Jumpers W12 through W17 are terminating resistors which may be connected for RS-422-A but must be disconnected for RS-423-A.

4.3.9 EIA Drivers
Two types of drivers are used to send signals and data to the modem. 9638 drivers are used for RS-422-A and 9636 drivers are used for RS-423-A.

4.3.10 Maintenance Mode
The USYNRT is placed in maintenance mode by setting Maintenance Mode Select (bit 3 of the PCSCR). When this happens, the serial output of the transmit section is internally looped back as serial input and the transmit serial output is held asserted. All clocking of both the receive and transmit sections is controlled by the transmitter clock input. This signal is derived from the 2 kHz clock as determined by the 74LS51 AND-OR inverter.

CHAPTER 5
MAINTENANCE

5.1 SCOPE

This chapter provides a complete maintenance procedure for the DPV11 and includes a list of required test equipment and diagnostics. The maintenance philosophy and procedures for preventive and corrective maintenance are discussed.

5.2 TEST EQUIPMENT RECOMMENDED

Maintenance procedures for the DPV11 require the test equipment and diagnostic programs listed in Table 5-1.

Table 5-1 Test Equipment Recommended

Equipment	Manufacturer	Designation
Multimeter	Triplett or Simpson	Model 630-NA or 260 or equivalent
Oscilloscope	Tektronix	Type 453 or equivalent
X10 Probes (2)	Tektronix	P6008 or equivalent
Module extenders	DIGITAL	W984 (double)
Cable turn-around connector	DIGITAL	H3259
On-board test connector	DIGITAL	H3260
Breakout box	IDS	
LIB kit	DIGITAL	ZJ314-RB
Document only		ZJ314-RZ
Document and paper tape		ZJ314-RB
Paper tape only		ZJ314-PB
Fiche		ZJ314-FR

5.3 MAINTENANCE PHILOSOPHY

The basic approach to DPV11 fault isolation is the use of stand-alone diagnostic programs and maintenance mode features supported by the hardware.

Typical applications of the DPV11 do not allow lengthy troubleshooting sessions; therefore, the maintenance philosophy in the field is module swapping. The defective module is returned to the factory for repair.

5.4 PREVENTIVE MAINTENANCE

There is no scheduled preventive maintenance for the DPV11. Preventive maintenance for the DPV11 is integrated into its corresponding system preventive maintenance and consists of checking the power supply voltage. Whenever the module or cables have been disturbed, diagnostics (specifically DEC/X11 and DCLT) should be run to verify proper operation.

5.5 CORRECTIVE MAINTENANCE

Since the field replaceable units are the M8020 and the cables, all diagnosis should be directed to isolation of one of these components.

NOTE

The operating jumper configuration of the DPV11 module being serviced should be recorded prior to any changes for maintenance purposes. This will facilitate reconfiguring the module when the service activity is complete.

5.5.1 Maintenance Mode

To aid in troubleshooting, the DPV11 has a software-selectable maintenance mode which causes the serial output of the USYNRT to be internally connected to its serial input. When in maintenance mode, serial data from the modem is disabled and the send and receive timing from the modem are replaced with a clock signal generated on the M8020 module. The clock rate is 2K b/s.

The diagnostics normally operate with and without the Maintenance Mode Select (MSEL) bit set. In this way the USYNRT chip can be isolated from the remainder of the circuitry.

5.5.2 Loopback Connectors

The cable loopback connector shipped with the DPV11-DB (bundled version) is the H3259. This connector is connected to the modem end of the BC26L cable when it is used. No cables or test connectors are shipped with the DPV11-DA (unbundled versions). An on-board connector (H3260) can be purchased separately (see Paragraph 2.5 and Figure 2-4) for connecting to J1 on the M8020 module. It provides for testing of all M8020 logic.

5.5.3 Diagnostics

DPV11 diagnostics aid in the isolation process and should be run when a malfunction is indicated. Diagnostics should also be run to verify operation after repair.

NOTE

To ensure that all M8020 logic circuits are checked, on-board test connector H3260 must be used. However, the DPV11 system cannot be assumed to be thoroughly checked unless the DIGITAL-supplied cable is also tested.

Diagnostics must be run with a cable turn-around connector (H3259) at the modem end of the BC26L-25 cable.

The following diagnostics are available to aid in the isolation and verification process.

5.5.3.1 CVDPV* Functional Diagnostic – CVDPV* is designed to verify the functionality of the DPV11. No resolution to the chip is intended. CVDPV* is a stand-alone program that must be executed under control of the PDP-11 diagnostic supervisor (DS). Errors are reported as they occur in the program, unless they are inhibited, and conform to the DS error report format. For information on loading and running of the DS, see Appendix A.

CVDPV* is compatible with XXDP+, ACT/SLIDE, APT or ABS. It consists of a number of tests which function as follows.

Test No.	Description
1	Verifies that addressing the RXCSR does not cause a nonexistent memory trap.
2	Verifies that the DPV11 may be properly initialized by a Master Clear or LSI-11 Reset.
3	Writes and reads data patterns into all writable bits to verify bit validity and addressing paths.
4	Enables and ensures that the transmitter is activated.
5	Verifies that TBEMTY is asserted and cleared properly for all possible conditions.
6	Verifies proper operation of the transmit interrupt.
7	Enables and ensures that the receiver is activated, and RDATRY is asserted properly.
8	Verifies proper operation of the receive interrrupt for the reception of data.
9	Verifies proper operation of RSTARY for all possible conditions.
10	Verifies proper operation of the receive interrupt for status.
11	Ensures that both transmit and receive interrupts are recognized.
12	Verifies proper operation of all modem status bits and ensures that DSCNG is set when a transition occurs.
13	Verifies that an interrupt is received when DSCNG is set.
14	Verifies that if a DSCNG occurs during a receive interrupt, it will be recognized.
15–20	Verifies proper operation with bit-oriented protocols (BOP).
21–23	Verifies proper operation with byte count-oriented protocols (BCP).
24–28	Verifies CRC and VRC functions.
29	Verifies maintenance mode noninterrupt data operations.
30–36	Verifies BOP data operation.
37–40	Verifies BCP data operation.

41 Verifies DDCMP message protocol and message transmission.

42 Verifies high-speed BCP data operation.

43 Verifies high-speed BOP data operation.

5.5.3.2 DEC/X11 CXDPV Module – CXDPV exercises up to six consecutively addressed DPV11 synchronous interfaces as they relate to the total system configuration. It is very useful in determining whether a DPV11 is the failing component among other system components in a system environment. It is a system exerciser and does not operate as a stand-alone test. It must be configured and run as part of a total system exerciser.

The DEC/X11 System Exerciser must be run after the stand-alone diagnostic CVDPV* has been run. It determines if the DPV11 or another device adversely affects the total system operation. For more information on DEC/X11, refer to the *DEC/X11 User Manual* (AC-FO53B-MC) and *DEC/X11 Cross-Reference* (9AC-F055C-MC)

5.5.3.3 Data Communications Link Test CVCLH* (DCLT) – DCLT is a communications equipment maintenance tool designed to isolate failures to either the interface, the telephone communication line, or the modem. It exercises DPV11 to DPV11 links.

DCLT is XXDP+ or APT compatible and runs under control of the diagnostic supervisor (DS) (see Appendix A). It requires 24K of memory. For more information on DCLT refer to CVCLH* document AC-F582A-MC.

APPENDIX A
DIAGNOSTIC SUPERVISOR SUMMARY

A.1 INTRODUCTION

The PDP-11 diagnostic supervisor is a software package that performs the following functions.

- Provides run-time support for diagnostic programs running on a PDP-11 in stand-alone mode
- Provides a consistent operator interface to load and run a single diagnostic program or a script of programs
- Provides a common programmer interface for diagnostic development
- Imposes a common structure upon diagnostic programs
- Guarantees compatibility with various load systems such as APT, ACT, SLIDE, XXDP+, ABS Loader
- Performs nondiagnostic functions for programs, such as console I/O, data conversion, test sequencing, program options

A.2 VERSIONS OF THE DIAGNOSTIC SUPERVISOR

File Name	Environment
HSAA **.SYS	XXDP+
HSAB **.SYS	APT
HSAC **.SYS	ACT/SLIDE
HSAD **.SYS	Paper Tape (Absolute Loader)

In the above file names, "**" stands for revision and patch level, such as "A0".

A.3 LOADING AND RUNNING A SUPERVISOR DIAGNOSTIC

A supervisor-compatible* diagnostic program may be loaded and started in the normal way, using any of the supported load systems. Using XXDP+ for example, the program CVDPVA.BIN is loaded and started by typing .R CVDPVA.

The diagnostic and the supervisor will automatically be loaded as shown in Figure A-1 and the program started. The program types the following message.

```
DRS LOADED
DIAG.RUN-TIME SERVICES
CVDPV-A-0
```

*To determine if diagnostics are supervisor-compatible, use the List command under the Setup utility (see Paragraph A.5.).

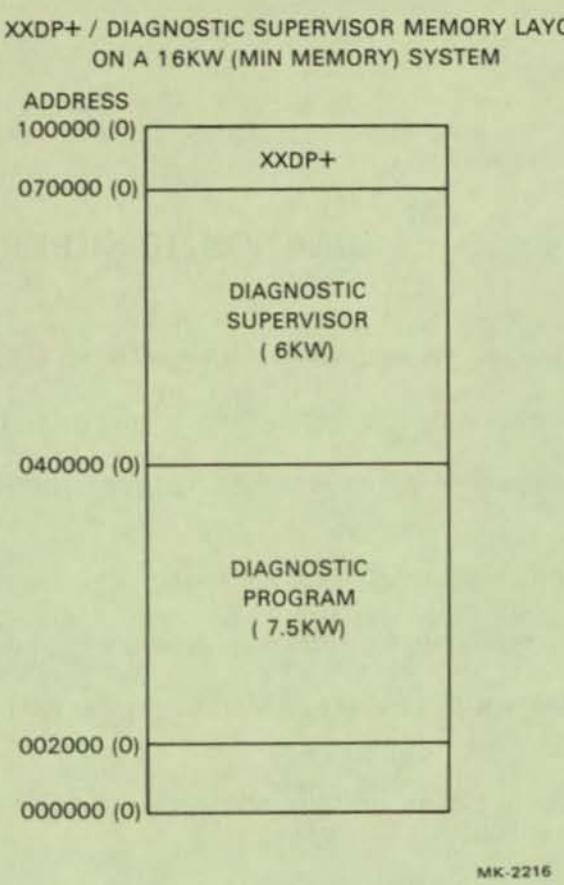


Figure A-1 Typical XXDP+/Diagnostic Supervisor Memory Layout

```
DIAGNOSTIC TESTS
UNIT IS DPV11
DR>
```

DR> is the prompt for the diagnostic supervisor routine. At this point a supervisor command must be entered (the supervisor commands are listed in Paragraph A.4).

Five Steps to Run a Supervisor Diagnostic

1. Enter Start command.

   When the prompt DR> is issued, type:

   STA/PASS:1/FLAGS:HOE <CR>

   The switches and flags are optional.

2. Enter number of units to be tested.

   The program responds to the Start command with:

   # UNITS?

   At this point enter the number of devices to be tested.

3. Answer hardware parameter questions.

   After the number of devices to be tested has been entered, the program responds by asking a number of hardware questions. The answers to these questions are used to build hardware parameter tables in memory. A series of questions is posed for each device to be tested. A "Hardware P-Table" is built for each device.

4. Answer software parameter questions.

   When all the "Hardware P-Tables" are built, the program responds with:

   CHANGE SW?

   If other than the default parameters are desired for the software, type Y. If the default parameters are desired, type N.

   If you type Y, a series of software questions will be asked and the answers to these will be entered into the "Software P-Table" in memory. The software questions will be asked only once, regardless of the number of units to be tested.

5. Diagnostic execution.

   After the software questions have been answered, the diagnostic begins to run.

   What happens next is determined by the switch options selected with the Start command, or errors occurring during execution of the diagnostic.

A.4 SUPERVISOR COMMANDS

The supervisor commands that may be issued in response to the DR> prompt are as follows.

- Start – Starts a diagnostic program.
- Restart – When a diagnostic has stopped and control is given back to the supervisor, this command restarts the program from the beginning.
- Continue – Allows a diagnostic to continue running from where it was stopped.
- Proceed – Causes the diagnostic to resume with the next test after the one in which it halted.
- Exit – Transfers control to the XXDP+ monitor.
- Drop – Drops units specified until an Add or Start command is given.
- Add – Adds units specified. These units must have been previously dropped.
- Print – Prints out statistics if available.
- Display – Displays P-Tables.
- Flags – Used to change flags.
- ZFLAGS – Clears flags.

All of the supervisor commands except Exit, Print, Flags, and ZFLAGS can be used with switch options.

A.4.1 Command Switches

Switch options may be used with most supervisor commands. The available switches and their function are as follows.

- ./TESTS: – Used to specify the tests to be run (the default is all tests). An example of the tests switch used with the Start command to run tests 1 through 5, 19, and 34 through 38 would be:

  DR> START/TESTS : 1-5 : 19 : 34-38 <CR>

- ./PASS: – Used to specify the number of passes for the diagnostic to run. For example:

  DR> START/PASS : 1

  In this example, the diagnostic would complete one pass and give control back to the supervisor.

- ./EOP: – Used to specify how many passes of the diagnostic will occur before the end of pass message is printed (the default is one).

- ./UNITS: – Used to specify the units to be run. This switch is valid only if N was entered in response to the CHANGE HW? question.

- ./FLAGS: – Used to check for conditions and modify program execution accordingly. The conditions checked for are as follows.

  :HOE –Halt an error (transfers control back to the supervisor)

  :LOE – Loop on error

  :IER – Inhibit error reports

  :IBE – Inhibit basic error information

  :IXE – Inhibit extended error information

  :PRI – Print errors on line printer

  :PNT – Print the number of the test being executed prior to execution

  :BOE – Ring bell on error

  :UAM – Run in unattended mode, bypass manual intervention tests

  :ISR – Inhibit statistical reports

  :IOU – Inhibit dropping of units by program

A.4.2 Control/Escape Characters Supported

The keyboard functions supported by the diagnostic supervisor are as follows.

- CONTROL C (↑C) – Returns control to the supervisor. The DR> prompt would be typed in response to CONTROL C. This function can be typed at any time.

- CONTROL Z (↑Z) – Used during hardware or software dialogue to terminate the dialogue and select default values.
- CONTROL O (↑O) – Disables all printouts. This is valid only during a printout.
- CONTROL S (↑S) – Used during a printout to temporarily freeze the printout.
- CONTROL Q (↑Q) – Resumes a printout after a CONTROL S.

A.5 THE SETUP UTILITY

Setup is a utility program that allows the operator to create parameters for a supervisor diagnostic prior to execution. This is valid for either XXDP+ or ACT/SLIDE environments. Setup asks the hardware and software questions and builds the P-Tables.

The following commands are available under Setup.

List – list supervisor diagnostics
Setup – create P-Tables
Exit – return control to the supervisor

The format for the List command is:

LIST DDN:FILE.EXT

Its function is to type the file name and creation date of the file specified if it is a revision C or later supervisor diagnostic. If no file name is given, all revision C or later supervisor diagnostics are listed. The default for the device is the system device, and wild cards are accepted.

The format for the Setup command is:

SETUP DDN:FILE.EXT=DDN:FILE.EXT

It reads the input file specified and prompts the operator for information to build P-Tables. An output file is created to run in the environment specified. File names for the output and input files may be the same. The output and input device may be the same. The default for the device is the system device and wild cards are not accepted.

APPENDIX B
USYNRT DESCRIPTION

5025 Universal Synchronous Receiver/Transmitter (USYNRT)
The data paths of the USYNRT provide complete serialization, deserialization and buffering. Output signals are provided to the USYNRT controller to indicate the state of the data paths, the command fields or recognition of extended address fields. These tasks must be performed by the USYNRT controller.

The USYNRT is a 40-pin dual-in-line package (DIP). Figure B-1 is a terminal connection (identification) diagram.

Data port bits DP07:DP00 are dedicated to service four 8-bit wide registers. Bits DP15:DP08 service either control information or status registers. The PCSCR register is reserved. (See Figure B-2.)

Purchase Specification 2112517-0-0 provides a detailed description of the 5025 USYNRT.

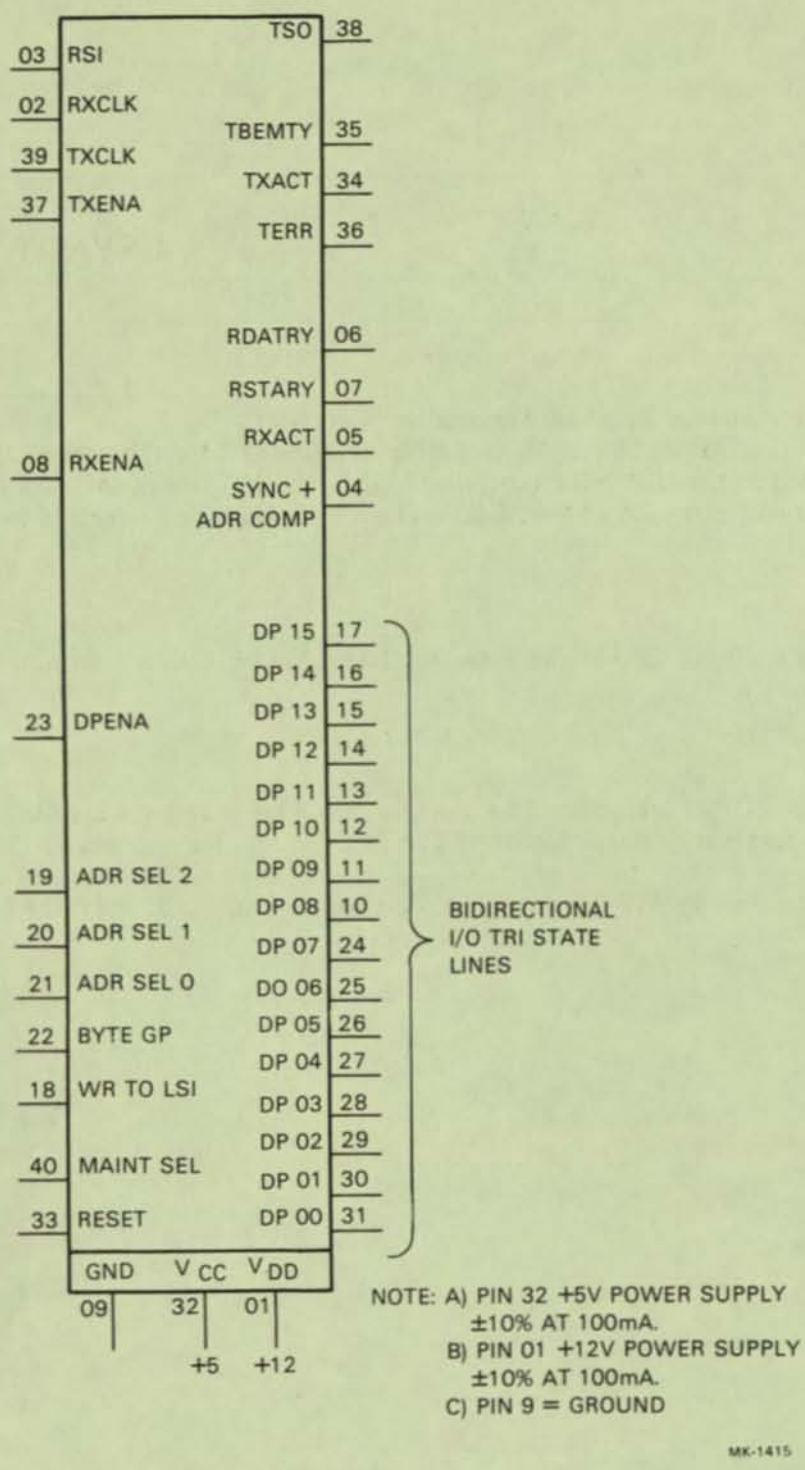


Figure B-1 Terminal Connection Identification Diagram (2112517-0-0 Variation)

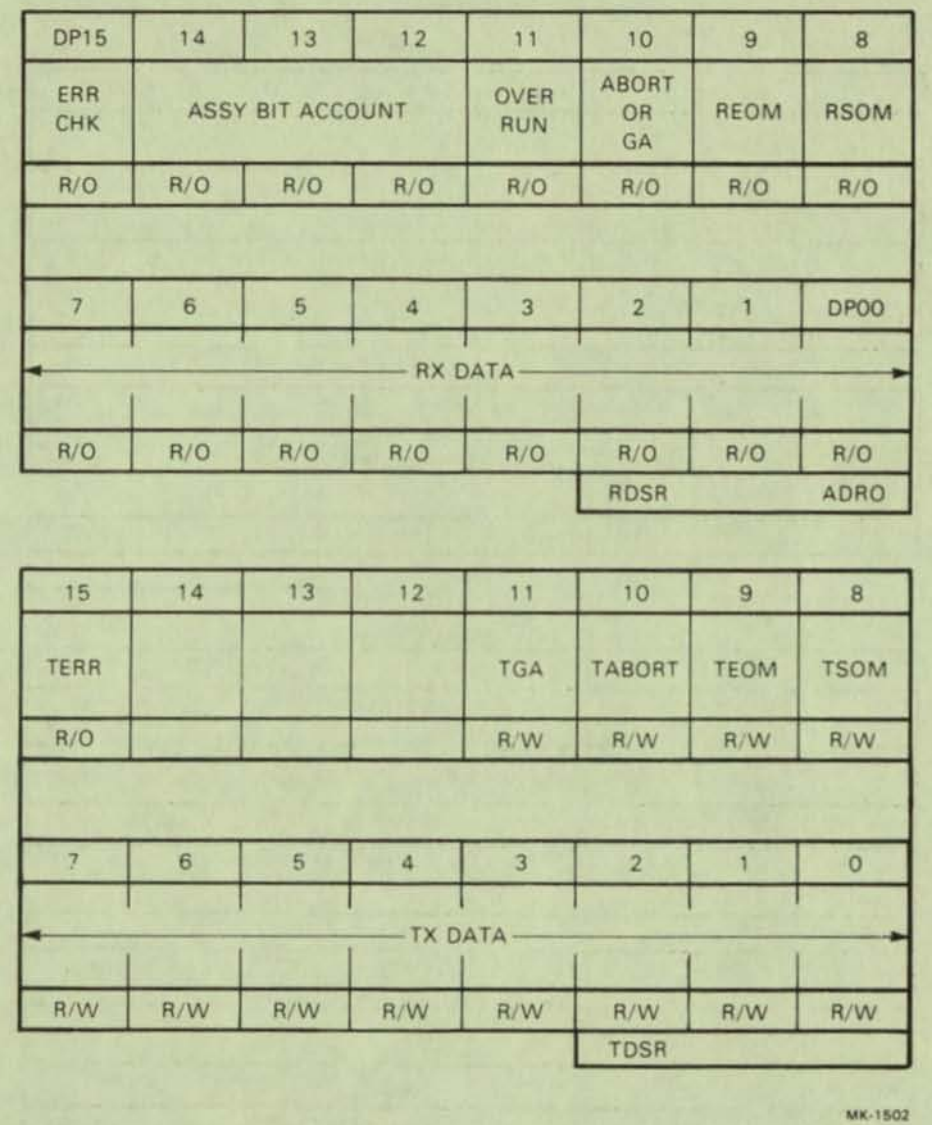


Figure B-2 5025 Internal Register Bit Map (2112517-0-0 Variation) (Sheet 1 of 2)

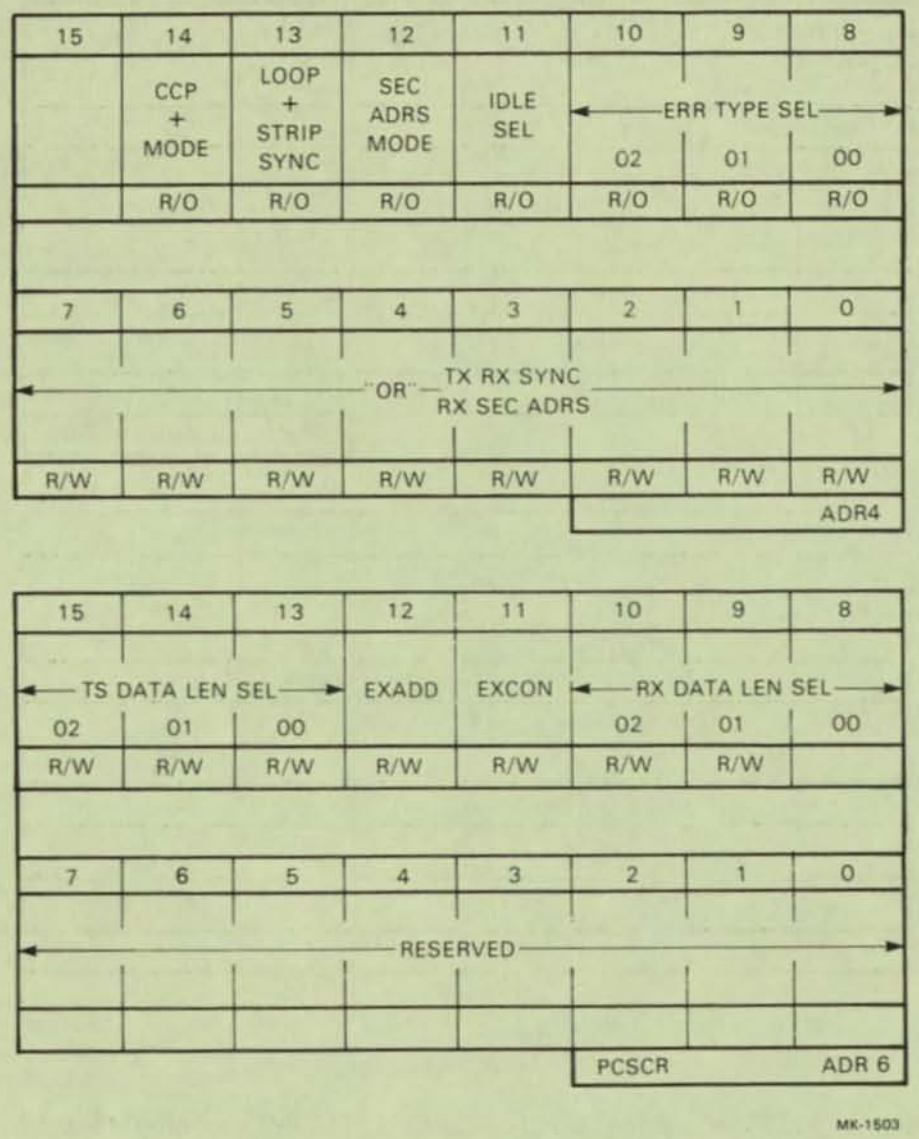


Figure B-2 5025 Internal Register Bit Map (2112517-0-0 Variation) (Sheet 2 of 2)

APPENDIX C
IC DESCRIPTIONS

C.1 GENERAL

This appendix contains data on the LSI-11 chips and some of the unusual ICs used by the DPV11. The other ICs are common, widely-used logic devices. Detailed specifications on these chips are readily available, and hence are not included here.

C.2 DC003 INTERRUPT CHIP

The interrupt chip is an 18-pin DIP device. It provides the circuits to perform an interrupt transaction in a computer system that uses a "pass-the-pulse" type arbitration scheme. The device provides two interrupt channels labeled A and B, with the A section at a higher priority than the B section. Bus signals use high-impedance input circuits or high-drive open-collector outputs, which allow the device to directly attach to the computer system bus. Maximum current required from the V_{cc} supply is 140 mA.

Figure C-1 is a simplified logic diagram of the DC003 IC. Table C-1 describes the signals and pins of the DC003.

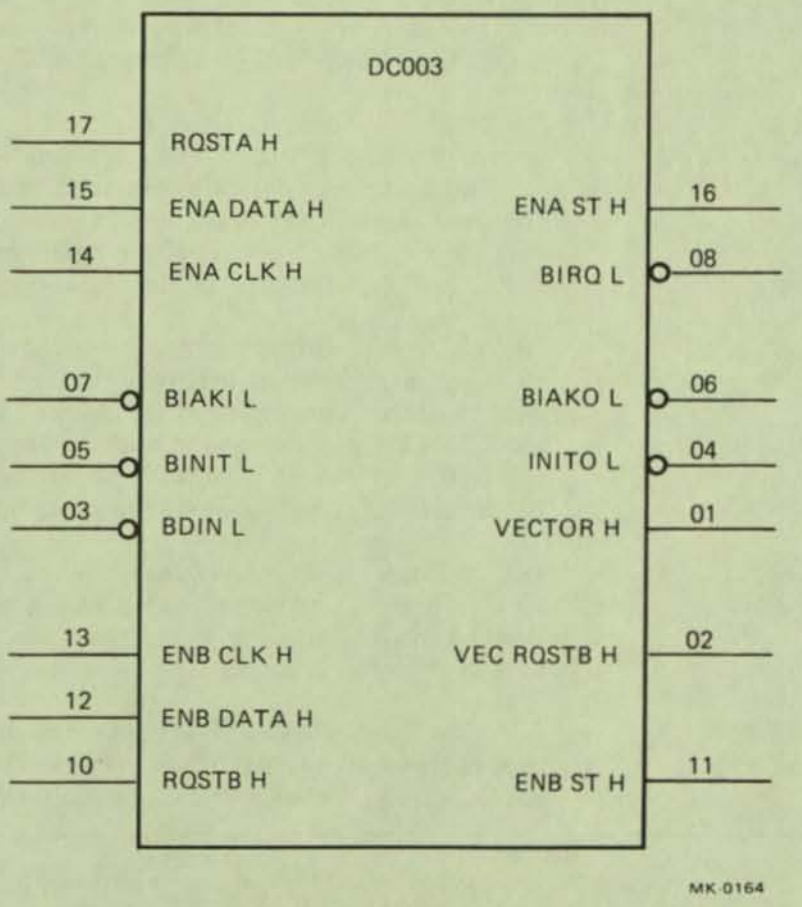


Figure C-1 DC003 Logic Symbol

Table C-1 DC003 Pin/Signal Descriptions

Pin	Signal	Description
1	VECTOR H	Interrupt Vector Gating Signal – This signal gates the appropriate vector address onto the bus and forms the bus signal BRPLY L. Not used in the DPV11.
2	VEC RQSTB H	Vector Request B Signal – When asserted, this signal indicates RQST B service vector address is required. When negated, it indicates RQST A service vector address is required. VECTOR H is the gating signal for the entire vector address; VEC RQST B H is normally bit 2 of the address.
3	BDIN L	Bus Data In – THE BDIN signal always precedes a BIAK signal.
4	INITO L	Initialize Out – This is the buffered BINIT L signal used in the device interface for general initialization.
5	BINIT L	Bus Initialize – When asserted, this signal brings all drive lines to their negated state (except INITO L).
6	BIAKO L	Bus Interrupt Acknowledge – This signal is the daisy-chained signal that is passed by all devices not requesting interrupt service (see BIAKI L). Once passed by a device, it must remain passed until a new BAIKI L is generated.
7	BIAKI L	Bus Interrupt Acknowledge – This signal is the processor's response to BIRQ L true. This signal is daisy-chained such that the first requesting device blocks the signal propagation while nonrequesting devices pass the signal on as BIAKO L to the next device in the chain. The leading edge of BIAKI L causes BIRQ L to be unasserted by the requesting device.
8	BIRQ L	Asynchronous Bus Interrupt Request – The request is generated by a true RQST signal along with the associated true Interrupt Enable signal. The request is removed after the acceptance of the BDIN L signal and on the leading edge of the BAIKI L signal, or the removal of the associated interrupt enable, or due to the removal of the associated request signal.
17 10	RQSTA H RQSTB H	Device Interrupt Request Signal – When asserted with the enable A/B flip-flop asserted, this signal causes the assertion of BIRQ L on the bus. This signal line normally remains asserted until the request is serviced.
16 11	ENA ST H ENB ST H	Interrupt Enable – This signal indicates the state of the interrupt enable A/B internal flip-flop which is controlled by the signal line ENA/B DATA H and the ENA/B CLK H clock line.

Table C-1 DC003 Pin/Signal Descriptions (Cont)

Pin	Signal	Description
15 12	ENA DATA H ENB DATA	Interrupt Enable Data – The level on this line, in conjunction with the ENA/B CLK H signal, determines the state of the internal interrupt enable A flip-flop. The output of this flip-flop is monitored by the ENA/B ST H signal.
14 13	ENA CLK H ENB CLK H	Interrupt Enable Clock – When asserted (on the positive edge), interrupt enable A/B flip-flop assumes the state of the ENA/B DATA H signal line.

C.3 DC004 PROTOCOL CHIP

The protocol chip is a 20-pin DIP device that functions as a register selector, providing the signals necessary to control data flow into and out of up to four word registers (8 bytes). Bus signals can directly attach to the device because receivers and drivers are provided on the chip. An RC delay circuit is provided to slow the response of the peripheral interface to data transfer requests. The circuit is designed such that if tight tolerance is not required, then only an external 1K $\times$20 percent resistor is necessary. External RCs can be added to vary the delay. Maximum current required from the V_{cc} supply is 120 mA.

Figure C-2 is a simplified logic diagram of the DC004 IC. Signal and pin definitions for the DC004 are shown in Table C-2.

C.4 DC005 BUS TRANSCEIVER CHIP

The 4-bit transceiver is a 20-pin DIP, low-power Schottky device for primary use in peripheral device interfaces, functioning as a bidirectional buffer between a data bus and peripheral device logic. In addition to the isolation function, the device also provides a comparison circuit for address selection and a constant generator, useful for interrupt vector addresses. The bus I/O port provides high-impedance inputs and high-drive (70 mA) open-collector outputs to allow direct connection to a computer's data bus. On the peripheral device side, a bidirectional port is also provided, with standard TTL inputs and 20 mA tri-state drivers. Data on this port is the logical inversion of the data on the bus side.

Three address jumper inputs are used to compare against three bus inputs and to generate the signal MATCH. The MATCH output is open-collector, which allows the output of several transceivers to be wire-ANDed to form a composite address match signal. The address jumpers can also be put into a third logical state that disconnects that jumper from the address match, allowing for "don't care" address bits. In addition to the three address jumper inputs, a fourth high-impedance input line is used to enable/disable the MATCH output.

Three vector jumper inputs are used to generate a constant that can be passed to the computer bus. The three inputs directly drive three of the bus lines, overriding the action of the control lines.

Two control signals are decoded to give three operational states: receive data, transmit data, and disable.

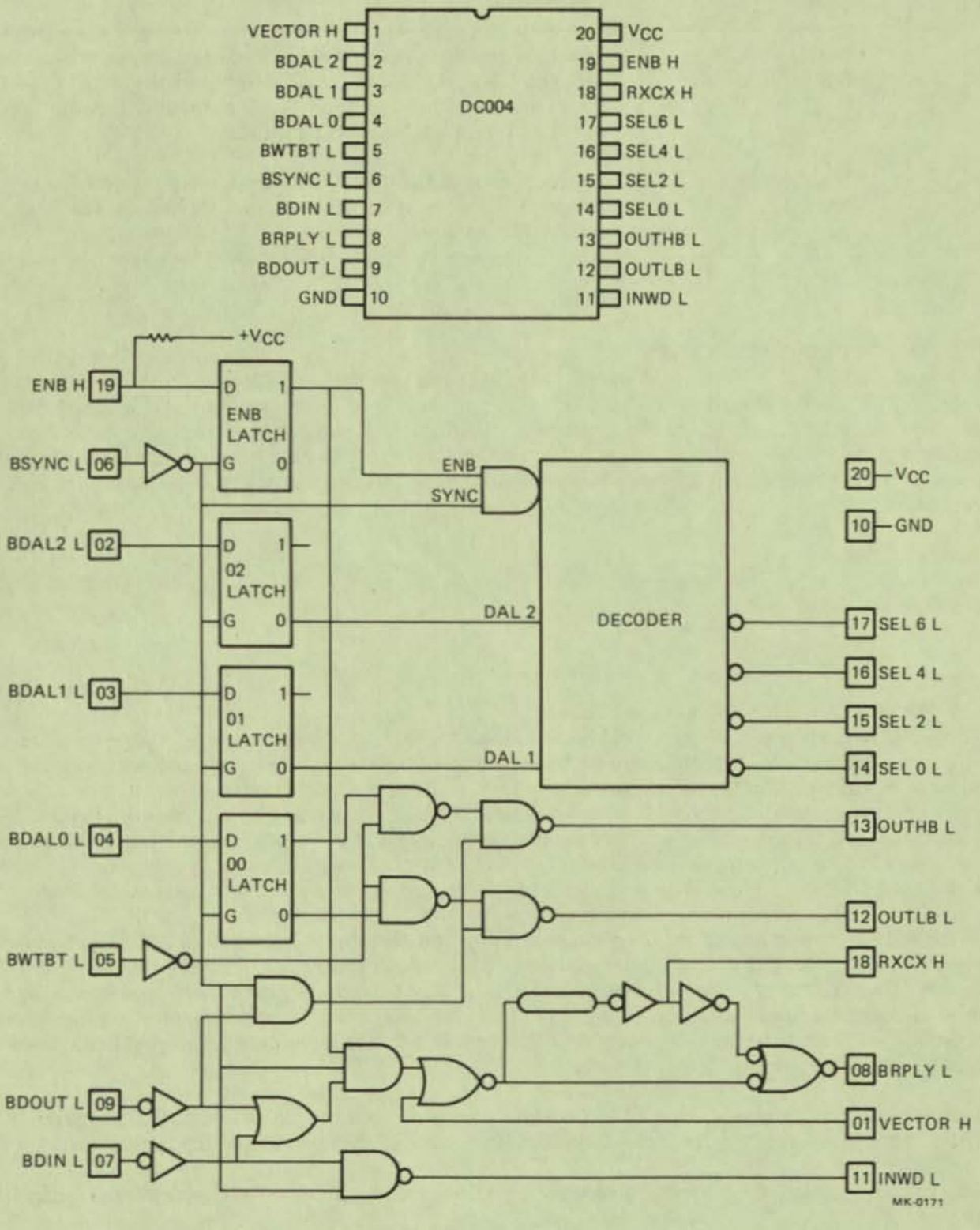


Figure C-2 DC004 Simplified Logic Diagram

Table C-2 DC004 Pin/Signal Descriptions

Pin	Signal	Description
1	VECTOR H	Vector – This input causes BRPLY L to be generated through the delay circuit. Independent of BSYNC L and ENB H.
2 3 4	BDAL2 L BDAL1 L BDAL0 L	Bus Data Address Lines – These signals are latched at the assert edge of BSYNC L. Lines 2 and 1 are decoded for the select outputs; line 0 is used for byte selection.
5	BWTBT L	Bus Write/Byte – While the BDOUT L input is asserted, this signal indicates a byte or word operation: asserted = byte, unasserted = word. Decoded with BDOUT L and latched BDAL0 L, BWTBT L is used to form OUTLB L and OUTHB L.
6	BSYNC L	Bus Synchronize – At the assert edge of this signal, address information is trapped in four latches. While unasserted, this signal disables all outputs except the vector term of BRPLY L.
7	BDIN L	Bus Data In – This is a strobing signal to effect a data input transaction. BDIN L generates BRPLY L through the delay circuit and INWD L.
8	BRPLY L	Bus Reply – This signal is generated through an RC delay by VECTOR H, and strobed by BDIN L or DBOUT L, and BSYNC L and latched ENB H.
9	BDOUT L	Bus Data Out – This is a stobing signal to effect a data output transaction. Decoded with BWTBT L and BDALO, it is used to form OUTLB L and OUTHB L. BDOUT L generates BRPLY L through the delay circuit.
11	INWDL	In Word – Used to gate (read) data from a selected register onto the data bus. It is enabled by BSYNC L and strobed by BDIN L.
12 13	OUTLB L OUTHB L	Out Low Byte, Out High Byte – Used to load (write) data into the lower, higher, or both bytes of a selected register. It is enabled by BSYNC L and the decode of BWTBT L and latched BDAL0 L. It is strobed by BDOUT L.
14 15 16 17	SEL0 L SEL2 L SEL4 L SEL6 L	Select Lines – One of these four signals is true as a function of BDAL2 L and BDAL1 L if ENB H is asserted at the assert edge of BYSNC L. They indicate that a word register has been selected for a data transaction. These signals never become asserted except at the assertion of BSYN L (then only if ENB H is asserted at that time) and, once asserted, are not negated until BSYNC L is negated.
18	RXCX	External Resistor Capacitor Node – This node is provided to vary the delay between the BDIN L, BDOUT L, and VECTOR H inputs and BRPLY L output. The external resistor should be tied to V_{cc} and the capacitor to ground. As an output, it is the logical inversion of BRPLY L.

Table C-2 DC004 Pin/Signal Descriptions (Cont)

Pin	Signal	Description
19	ENB H	Enable – This signal is latched at the asserted edge of BSYNC L and is used to enable the select outputs and the address term of BRPLY L.

Maximum current required from the V_{cc} supply is 100 mA.

Figure C-3 is a simplified logic diagram of the DC005 IC. Signal and pin definitions for the DC005 are shown in Table C-3.

C.5 26LS32 QUAD DIFFERENTIAL LINE RECEIVER

The 26LS32 line receiver is a 16-pin DIP device. Terminal connections are shown in Figure C-4.

C.6 8640 UNIBUS RECEIVER

The 8640 is a quad 2-input NOR. Its equivalent circuit is shown in Figure C-5.

C.7 8881 NAND

The 8881 is a quad 2-input NAND. The schematic and pin identifications are shown in Figure C-6.

C.8 9636A DUAL LINE DRIVER

The 9636A is an 8-pin DIP device specified to satisfy the requirements of EIA standards RS-423-A and RS-232-C. Additionally, it satisfies the requirements of CCITT V.28, V.10 and the federal standard FIPS 1030.

The output slew rates are adjustable by a single external resistor connected from pin 1 to ground.

The logic diagram and terminal identification are shown in Figure C-7.

C.9 9638 DUAL DIFFERENTIAL LINE DRIVER

The 9638 is an 8-pin DIP device specified to satisfy the requirements of EIA RS-422-A and CCITT V.11 specifications.

The logic diagram and terminal identification are shown in Figure C-8.

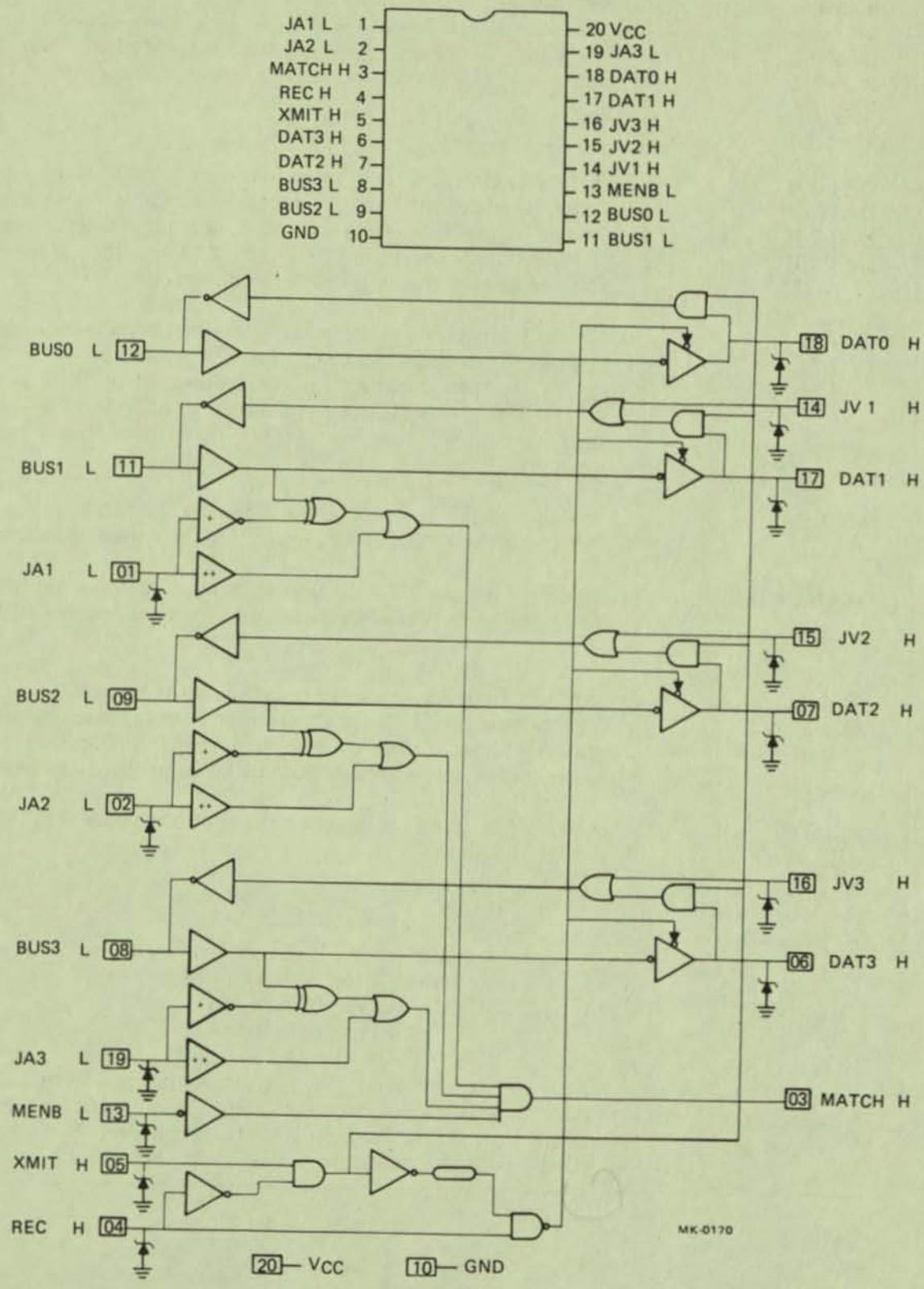
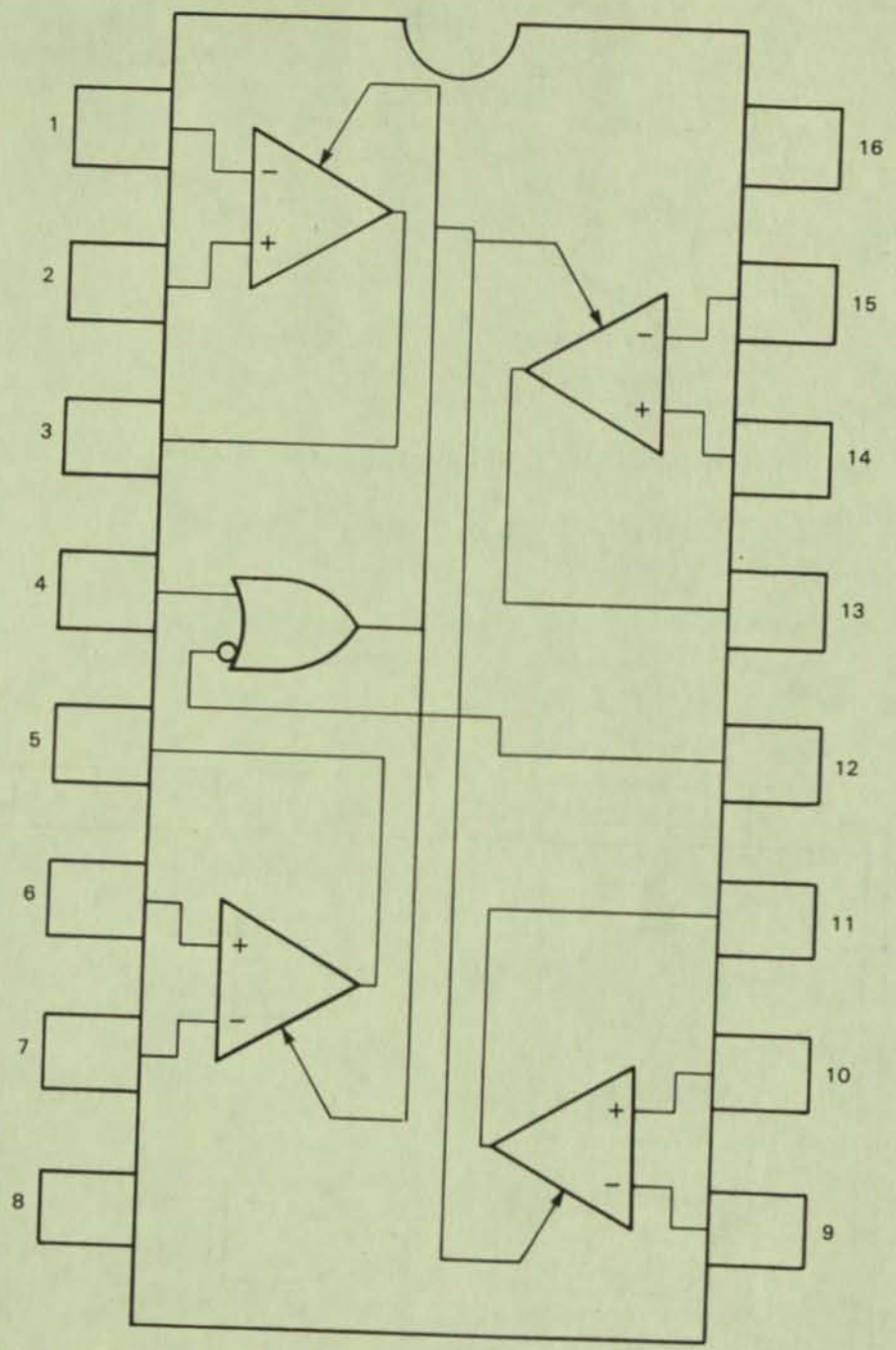


Figure C-3 DC005 Simplified Logic Diagram

Table C-3 DC005 Pin/Signal Descriptions

Pin	Signal	Description
12 11 9 8	BUS 0 L BUS 1 L BUS 2 L BUS 3 L	Bus Data – This set of four lines constitutes the bus side of the transceiver. Open-collector output; high-impedance inputs. Low = 1.
18 17 7 6	DAT0 H DAT1 H DAT2 H DAT3 H	Peripheral Device Data – These four tri-state lines carry the inverted received data from BUS (3:0) when the transceiver is in the receive mode. When in transmit data mode, the data carried on these lines is passed inverted to BUS (3:0). When in the disabled mode, these lines go open (high impedance). High = 1.
14 15 16	JV 1 H JV 2 H JV 3 H	Vector Jumpers – These inputs, with internal pull-down resistors, directly drive BUS (3:1). A low or open on the jumper pin causes an open condition on the corresponding BUS pin if XMIT H is low. A high causes a one (low) to be transmitted on the BUS pin. Note that BUS 0 L is not controlled by any jumpr input.
13	MENB L	Match Enable – A low on this line enables the MATCH output. A high forces MATCH low, overriding the match circuit.
3	MATCH H	Address Match – When BUS (3:1) matches with the state of JA (3:1) and MENB L is low, this output is open; otherwise, it is low.
1 2 19	JA 1 L JA 2 L JA 3 L	Address Jumpers – A strap to ground on these inputs allows a match to occur with a one (low) on the corresponding BUS line; an open allows a match with a zero (high); a strap to V_{cc} disconnects the corresponding address bit from the comparison.
5 4	XMIT H REC H	Control Inputs – These lines control the operational of the transceiver as follows. REC XMIT 0 0 DISABLE: BUS and DAT open 0 1 XMIT DATA: DAT to BUS 1 0 RECEIVE: BUS to DAT 1 1 RECEIVE: BUS to DAT To avoid tri-state overlap conditions, an internal circuit delays the change of modes between Transmit data mode, and delays tri-state drivers on the DAT lines from enabling. This action is independent of the disable mode.



NOTE: PIN 1 IS MARKED FOR ORIENTATION. NUMBERS INDICATED DENOTE TERMINAL NUMBERS.

TERMINAL IDENTIFICATION

1. INPUT A	16. POSITIVE SUPPLY VOLTAGE (V_{CC})
2. INPUT A	15. INPUT B
3. OUTPUT A	14. INPUT B
4. ENABLE	13. OUTPUT B
5. OUTPUT C	12. $\overline{\text{ENABLE}}$
6. INPUT C	11. OUTPUT D
7. INPUT C	10. INPUT D
8. GROUND	9. INPUT D

MK-1340

Figure C-4 26LS32 Terminal Connection Diagram and Terminal Identification

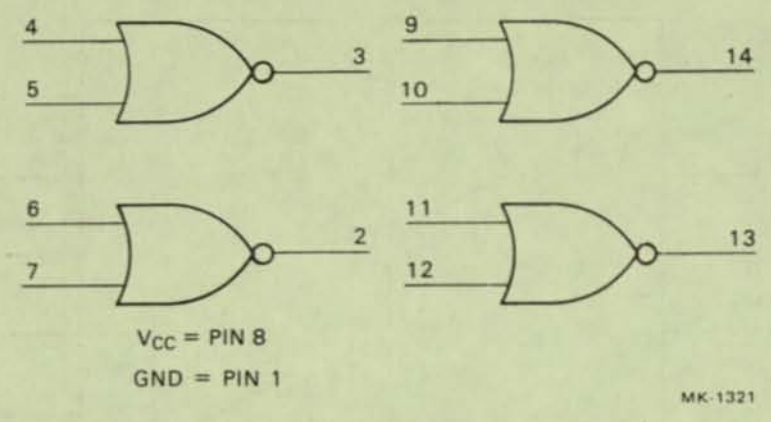


Figure C-5 8640 Equivalent Logic Diagram

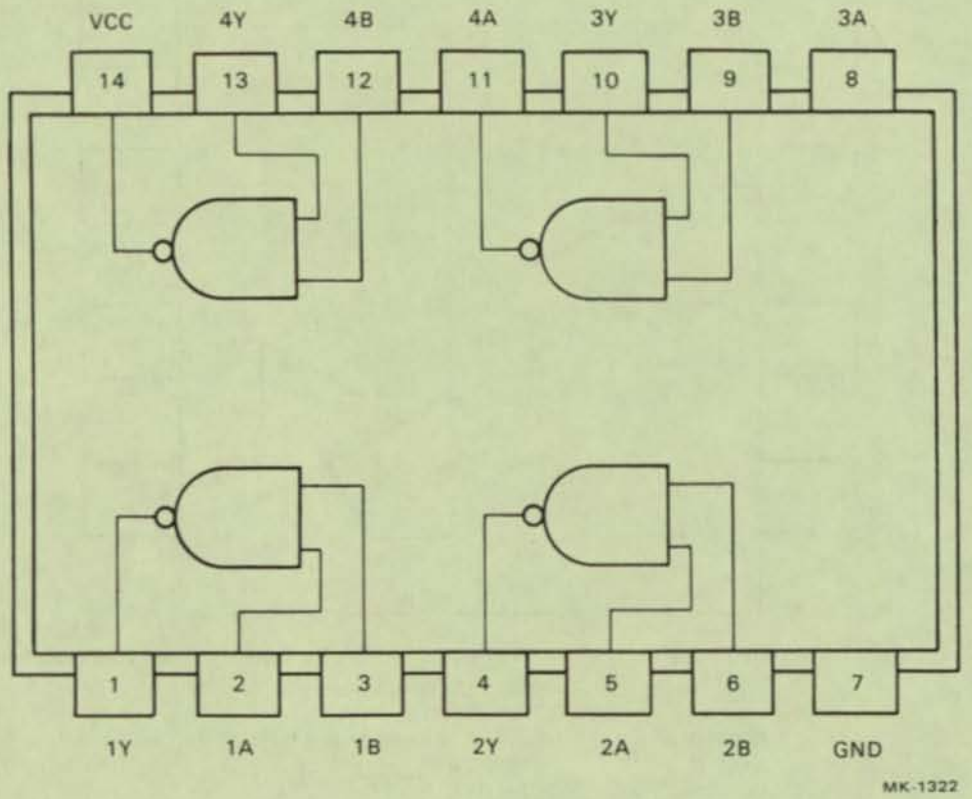
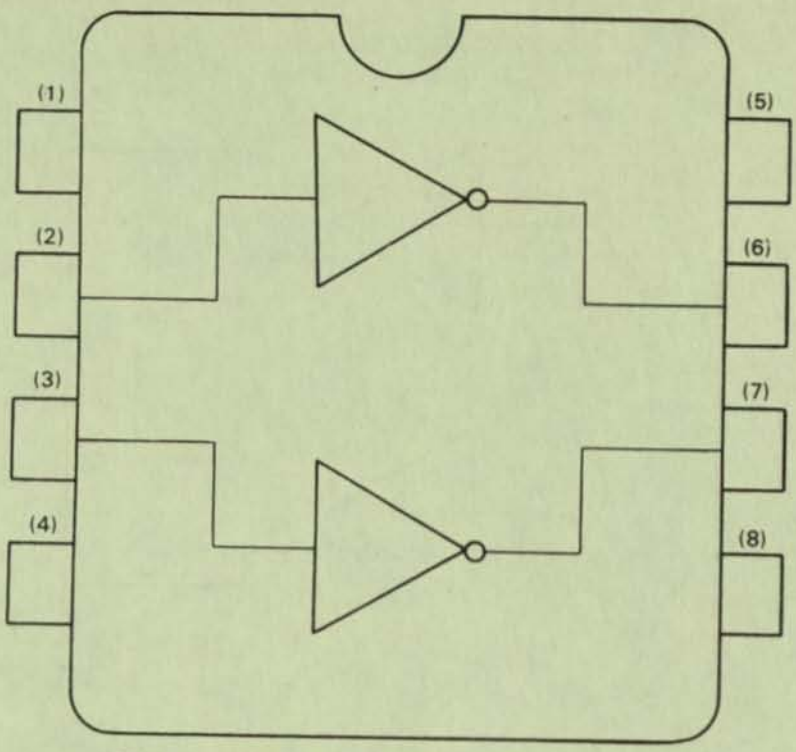


Figure C-6 8881 Pin Identification



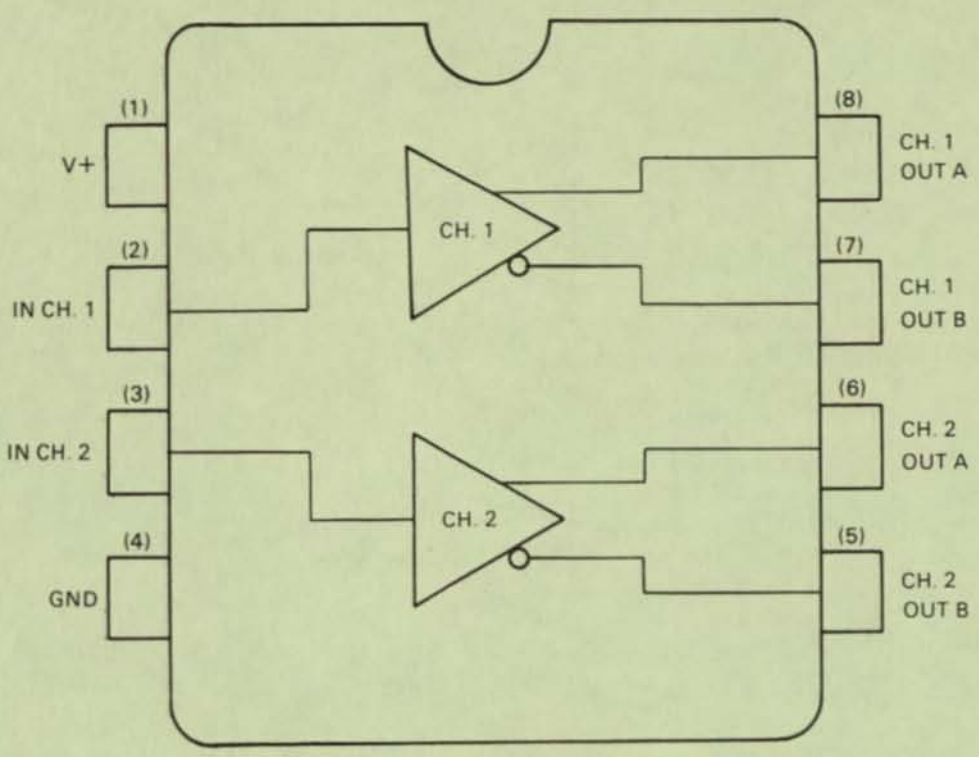
NOTE NUMBERS IN ( ) DENOTE TERMINAL NUMBERS.

TERMINAL IDENTIFICATION

(1) WAVESHAPE CONTROL (RISE AND FALL TIME)
(2) INPUT A
(3) INPUT B
(4) POWER AND SIGNAL GROUND
(5) NEGATIVE SUPPLY VOLTAGE
(6) OUTPUT B
(7) OUTPUT A
(8) POSITIVE SUPPLY VOLTAGE (V_{CC})

MK-1323

Figure C-7 9636A Logic Diagram and Terminal Identification



NOTE: NUMBERS IN ( ) DENOTE TERMINAL NUMBERS.

TERMINAL IDENTIFICATION

1. POSITIVE SUPPLY VOLTAGE
2. CHANNEL 1 INPUT
3. CHANNEL 2 OUTPUT
4. SUPPLY AND SIGNAL GROUND
5. CHANNEL 2 INVERTED OUTPUT
6. CHANNEL 2 NON INVERTED OUTPUT
7. CHANNEL 1 INVERTED OUTPUT
8. CHANNEL 1 NON INVERTED OUTPUT

MK-1324

Figure C-8 9638 Logic Diagram and Terminal Identification

APPENDIX D
PROGRAMMING EXAMPLES

Two examples are included in this appendix. The first is an example for bit-oriented protocols, and the second is an example for byte count-oriented protocols.

These are only examples and are not intended for any other purpose.

```
        .TITLE  DPV11  --  DPV-11 DDM FOR BIT ORIENTED PROTOCOLS
        .IDENT  /X00/
;
; COPYRIGHT (C) 1980 BY
; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
;
;       EXAMPLE OF AN APPLICATION RSX-11M BIT ORIENTED DPV-11 DEVICE DRIVER
;               *** NOTE - THIS IS NOT A RUNNING DRIVER
;
        .MCALL  HWDDF$,$INTSX,$INTXT,MDCDF$,CCBDF$,TMPDF$,ASYRET,SYNRET
        HWDDF$          ; DEFINE THE HARDWARE REGISTERS
        CCBDF$          ; DEFINE THE CCB OFFSETS
        MDCDF$          ; DEFINE THE MODEM CONTROL SYMBOLS
        TMPDF$          ; DEFINE LINE-TABLE TEMPLATE OPERATORS
;
;       DEVICE CHARACTERISTICS DEFINED IN  -D.DCHR-
;
DC.HDX  =  000001       ; HALF-DUPLEX LINE INDICATOR    (WORD #0)
DC.PRT  =  000007       ; PROTOCOL SELECTION FIELD      (WORD #1)
DC.MPT  =  000010       ; MULTI-POINT CONFIGURATION     (WORD #1)
DC.SEC  =  000020       ; MULTI-POINT SECONDARY MODE    (WORD #1)
DC.ADR  =  000040       ; STATION ADDRESS IS 16 BITS    (WORD #1)
DC.SPS  =  000013       ; SDLC PRIMARY STATION          (COMPOSITE)
DC.SSS  =  000033       ; SDLC SECONDARY STATION        (COMPOSITE)
;
;       DEVICE STATUS FLAGS DEFINED IN  -D.FLAG-
;
DD.ENB  ==  001         ; IF ZERO, LINE HAS BEEN ENABLED
DD.STR  ==  002         ; IF ZERO, LINE HAS BEEN STARTED
DD.EOM  ==  CF.EOM      ;  --(UNUSED)--
DD.SOM  ==  CF.SOM      ;  --(UNUSED)--
DD.ABT  ==  020         ; TRANSMIT ABORTED DUE TO UNDERRUN
DD.SYN  ==  CF.SYN      ; TRANSMIT SYNC-TRAIN REQUIRED
DD.TRN  ==  CF.TRN      ; TRANSMIT LINE TURN-AROUND REQUIRED
DD.ACT  ==  200         ; TRANSMITTER READY FOR NEXT FRAME
DD.DIS  ==  DD.ENB!DD.STR ; INITIAL STATUS = DISABLED, STOPPED
;
;       [ SEL 0 ]  --  MODEM CONTROL BITS
;
DSCHG   =  100000       ; DATA SET CHANGE
DSRING  =  040000       ; RING INDICATOR
DSCTS   =  020000       ; CLEAR TO SEND
DSCARY  =  010000       ; CARRIER INDICATOR
DSMODR  =  001000       ; MODEM READY
DSITEN  =  000040       ; DATA SET INTERRUPT ENABLE
DSLOOP  =  000010       ; DATA SET LOOPBACK
DSRTS   =  000004       ; REQUEST TO SEND
DSDTR   =  000002       ; DATA TERMINAL READY
DSSEL   =  000001       ; SELECT FREQUENCY OR REMOTE LOOPBACK
;
;       [ SEL 0 ]  --  RECEIVER CONTROL BITS
;
RXACT   =  004000       ; RECEIVER ACTIVE
RXSRDY  =  002000       ; RECEIVER STATUS READY
RXFLAG  =  000400       ; RECEIVER FLAG DETECT
RXDONE  =  000200       ; RECEIVER DONE
RXITEN  =  000100       ; RECEIVER INTERRUPT ENABLE
RXREN   =  000020       ; RECEIVER ENABLE
;
;       [ SEL 2 ]  --  RECEIVER STATUS INPUTS
;
RXERR   =  100000       ; RECEIVER CRC ERROR
RXABC   =  070000       ; RECEIVER ASSEMBLED BIT COUNT
RXBFOV  =  010000       ; RECEIVER BUFFER OVERFLOW (SOFTWARE ERROR)
RXOVRN  =  004000       ; RECEIVER DATA OVERRUN
```

```
RXABRT  =  002000        ; RECEIVED ABORT
RXENDM  =  001000        ; RECEIVED END OF MESSAGE
RXSTRM  =  000400        ; RECEIVED START OF MESSAGE
;
;       [ SEL 2 ]  --  MODE CONTROL OUTPUTS
;
DPAPA   =  100000        ; ALL PARTIES ADDRESSED
DPDECM  =  040000        ; DDCMP / BISYNC OPERATION
DPSTRP  =  020000        ; STRIP SYNC OR LOOP MODE
DPSECS  =  010000        ; SDLC / ADCCP SECONDARY STATION SELECT
DPIDLE  =  004000        ; IDLE MODE SELECT
DPCRC   =  3*400         ; USE CRC 16 ERROR DETECTION
DPADRC  =  000377        ; STATION ADDRESS OR SYNC CHARACTER
INPRM   =  DPSTRP!DPCRC  ; INITIAL STARTUP PARAMETERS
;
;       [ SEL 4 ]  --  TRANSMITTER STATUS AND CONTROL
;
TCLEN   =  160000        ; TRANSMIT CHARACTER LENGTH
EXADD   =  010000        ; EXTENDED ADDRESS FIELD
EXCON   =  004000        ; EXTENDED CONTROL FIELD
RCLEN   =  003400        ; RECEIVE CHARACTER LENGTH
TXITEN  =  000100        ; TRANSMITTER INTERRUPT ENABLE
TXREN   =  000020        ; TRANSMITTER ENABLE
TXMAI   =  000010        ; MAINTENANCE MODE SELECT
TXDONE  =  000004        ; TRANSMITTER DONE
TXACT   =  000002        ; TRANSMITTER ACTIVE
TXRES   =  000001        ; DEVICE RESET
;
;       [ SEL 6 ]  --  TRANSMITTER OUTPUT CONTROLS
;
TXLATE  =  100000        ; TRANSMITTER DATA LATE (UNDERRUN)
TXGO    =  004000        ; TRANSMITTER GO AHEAD
TXABRT  =  002000        ; TRANSMITTER ABORT
TXENDM  =  001000        ; TRANSMIT END OF MESSAGE
TXSTRM  =  000400        ; TRANSMIT START OF MESSAGE
;
; PROCESS DISPATCH TABLE
;
$DXPTB::
        .WORD   $SDASX          ; TRANSMIT ENABLE
        .WORD   $SDASR          ; RECEIVE  ENABLE (ASSIGN BUFFER)
        .WORD   $SDKIL          ; KILL I/O ENABLE
        .WORD   $SDCTL          ; CONTROL  ENABLE
        .WORD   $SDTIM          ; TIME OUT

        .SBTTL  $SDPRI  --  RECEIVE INTERRUPT SERVICE ROUTINE
;+
; FUNCTION:
;
;       THE DEVICE INTERRUPT IS VECTORED BY THE HARDWARE TO THE
;       DEVICE LINE TABLE. THE '$SDPRI' LABEL IS ENTERED VIA A
;       CALLING SEQUENCE IN THE LINE TABLE AT OFFSET 'D.RXIN'.
;
; ON ENTRY:
;
;         R5  = ADDRESS OF 'D.RDBF' IN THE LINE TABLE
;       0(SP) = SAVED R5
;       2(SP) = INTERRUPTED PC
;       4(SP) = INTERRUPTED PS
;
; OUTPUTS:
;
;       R5 = ADDRESS OF 'D.RDB2' IN THE LINE TABLE
;       D.RVAD = RECEIVER STATUS BITS FROM CSR [SEL 2]
;-
```

```
$SDPRI::
        MOV     R3,-(SP)        ;;; SAVE REGISTERS
        MOV     R4,-(SP)        ;;; ...
        MOV     @(R5)+,R4       ;;; GET CHARACTER AND FLAGS
        BIC     #RXABC,R4       ;;; DON'T WORRY ABOUT ASSEMBLED BIT COUNT
    .IF DF M$$MGE
        MOV     KISAR6,-(SP)    ;;; SAVE CURRENT MAP
        MOV     (R5)+,KISAR6    ;;; MAP TO DATA BUFFER
    .IFTF
        DEC     (R5)+           ;;; DECREMENT BUFFER BYTE COUNT
        BMI     DPRBO           ;;; BUFFER OVERFLOW - POST COMPLETE

        MOV     2(R5),R3        ;;; GET CSR+2 ADDRESS
        BIT     #RXSRDY,-(R3)   ;;; ERROR OR END-OF-MESSAGE ?
        BNE     DPRCP           ;;; YES - POST RECEIVE COMPLETE

        MOVB    R4,@(R5)+       ;;; STORE CHARACTER IN RECEIVE BUFFER
    .IFT
        MOV     (SP)+,KISAR6    ;;; RESTORE PREVIOUS MAPPING
    .IFTF
        INC     -(R5)           ;;; ADVANCE BUFFER ADDRESS
        MOV     (SP)+,R4        ;;; RESTORE REGISTERS
        MOV     (SP)+,R3        ;;; ...
        $INTXT                  ;;; EXIT THE INTERRUPT


DPRBO:                          ;;; BUFFER OVERRUN HAS OCCURRED
        BIS     #RXBFOV,R4      ;;; SET (SOFTWARE) ERROR INDICATOR

DPRCP:                          ;;; END-OF-MESSAGE OR ERROR INDICATION
    .IFT
        MOV     (SP)+,KISAR6    ;;; RESTORE PREVIOUS MAPPING
    .ENDC
        MOV     R4,(R5)+        ;;; SAVE STATUS FLAGS IN 'D.RVAD'
        MOV     (R5)+,R4        ;;; GET CSR+2 ADDR + POINT TO 'D.RPRI'
        BIC     #RXITEN,-(R4)   ;;; CLEAR RECEIVER INTERRUPT ENABLE
        MOV     (SP)+,R4        ;;; RESTORE R4 SO '$INTSV' IS HAPPY
        MOV     (SP)+,R3        ;;; AND R3
        $INTSX                  ;;; DO A TRICKY $INTSV (R5 SAVED BUT NOT R4)
;
;       CHECK FOR ERRORS, POST RECEIVE COMPLETE, ASSIGN NEW BUFFER
;
        MOV     R3,-(SP)                ;; SAVE AN ADDITIONAL REGISTER
        MOV     (R5),R4                 ;; CCB ADDRESS TO R4 (R5 POPPED)
        ADD     #D.RCNT-D.RCCB,R5       ;; BACK UP TO THE RESI DUAL COUNT
        SUB     (R5)+,C.CNT1(R4)        ;; COMPUTE RECEIVED FRAME BYTE COUNT
        CLR     R3                      ;; SET R3 FOR COMPLETION STATUS

        BIC     #61777,(R5)+    ;; ANY ERRORS REPORTED ?
        BEQ     40$             ;; NO -- POST RECEIVE COMPLETE O.K.
        ASR     -(R5)           ;; SHIFT ERROR INDICATORS...
        ASR     (R5)+           ;; ...TWO PLACES RIGHT
        ASRB    -(R5)           ;; SHIFT 'RXABRT' INTO C-BIT
        MOVB    (R5)+,R3        ;; USE INDICATORS AS TABLE INDEX
        MOV     RCVERR-2(R3),R3 ;; R3 NOW = CCB STATUS FLAGS
        BCC     40$             ;; FRAME NOT ABORTED - POST COMPLETE
        INC     D.RABT-D.RDB2(R5) ;COUNT NUMBER OF ABORTED FRAMES
        CALL    RBFUSE          ;; RE-INITIALIZE WITH THE SAME BUFFER
        BR      60$             ;; RE-ENABLE INTERRUPTS FOR NEXT FRAME

40$:    BIS     C.STS(R4),R3    ;; INCLUDE RE-SYNC STATUS, IF ANY
        MOV     R3,-(SP)        ;; SAVE STATUS REPORTED TO DLC
        CALL    $DDRCP          ;; POST RECEIVE COMPLETE
        MOV     (SP)+,R3        ;; RECOVER COMPLETION STATUS
        CALL    RBFSET          ;; ASSIGN NEW CCB TO THE RECEIVER
```

```
        BCS     DREXIT          ;; FAILED - LEAVE RECEIVER INACTIVE
        TST     R3              ;; WAS AN ERROR REPORTED TO DLC ?
        BMI     DRCLRA          ;; YES - DISABLE RCVR FOR RE-SYNC

60$:    MOV     -(R5),R3        ;; RECEIVER CSR [SEL 2] TO R3
        BIS     #RXITEN,-(R3)   ;; RE-ENABLE RECEIVER INTERRUPTS
DREXIT:
        MOV     (SP)+,R3        ;; RESTORE REGISTER R3
        RETURN                  ;; EXIT TO THE SYSTEM

;+
; DRCLRA:
;
;       MOMENTARILY RESET 'RXREN' FLAG IN ORDER TO FORCE RECEIVER
;       RE-SYNCHRONIZATION.  THIS IS REQUIRED FOR ANY ERROR WHICH
;       TERMINATES THE RECEIVE OPERATION IN MID-FRAME.
;
; ON ENTRY:
;
;       R5 = ADDRESS OF 'D.RCCB' IN THE LINE TABLE
;       R4 = ADDRESS OF 'C.STS' IN THE NEWLY-ASSIGNED CCB
;      (SP)= SAVED R3 VALUE
;-
DRCLRA:
        MOV     -(R5),R3                ;; RCVR CSR ADDRESS [SEL 2] TO R3
        BIC     #RXREN,-(R3)            ;; RESET RCVR ENABLE FOR RE-SYNC
        BIS     #CS.RSN,(R4)            ;; SET RE-SYNC IN CCB 'C.STS'
        BIS     #RXREN!RXITEN,(R3)      ;; RE-ENABLE THE RECEIVER
        BR      DREXIT                  ;; RESTORE R3 AND EXIT

        .SBTTL  $SDPTI  --  TRANSMIT INTERRUPT SERVICE ROUTINE
;+
; FUNCTION:
;
;       THE DEVICE INTERRUPT IS VECTORED BY THE HARDWARE TO THE
;       DEVICE LINE TABLE. THE '$SDPTI' LABEL IS ENTERED VIA A
;       CALLING SEQUENCE IN THE LINE TABLE AT OFFSET 'D.TXIN'.
;       ONCE FRAME TRANSMISSION IS INITIATED, EACH INTERRUPT IS
;       HANDLED BY THE ROUTINE ADDRESSED VIA THE 'D.TSPA' WORD.
;
; ON ENTRY:
;
;         R5  = ADDRESS OF 'D.TCSR' IN THE LINE TABLE
;       0(SP) = SAVED R5
;       2(SP) = INTERRUPTED PC
;       4(SP) = INTERRUPTED PS
;
; ON EXIT:
;
;       R5 = ADDRESS OF 'D.TCCB' IN THE LINE TABLE
;-

$SDPTI::
        MOV     R4,-(SP)        ;;; SAVE R4
        MOV     (R5)+,R4        ;;; GET TRANSMITTER CSR ADDRESS
        TST     (R4)+           ;;; POINT TO [SEL 6] + TEST UNDERRUN
        JMP     @(R5)+          ;;; GO TO CORRECT STATE PROCESSOR
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
;       CURRENT STATE =         MONITOR CSR FOR 'CLEAR TO SEND' ;
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
TISCTS:
        BIT     #DSCTS,-6(R4)               ;;; IS 'CLEAR TO SEND' ACTIVE YET ?
        BNE     TISIFL                      ;;; YES - START TO SEND THE FRAME
        BITB    #DD.SYN,D.FLAG-D.TCNT(R5)   ;;; SYNC-TRAIN REQUIRED ?
        BEQ     TISIFX                      ;;; NO -- SEND FLAGS UNTIL 'CTS'
```

```
        MOV     #TXSTRM!TXENDM,(R4)       ;;; START + END SENDS SYNC STRING
        BR      TISEXT
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
;       CURRENT STATE =         SEND INITIAL FRAME 'FLAG'       ;
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
TISIFL:
        MOV     #TISTRT,-(R5)   ;;; NEXT STATE = SEND ADDRESS BYTE
TISIFX:
        MOV     #TXSTRM,(R4)    ;;; SEND AN SDLC FLAG CHARACTER
        BR      TISEXT

;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
;       CURRENT STATE =         SEND ADDR BYTE FOLLOWING 'FLAG' ;
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
TISTRT:
        DEC     (R5)                      ;;; DECREMENT COUNT FOR ADDR BYTE
        MOV     D.TADC-D.TCNT(R5),(R4)    ;;; SEND ADDR, CLEAR 'TXSTRM'
        MOV     #TISDAT,-(R5)             ;;; NEXT STATE = DATA TRANSFER
        BR      TISEXT
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
;       CURRENT STATE =         TRANSFER FRAME DATA BYTES       ;
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
TISDAT:
        BMI     TISLAT            ;;; UNDERRUN - ABORT AND RE-TRANSMIT
        DEC     (R5)+             ;;; DECREMENT DATA BYTE COUNT
        BMI     TISEND            ;;; ALL DONE - SEND END-MSG SEQUENCE
    .IF DF M$$MGE
        MOV     KISAR6,-(SP)      ;;; SAVE CURRENT MAPPING
        MOV     (R5)+,KISAR6      ;;; MAP TO THE TRANSMIT BUFFER
    .IFTF
        INC     (R5)              ;;; ADVANCE THE BUFFER ADDRESS
        MOVB    @(R5)+,(R4)       ;;; NEXT CHARACTER TO BE SENT
    .IFT
        MOV     (SP)+,KISAR6      ;;; RESTORE PREVIOUS MAPPING
    .ENDC

TISEXT:                           ;;; COMMON LEVEL-7 INTERRUPT EXIT
        MOV     (SP)+,R4          ;;; RESTORE R4
        $INTXT                    ;;; EXIT INTERRUPT SERVICE
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
;       CURRENT STATE =         DATA BYTE-COUNT EXHAUSTED       ;
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
TISEND:
        MOV     #TXENDM,(R4)              ;;; TRANSMIT END-OF-MSG SEQUENCE
        INC     -(R5)                     ;;; ADJUST R5 AND CLEAR 'D.TCNT'
        MOV     #TISFLG,-(R5)             ;;; NEXT STATE = IDLE FLAGS (ASSUMED)
        ASLB    D.FLAG-D.TSPA(R5)         ;;; TEST FOR LINE TURN-AROUND
        BPL     TISEXT                    ;;; NO -- IDLE THE LINE WITH FLAGS
        MOV     #TISPAD,(R5)              ;;; YES - SEND PADS, THEN DISABLE
        BR      TISEXT
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
;       CURRENT STATE =         SEND 'ABORT' AS PAD AFTER 'FLAG';
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
TISPAD:
        CLRB    D.FLAG-D.TCNT(R5)         ;;; RESET THE DEVICE FLAG BYTE
        MOV     #TISCLR,-(R5)             ;;; NEXT STATE = SEND SECOND PAD
        MOV     #TXABRT,(R4)              ;;; SET 'TXABRT' TO SEND A PAD
        BR      TISEXT
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
;       CURRENT STATE =         SEND SECOND 'ABORT' AS PAD      ;
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
TISCLR:
        MOV     #TISRTS,-(R5)   ;;; NEXT STATE = DROP 'REQUEST TO SEND'
```

```
TISCLX:
        MOV     #TXABRT,(R4)    ;;; SETUP TO SEND ANOTHER 'ABORT' CHAR
        BIC     #TXREN,-(R4)    ;;; DISABLE THE TRANSMITTER
        BR      TISEXT

;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
;       CURRENT STATE =         DROP REQUEST TO SEND + EXIT      ;
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
TISRTS:
        BIT     #DC.HDX,D.DCHR-D.TCNT(R5) ;;; HALF-DUPLEX CHANNEL ?
        BEQ     TISDON                    ;;; NO -- LEAVE 'RTS' ACTIVE
        BIC     #DSRTS,-6(R4)             ;;; DROP 'REQUEST TO SEND' LINE
        BR      TISDON                    ;;; POST TRANSMIT COMPLETE
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
;       CURRENT STATE =         TRANSMITTER DATA UNDERRUN        ;
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
TISLAT:
        MOV     #TISDON,-(R5)             ;;; NEXT STATE = RE-TRANSMIT
        MOVB    #DD.ABT,D.FLAG-D.TSPA(R5) ;;; THIS FRAME WAS ABORTED
        INC     D.TURN-D.TSPA(R5)         ;;; COUNT THE ERROR EVENTS
        BR      TISCLX                    ;;; SEND PAD, DISABLE TRANSMITTER
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
;       CURRENT STATE =         IDLE FLAGS BETWEEN FRAMES        ;
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
TISFLG:
        MOV     #TXSTRM,(R4)              ;;; CLEAR 'TXENDM', IDLE FLAGS
        MOVB    #DD.ACT,D.FLAG-D.TCNT(R5) ;;; TRANSMITTER IS ACTIVE
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
;       CURRENT STATE =         POST COMPLETE OR RE-TRANSMIT     ;
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
TISDON:
        ADD     #D.TPRI-D.TCNT,R5 ;;; ADJUST LINE TABLE POINTER
        BIC     #TXITEN,-(R4)     ;;; DISABLE 'TXDONE' INTERRUPTS
        MOV     (SP)+,R4          ;;; RESTORE R4 FOR PRIORITY DROP
        $INTSX                    ;;; '$INTSV' W/O R4 SAVED (POPS R5)

        MOV     R3,-(SP)                    ;; SAVE AN ADDITIONAL REGISTER
        MOV     (R5),R4                     ;; ACTIVE CCB ADDRESS TO R4
        CLR     (R5)+                       ;; THIS CCB IS NO LONGER ACTIVE
        BITB    #DD.ABT,D.FLAG-D.TCBQ(R5)   ;; WAS THE FRAME ABORTED ?
        BNE     TRSTRT                      ;; YES - SETUP RE-TRANSMISSION
        TST     D.KCCB-D.TCBQ(R5)           ;; TRANSMIT KILL IN PROGRESS ?
        BNE     CKILLT                      ;; YES - RETURN CCB'S TO THE DLC
        CLR     R3              ;; SET COMPLETION STATUS = SUCCESS
        CALL    $DDXMP          ;; POST TRANSMIT COMPLETE TO THE DLC
        MOV     (R5),R4         ;; FIRST CCB ON SECONDARY CHAIN
        BEQ     TREXIT          ;; NONE THERE - TRANSMITTER IDLE
        MOV     (R4),(R5)       ;; REMOVE CCB FROM SECONDARY CHAIN

;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
;       CURRENT STATE =         START UP FRAME TRANSMISSION      ;
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
TRSTRT:
        CLR     (R4)                        ;; CLEAR CCB LINKAGE WORD
        MOV     R4,-(R5)                    ;; SETUP AS THE ACTIVE CCB
        TST     -(R5)                       ;; SKIP BACK OVER 'D.TPRI'
        ADD     #C.FLG1,R4                  ;; POINT TO THE CCB BUFFER FLAGS
        BISB    (R4),D.FLAG-D.TPRI(R5)      ;; SAVE FLAGS FOR LEVEL-7 USE
        BICB    #DD.ABT,D.FLAG-D.TPRI(R5) ;MAKE SURE 'ABORT' FLAG IS OFF

        MOV     -(R4),D.TCNT-D.TPRI(R5) ;; SET TRANSMIT BYTE COUNT
        CLR     -(R5)                   ;; INITIALIZE 'D.TADC' WORD
        MOV     -(R4),-(R5)             ;; SET TRANSMIT BUFFER ADDRESS
```

```
    .IF DF M$$MGE
        MOV     -(R4),-(R5)             ;; SET TRANSMIT BUFFER RELOCATION
        MOV     KISAR6,-(SP)            ;; SAVE THE CURRENT APR6 MAPPING
        MOV     (R5)+,KISAR6            ;; MAP TO THE TRANSMIT BUFFER
    .IFTF
        MOVB    @(R5)+,(R5)             ;; MOVE ADDRESS BYTE TO 'D.TADC'
    .IFT
        MOV     (SP)+,KISAR6            ;; RESTORE PREVIOUS APR6 MAPPING
    .ENDC

        ADD     #D.TSPA-D.TADC,R5       ;; BACK UP TO STATE PROCESSOR CELL
        TSTB    D.FLAG-D.TSPA(R5)       ;; IS THE TRANSMITTER READY NOW ?
        BPL     20$                     ;; NO -- ENABLE IT, THEN START

        MOV     #TISTRT,(R5)            ;; INITIAL STATE = SEND ADDR BYTE
        BR      40$                     ;; ENABLE INTERRUPTS AND EXIT

20$:    MOV     -2(R5),R3               ;; TRANSMITTER CSR [SEL 4] TO R3
        BIS     #DSRTS,-4(R3)           ;; ASSERT 'REQUEST TO SEND'
        BIS     #TXREN,(R3)+            ;; ENABLE THE TRANSMITTER

        MOV     #TISCTS,(R5)            ;; INITIAL STATE = WAIT FOR 'CTS'

40$:    BIS     #TXITEN,@-(R5)          ;; RE-ENABLE TRANSMIT INTERRUPTS

TREXIT:
        MOV     (SP)+,R3                ;; RESTORE R3 FROM ENTRY
        ASYRET                          ;; EXIT WHEREVER APPROPRIATE, ASYNC

;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
;       CURRENT STATE =         TRANSMIT KILL OR TIMEOUT        ;
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
CKILLT:
        MOV     #CS.ERR!CS.ABO,-(SP) ;; TRANSMIT COMPLETION STATUS

CKTTMO:
        BIC     #TXREN,@D.TCSR-D.TCBQ(R5) ;; DISABLE TRANSMITTER
        MOV     (R5),(R4)           ;; ADD SECONDARY CHAIN TO PRIMARY
        CLR     (R5)+               ;; CLEAR SECONDARY CHAIN POINTER

20$:    MOV     (SP),R3             ;; COMPLETION STATUS TO R3
        MOV     (R4),-(SP)          ;; NEXT CCB ADDRESS TO STACK
        CLR     (R4)                ;; MAKE SURE LINK WORD IS ZERO
        CALL    $DDXMP              ;; POST A CCB COMPLETE W/ERROR
        MOV     (SP)+,R4            ;; NEXT CCB ADDRESS TO R4
        BNE     20$                 ;; MORE TO GO - CONTINUE
        TST     (SP)+               ;; CLEAN STATUS OFF THE STACK

        MOV     (R5),R4             ;; KILL CCB ADDRESS TO R4
        BEQ     TREXIT              ;; NONE - RESTORE R3 AND EXIT
        CLR     (R5)                ;; KILL NO LONGER IN PROGRESS
        CLR     R3                  ;; STATUS = SUCCESSFUL

        CMPB    #FC.KIL,C.FNC(R4)   ;; KILL-I/O OR CONTROL FUNCTION ?
        BNE     40$                 ;; CONTROL - POST IT COMPLETE
        CALL    $DDKCP              ;; POST KILL-I/O COMPLETE
        BR      TREXIT              ;; RESTORE R3 AND EXIT

40$:    CALL    $DDCCP              ;; POST CONTROL COMPLETE
        BR      TREXIT              ;; RESTORE R3 AND EXIT

        .SBTTL  $SDASX  --  TRANSMIT ENABLE ENTRY
;+
; FUNCTION:
;
;       '$SDASX' IS ENTERED (VIA THE DISPATCH TABLE) TO QUEUE A
```

```
;       CCB CONTAINING AN SDLC FRAME TO BE TRANSMITTED.  IF THE
;       TRANSMITTER IS BUSY, THE CCB IS QUEUED TO THE SECONDARY
;       CCB CHAIN.  IF NOT, THE TRANSMITTER IS ENABLED TO START
;       TRANSMITTING THE NEW FRAME.
;
; ON ENTRY:
;
;       R4 = ADDRESS OF TRANSMIT ENABLE CCB
;       R5 = ADDRESS OF DEVICE LINE TABLE
;       PS = PRIORITY OF CALLING DLC PROCESS
;
; ON EXIT:
;
;       ALL REGISTERS ARE UNPREDICTABLE
;-

$SDASX::
        MOV     R3,-(SP)        ;; SAVE R3 FOR EXIT VIA 'TRSTRT'
        MOV     D.TCSR(R5),R3   ;; TRANSMIT CSR ADDRESS [SEL 4] TO R3
        BIC     #TXITEN,(R3)    ;; DISABLE TRANSMITTER INTERRUPTS
        ADD     #D.TCCB,R5      ;; POINT TO ACTIVE CCB ADDRESS CELL

        TST     (R5)+           ;; IS THERE AN ACTIVE CCB ?
        BEQ     TRSTRT          ;; NO -- START UP THE TRANSMITTER
        MOV     R4,-(SP)        ;; SAVE POINTER TO FIRST CCB

20$:    MOV     R5,R4           ;; COPY THE CCB ADDRESS TO R4
        MOV     (R4),R5         ;; ADDRESS OF THE NEXT CCB TO R5
        BNE     20$             ;; LOOP UNTIL WE FIND THE END

        MOV     (SP)+,(R4)      ;; LINK NEW CCB TO END OF CHAIN
        CLR     @(R4)+          ;; MARK NEW END OF CCB CHAIN
        BIS     #TXITEN,(R3)    ;; RE-ENABLE TRANSMITTER INTERRUPTS
        BR      TREXIT          ;; RESTORE R3 AND EXIT

        .SBTTL  $SDASR  --  RECEIVE ENABLE AFTER BUFFER WAIT
;+
; FUNCTION:
;
;       THIS ROUTINE IS CALLED BY THE BUFFER POOL MANAGER WHEN
;       A BUFFER ALLOCATION REQUEST CAN BE SATISFIED, FOLLOWING
;       AN ALLOCATION FAILURE AND A CALL TO '$RDBWT'.
;
; ON ENTRY:
;
;       R4 = ADDRESS OF CCB AND RECEIVE BUFFER
;       R5 = ADDRESS OF DEVICE LINE TABLE
;
; ON EXIT:
;
;       R5 = ADDRESS OF 'D.RCCB' IN THE LINE TABLE
;       R4 = ADDRESS OF 'C.STS' IN THE CCB
;      (SP)= SAVED VALUE OF R3
;-

$SDASR::
        ADD     #D.RDB2,R5      ;; POINT TO SECOND RCVR-CSR WORD
        CALL    RBFUSE          ;; ASSIGN BUFFER TO THE RECEIVER
        BIS     #CS.BUF,(R4)    ;; PREV. ALLOC. FAILURE TO CCB 'C.STS'

        MOV     R3,-(SP)        ;; PUSH R3 FOR EXIT AT 'DREXIT', ABOVE
        JMP     DRCLRA          ;; RESET AND ACTIVATE THE RECEIVER

;+
; $SDSTR  --  START UP DEVICE AND LINE ACTIVITY
;-
```

```
$SDSTR::
        BITB    #DD.ENB,D.FLAG(R5)      ;; HAS THE LINE BEEN ENABLED ?
        BNE     60$                     ;; NO -- REJECT THE 'START'

        MOV     D.RDBF(R5),R3   ;; RECEIVER CSR ADDR [SEL 2] TO R3
        MOV     D.STN(R5),(R3)  ;; SET ADDRESS BYTE + OPERATING MODE
        BIS     #RXREN,-(R3)    ;; ENABLE THE RECEIVER

        MOV     R5,-(SP)        ;; SAVE LINE TABLE START ADDRESS
        ADD     #D.RDB2,R5      ;; ADJUST R5 FOR BUFFER ROUTINE
        CALL    RBFSET          ;; ASSIGN A RECEIVE CCB AND BUFFER
        BCS     20$             ;; FAILED - START THE TRANSMITTER
        BIS     #RXITEN,(R3)    ;; ENABLE RECEIVER INTERRUPTS

20$:    MOV     (SP)+,R5                ;; RECOVER LINE TABLE START
        CLRB    D.FLAG(R5)              ;; LINE HAS BEEN STARTED
        BIT     #DC.HDX,D.DCHR(R5)      ;; CHECK THAT ASSUMPTION
        BNE     CTLCMP                  ;; CORRECT - STARTUP COMPLETE
        BIS     #DSRTS,(R3)             ;; ASSERT 'REQUEST TO SEND' LINE
        BR      CTLCMP                  ;; ...AND POST START COMPLETE
60$:    MOV     #CS.ERR!CS.DIS,R3       ;; STATUS = LINE DISABLED
        BR      CTLERR                  ;; RETURN ERROR W/COMPLETION


DP.NOP:                         ;; CONTROL FUNCTION = NO-OPERATION
CTLCMP:
        CLR     R3              ;; STATUS = SUCCESSFUL
CTLERR:
        MOV     (SP)+,R4        ;; RECOVER SAVED R4 VALUE
        SYNRET                  ;; SYNCHRONOUS RETURN

        .SBTTL  $SDSTP  --  STOP DEVICE AND LINE ACTIVITY
;-----------------------------------------------------------;
;       'S T O P'   C O N T R O L   F U N C T I O N         ;
;-----------------------------------------------------------;
$SDSTP::
        MOV     D.RDBF(R5),R3   ;; RECEIVER CSR ADDR [SEL 2] TO R3
        MOV     #DSDTR,-(R3)    ;; DISABLE RECEIVER, LEAVE 'DSDTR' ACTIVE
        CLR     4(R3)           ;; DISABLE TRANSMITTER

        MOV     D.RCCB(R5),R4   ;; ACTIVE RECEIVE CCB TO R4
        BEQ     20$             ;; NONE THERE - SKIP IT
        CALL    $RDBRT          ;; RETURN BUFFER TO THE POOL

20$:    CLR     D.RCCB(R5)      ;; NO RECEIVE CCB ASSIGNED
        CLR     R4              ;; CLEAR R4 FOR PARAMETER USE
        BISB    D.SLN(R5),R4    ;; SET SYSTEM LINE NUMBER IN R4
        CALL    $RDBQP          ;; PURGE BUFFER WAIT QUEUE REQUESTS

        BISB    #DD.STR,D.FLAG(R5) ;; LINE IS NO LONGER STARTED
        TST     D.TCCB(R5)      ;; IS THERE AN ACTIVE TRANSMIT CCB ?
        BEQ     CTLCMP          ;; NO -- POST CONTROL COMPLETE

        MOV     (SP)+,D.KCCB(R5)   ;; SAVE THE CONTROL CCB FOR TIMEOUT
        MOVB    #1,(R5)            ;; MAKE SURE THE TIMER IS ACTIVE
        ASYRET                     ;; RETURN WITH ASYNCHRONOUS COMPLETION


        .SBTTL  $SDENB  --  ENABLE THE LINE AND DEVICE
;-----------------------------------------------------------;
;        E N A B L E   L I N E   A N D   D E V I C E        ;
;-----------------------------------------------------------;
$SDENB::
        MOV     D.RDBF(R5),R3   ;; RECEIVER CSR ADDRESS [SEL 2] TO R3
        BIS     #TXRSET,2(R3)   ;; RESET THE DEVICE (1-US SINGLE-SHOT)
```

```
        ADD     #D.DCHR+2,R5    ;; POINT TO CHARACTERISTICS WORD #1
        BIT     #DC.ADR,(R5)+   ;; 16-BIT STATION ADDRESS ?
        BEQ     20$             ;; NO -- SHOULD BE ALL SET
        SWAB    (R5)            ;; USE THE HIGH-ORDER BYTE IN DPV-11
20$:    BIC     #^C<DPADRC>,(R5) ;;CLEAR HIGH-ORDER BYTE OF 'D.STN' WORD
        BIS     #INPRM,(R5)     ;;SETUP INITIAL PARAMETERS
        BIC     #DC.ADR,-(R5)   ;; ADDRESS-SIZE NO LONGER SIGNIFICANT
        CMPB    #DC.SPS,(R5)    ;; SDLC PRIMARY-STATION MODE ?
        BEQ     40$             ;; YES - FLAGS ARE SETUP AS IS
        CMPB    #DC.SSS,(R5)    ;; SDLC SECONDARY-STATION MODE ?
        BNE     60$             ;; NO -- OPERATING MODE INVALID
        BIS     #DPSECS,2(R5)   ;; ENABLE STATION ADDRESS CHECKING

40$:    BIS     #DSDTR,-(R3)    ;; ASSERT 'DATA TERMINAL READY' LINE
        BICB    #DD.ENB,D.FLAG-D.DCHR-2(R5) ;; LINE IS ENABLED
        BR      CTLCMP          ;; POST CONTROL FUNCTION COMPLETE

60$:    MOV     #CS.ERR!CS.DEV,R3 ;; ERROR STATUS - INVALID PROTOCOL
        BR      CTLERR          ;; POST CONTROL COMPLETE WITH ERROR

        .SBTTL  $SDDIS  --  DISABLE THE LINE
;
$SDDIS::
        MOV     #CS.ERR!CS.ENB,R3       ;; ERROR CODE IF NOT STOPPED
        BITB    #DD.STR,D.FLAG(R5)      ;; IS LINE STATE CORRECT ?
        BEQ     CTLERR                  ;; NO -- REJECT THE DISABLE

        MOV     D.RDBF(R5),R3           ;; ADDRESS OF RECEIVER CSR [SEL 2]
        CLR     -(R3)                   ;; DISABLE RECEIVER + TURN DTR OFF
        MOVB    #DD.ENB!DD.STR,D.FLAG(R5) ;; LINE NO LONGER ENABLED
        BR      CTLCMP                  ;; CLEAR CARRY AND EXIT


        .SBTTL  $SDMSN  --  SENSE MODEM STATUS
;-----------------------------------------------------------;
;            S E N S E   M O D E M   S T A T U S            ;
;-----------------------------------------------------------;
$SDMSN::
        CLR     R4              ;; CLEAR R4 FOR RETURN CODES
        MOV     D.RDBF(R5),R3   ;; ADDRESS OF RECEIVER CSR [SEL 2]

        BIT     #DSDSR,-(R3)    ;; IS THE DATA-SET READY ?
        BEQ     20$             ;; NO --
        BIS     #MC.DSR,R4      ;; YES - SET INDICATOR IN R4

20$:    BIT     #DSRING,(R3)    ;; IS THE PHONE RINGING ?
        BEQ     40$             ;; NO --
        BIS     #MC.RNG,R4      ;; YES - SET INDICATOR IN R4

40$:    BIT     #DSCARY,(R3)    ;; IS THERE CARRIER PRESENT ?
        BEQ     60$             ;; NO -- POST COMPLETE
        BIS     #MC.CAR,R4      ;; YES - SET INDICATOR IN R4

60$:    MOV     R4,(SP)         ;; RETURN RESULTS IN (SAVED) R4
        BR      CTLCMP          ;; POST CONTROL FUNCTION COMPLETE

        .END
        .TITLE  DPV - BYTE ORIENTED DPV-11 DEVICE DRIVER MODULE
        .IDENT  /X00/
;
; COPYRIGHT (C) 1980 BY
; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
;
```

```
;       EXAMPLE OF AN APPLICATION RSX-11M BYTE ORIENTED DPV-11 DEVICE DRIVER
;
        .MCALL  $INTSX,$INTXT,INHIB$,ENABL$
        .MCALL  CCBDF$,TMPDF$,$LIBCL
        .MCALL  MDCDF$
        .MCALL  CHADF$
        MDCDF$                  ; DEFINE MODEM CONTROL SYMBOLS
        CCBDF$                  ; DEFINE THE CCB OFFSETS
        TMPDF$                  ; DEFINE LINE TABLE OFFSET MACROS
        CHADF$                  ; DEFINE DEVICE CHARACTERISTICS
;
; LOCAL SYMBOL DEFINITIONS
;
; TRANSMITTER FLAGS
;
TINIT=  000010                  ; INITIAL TRANSMIT STATUS (HALF DUPLEX)
TXENA=  000020                  ; TRANSMIT ENABLE
TXINT=  000100                  ; TRANSMIT INTERRUPT ENABLE
TXACT=  000002                  ; TRANSMIT ACTIVE
TSOM=   000400                  ; TRANSMIT START OF MESSAGE
TEOM=   001000                  ; TRANSMIT END OF MESSAGE
;
; RECEIVE CSR FLAGS
;
RCVEN=  000020                  ; RECEIVE ENABLE
RXINT=  000100                  ; RECEIVE INTERRUPT ENABLE
CRC=    3*400                   ; RECEIVE CRC CHECK
SSYN=   020000                  ; STRIP SYNC
PROSEL= 040000                  ; PROTOCOL SELECTION (BYTE)
RINIT=  RXINT!RCVEN!DTR         ; INITIAL RECEIVE STATUS
INPRM=  SSYN!PROSEL!CRC         ; INITIALIZATION FLAGS
;
; MODEM STATUS FLAGS
;
RTS=    000004                  ; REQUEST TO SEND LEAD
CTS=    020000                  ; CLEAR TO SEND
DTR=    000002                  ; DATA TERMINAL READY
DSR=    001000                  ; DATA SET READY
RING=   040000                  ; RING INDICATOR
;
; DPV11 DEVICE DRIVER DISPATCH TABLE
;
$DPVTB::.WORD   DPASX           ; TRANSMIT ENABLE
        .WORD   DPASR           ; RECEIVE ENABLE (ASSIGN BUFFER)
        .WORD   DPKIL           ; KILL I/O
        .WORD   DPCTL           ; CONTROL INITIATION
        .WORD   DPTIM           ; TIME OUT
;+
; **-$DPVRI-DPV11 RECEIVE INTERRUPT SERVICE ROUTINE
;
; THE DEVICE INTERRUPT IS VECTORED TO THE DEVICE LINE TABLE
; BY THE HARDWARE AND THIS ROUTINE IS ENTERED BY A
; 'JSR R5,$DPVRI' INSTRUCTION AT THE BEGINNING OF THE LINE
; TABLE.
;
; INPUTS:
;
;       R5 = ADDRESS OF DEVICE LINE TABLE + 4
;       STACK:
;       0(SP) = SAVED R5
;       2(SP) = INTERRUPTED BIAS
;       4(SP) = INTERRUPTED PC
;       6(SP) = INTERRUPTED PS
;
; OUTPUTS:
```

```
;
; ETC.
;-

$DPVRI::
        MOV     R4,-(SP)        ;;; SAVE R4
        MOV     (R5)+,R4        ;;; GET ADDRESS OF RECEIVER DATA BUFFER
        MOV     (R4),R4         ;;; GET CHARACTER AND FLAGS
        BMI     DPRHO           ;;; ANY ERROR IS RECEIVER OVERRUN

        .IF DF  M$$MGE

        MOV     KISAR6,-(SP)    ;;; SAVE CURRENT MAP
        MOV     (R5)+,KISAR6    ;;; MAP TO DATA BUFFER

        .IFTF

        MOVB    R4,@(R5)+       ;;; STORE CHARACTER IN RECEIVE BUFFER

        .IFT

        MOV     (SP)+,KISAR6    ;;; RESTORE PREVIOUS MAPPING

        .ENDC

        DEC     (R5)            ;;; DECREMENT REMAINING BYTE COUNT
        BEQ     DPRCP           ;;; IF EQ RECEIVE COMPLETE
        INC     -(R5)           ;;; ADVANCE BUFFER ADDRESS
        MOV     (SP)+,R4        ;;; RESTORE REGISTERS
        $INTXT                  ;;; EXIT THE INTERRUPT

;
; EXCEPTIONAL RECEIVE SERVICE ROUTINES
;
; HARDWARE OVERRUN
;
        .ENABL  LSB

DPRHO:  ADD     #<RCNT-RDBF-2>,R5 ;;; POINT TO COUNT CELL
        MOV     #100001,RFLAG-RCNT(R5) ;;; SET FLAGS TO COMPLETE REQUEST AND
                                ;;; CLEAR RECEIVE ACTIVE ON EXIT
        MOV     #CS.ERR+CS.ROV,RSTAT-RCNT(R5) ;;; SET OVERRUN STATUS

;
; RECEIVE BYTE COUNT RUNOUT
;

DPRCP:  MOV     R4,(R5)+        ;;; SAVE CRC FLAG AND POINT TO PRIORITY
        MOV     RDBF-RPRI(R5),R4 ;;; GET RECEIVE DATA BUFFER ADDRESS
        BIC     #RXINT,-(R4)    ;;; CLEAR RECEIVER INTERRUPT ENABLE
        MOV     (SP)+,R4        ;;; RESTORE R4 SO '$INTSV' IS HAPPY
        $INTSX                  ;;; DO A TRICKY $INTSV (R5 PRESAVED BUT NOT R4)
        MOV     R3,-(SP)        ;;  SAVE AN ADDITIONAL REGISTER
        TST     (R5)+           ;;  POINT TO FLAGS WORD
        ASR     (R5)+           ;;  LOAD C-BIT FROM FLAGS (BIT 0)
        BCS     20$             ;;  IF CS DATA, POST COMPLETION
        MOV     (R5),R4         ;;  GET PRIMARY CCB ADDRESS
        .LIST MEB
        $LIBCL  HDRA-RPRIM,R5,$DDHAR,SAV ;; CALL DDHAR THROUGH LINE TABLE
        .NLIST  MEB
        ROR     -2(R5)          ;;  SAVE 'FINAL SEEN' IN FLAGS (BIT 15 SET)
        TST     R3              ;;  EXAMINE BYTE COUNT FOR THIS MESSAGE
        BMI     10$             ;;  IF MI AN INVALID HEADER RECEIVED
        BEQ     7$              ;;  IF EQ SET TO RECEIVE REST OF HEADER
        ADD     #2,R3           ;;  ACCOUNT FOR BCC IN CURRENT COUNT
        MOV     R3,RPCNT-RPRIM(R5) ;; SAVE DATA COUNT UNTIL HEADER CRC
```

```
                                ;;   IS CHECKED
7$:     MOV     #5,R3           ;; GET REMAINING HEADER
        INC     -(R5)           ;;  MARK DATA IN PROGRESS IN FLAGS (BIT 0 SET)
        ADD     R3,@-(R5)       ;;  INCLUDE CURRENT COUNT IN TOTAL COUNT
        ADD     #RCNT-RTHRD,R5  ;;  POINT TO CURRENT COUNT
        MOV     R3,(R5)         ;;  SET UP CURRENT BYTE COUNT
        INC     -(R5)           ;;  MOVE BUFFER ADDRESS PAST BCC

        .IF DF  M$$MGE

        MOV     -4(R5),R3       ;;  GET ADDRESS OF RECEIVE DATA BUFFER

        .IFF

        MOV     -(R5),R3        ;;  GET ADDRESS OF RECEIVE DATA BUFFER

        .ENDC

        BR      REXT0           ;;  FINISH IN COMMON CODE

;
; INVALID HEADER RECEIVED
;
10$:    BIT     #CS.MTL,R3      ;;  MESSAGE TOO LONG ?
        BNE     31$             ;;  IF NE YES, POST COMPLETION
        MOV     (R5)+,R4        ;;  RECOVER PRIMARY CCB ADDRESS
        CALL    BUFUSE          ;;  SET UP THIS CCB AGAIN (CLEARS 'RSTAT')
        MOV     RDBF-RPRIM(R5),R3 ;; SET POINTER TO REC. DAT. BUFF.
        BR      40$             ;;  CLEAR RECEIVE ACTIVE TO FORCE RESYNC

;
; POST COMPLETION ON RECEIVE COMPLETE
;
;       R5 =  POINTS TO PRIMARY CCB ADDRESS
;

20$:    TST     RCNT-RPRIM(R5) ;;  IS CRC ERROR FLAG SET ?
        BMI     25$             ;;  IF MI, YES - CRC IS VALID
        MOV     #CS.ERR+CS.DCR,R3 ;;  ELSE SET CRC ERROR STATUS FOR DLC
        BR      31$             ;; GO RETURN BUFFER
25$:    MOV     RPCNT-RPRIM(R5),RCNT-RPRIM(R5) ;; SET REMAINING COUNT
        BEQ     30$             ;; NONE SO END OF MESSAGE
        ADD     RPCNT-RPRIM(R5),@RTHRD-RPRIM(R5) ;; SET TOTAL COUNT IN CCB
        SEC                     ;; FORCE C BIT
        ROL     RFLAG-RPRIM(R5) ;; PUT  Q SYNC BACK & MARK NON HEADER
        INC     RADD-RPRIM(R5)  ;; INCLUDE LAST CHAR IN BUFFER
        MOV     RDBF-RPRIM(R5),R3 ;; GET CSR FOR EXIT
        BR      REXT            ;; TAKE COMMON EXIT
30$:    CLR     R3              ;; GET GOOD STATUS
31$:    MOV     (R5)+,R4        ;;  GET PRIMARY CCB ADDRESS
        BIS     (R5),R3         ;;  PICK UP ADDITIONAL STATUS
        CALL    $DDRCP          ;;  POST RECEIVE COMPLETION
        MOV     RDBF-RSTAT(R5),R3 ;;  GET ADDRESS OF RECEIVE DATA BUFFER
        CALL    BUFSET          ;;  SET UP NEXT RECEIVE BUFFER
        BCS     REXT1           ;;  IF CS NO BUFFER AVAILABLE TURN OFF RECEIVER
        BNE     40$             ;;  IF NE CLEAR RECEIVE ACTIVE TO RESYNC
REXT:   CLR     RPCNT-RPRIM(R5) ;; RESET PARTIAL COUNT
REXT0:  BIS     #RXINT,-(R3)    ;;  ENABLE RECEIVER INTERRUPTS
REXT1:  MOV     (SP)+,R3        ;;  RESTORE R3
        RETURN                  ;;  RETURN TO SYSTEM

40$:                            ;;  REF LABEL
;
; CLEAR RECEIVE ACTIVE TO FORCE RESYNC
;
;       R3 = ADDRESS OF RECEIVE DAT BUFFER
```

```
;       R5 = ADDRESS OF 'RPRIM'
;

DPCRA:  CLR     -(R5)               ;;  CLEAR FLAGS WORD
        BIC     #RCVEN,-(R3)        ;;  CLEAR RECEIVE ACTIVE FOR RESYNC
        CLR     RPCNT-RFLAG(R5)     ;; RESET PARTIAL COUNT
        BIS     #CS.RSN,RSTAT-RFLAG(R5) ;;  INDICATE A RESYNC
        BIS     #RINIT,(R3)         ;;  ENABLE RECEIVER
        BR      REXT1               ;;  FINISH IN COMMON CODE

        .DSABL  LSB
;+
; **-$DPVTI-DPV11 TRANSMIT INTERRUPT SERVICE
;
;
; THIS ROUTINE IS ENTERED ON A TRANSMITTER INTERRRUPT VIA
; A 'JSR R5,DPVTI' WITH R5 CONTAINING THE ADDRESS OF THE
; DEVICE LINE TABLE OFFSET BY 'TCSR'.
;
; INPUTS:
;
;       R5 = ADDRESS OF DEVICE LINE TABLE + 'TCSR'
;       STACK CONTAINS:
;       0(SP) = INTERRUPTED R5
;       2(SP) = INTERRUPTED BIAS
;       4(SP) = INTERRUPTED PC
;       6(SP) = INTERRUPTED PS
;
; OUTPUTS:
;
; ETC.
;-
        .ENABL  LSB

$DPVTI::
        MOV     R4,-(SP)            ;;; SAVE R4
        MOV     (R5)+,R4            ;;; GET TRANSMITTER CSR ADDRESS
        TST     (R4)+               ;;; TEST FOR UNDERRUN
        BMI     10$                 ;;; IF MI, UNDERRUN - WAIT FOR TIMEOUT
        DEC     TCNT-TCSR-2(R5)     ;;; DECREMENT COUNT
        BEQ     20$                 ;;; IF EQ, BYTE COUNT RUNOUT

        .IF DF  M$$MGE

        MOV     KISAR6,-(SP)        ;;; SAVE CURRENT MAPPING
        MOV     (R5)+,KISAR6        ;;; MAP TO DATA BUFFER

        .IFTF

        MOVB    @(R5)+,(R4)         ;;; OUTPUT A CHARACTER

        .IFT

        MOV     (SP)+,KISAR6        ;;; RESTORE PREVIOUS MAPPING

        .IFTF

        INC     -(R5)               ;;; UPDATE BUFFER ADDRESS
        MOV     (SP)+,R4            ;;; RESTORE R4
        $INTXT

;
; TRANSMITTER UNDERRUN
;
; DISABLE TRANSMITTER INTERRUPTS AND WAIT FOR A TIMEOUT
```

```
;
10$:    BISB    #TSOM/400,1(R4) ;;; CLEAR UNDERRUN BIT
        MOV     #TUNST,TSTAT-TCSR-2(R5) ;;; SET STATE TO DISABLE TRANSMITTER

;
; TRANSMIT BYTE COUNT RUNOUT
;
; OUTPUT TO STATE PROCESSING ROUTINES:
;
;       R3 = ADDRESS OF TRANSMITTER CSR
;       R5 = ADDRESS OF THREAD WORD CELL
;
20$:    ADD     #TPRI-TCSR-2,R5 ;;; POINT TO PRIORITY DATA
        BIC     #TXINT,-(R4)    ;;; CLEAR INTERRUPT ENABLE
        MOV     (SP)+,R4        ;;; RESTORE R4 SO '$INTSV' IS HAPPY
        $INTSX                  ;SAVE WITH R5 ON STACK BUT NOT R4

        .IFT

        MOV     KISAR6,-(SP)    ;;  SAVE CURRENT MAPPING

        .IFTF

        MOV     R3,-(SP)        ;;  SAVE AN ADDITIONAL REGISTER
        MOV     TCSR-TSTAT(R5),R3 ;;  GET TRANSMITTER CSR ADDRESS
        CALLR   @(R5)+          ;;  DISPATCH TO PROCESSING ROUTINE

        .DSABL  LSB

;+
; **-DPASX-ASSIGN A TRANSMIT BUFFER
;
;
; THIS ROUTINE IS ENTERED VIA THE MATRIX SWITCH TO
; QUEUE A CCB FOR TRANSMISSION.
;
; INPUTS:
;
;       R4 = ADDRESS OF CCB TO TRANSMIT
;       R5 = ADDRESS OF DEVICE LINE TABLE
;
; OUTPUTS:
;
;       IF THE TRANSMITTER IS IDLE, TRANSMISSION IS
;       INITIATED; OTHERWISE, THE CCB (OR CHAIN) IS QUEUED TO
;       THE END OF THE SECONDARY CHAIN.
;
; REGISTERS MODIFIED:
;
;       R3, R4, AND R5
;-
DPASX:
        MOV     TCSR(R5),R3     ; GET TRANSMITTER CSR ADDRESS
        BIC     #TXINT,(R3)     ; DISABLE TRANSMITTER INTERRUPTS
        ADD     #TPRIM,R5       ; POINT TO PRIMARY CELL

        .IFT

        MOV     KISAR6,-(SP)    ; SAVE CURRENT MAPPING

        .IFTF

        MOV     R3,-(SP)        ; SAVE R3
        TST     (R5)+           ; PRIMARY ASSIGNED ?
```

```
        BNE     10$             ; IF NE, YES - QUEUE TO SECONDARY CHAIN
        CALL    TBSET           ; SET UP PRIMARY
        BIT     #TXACT,(R3)     ; TRANSMITTER ACTIVE ?
        BEQ     STSTR           ; IF EQ, NO - START IMMEDIATELY
        MOV     #STSTR,-(R5)    ; SET STATE FOR STARTUP
        BR      WAITI           ; WAIT FOR INTERRUPT

10$:    MOV     R4,-(SP)        ; SAVE POINTER TO FIRST CCB
20$:    MOV     R5,R4           ; COPY POINTER TO CCB
        MOV     (R4),R5         ; GET NEXT CCB
        BNE     20$             ; IF NE, KEEP GOING
        MOV     (SP)+,(R4)      ; LINK NEW CCB CHAIN TO LAST CCB
        BR      TEXT2           ; FINISH IN COMMON CODE

;+
; **-STSTR-STARTUP STATE PROCESSING
;
;-

STSTR:  BIS     #RTS,-4(R3)     ; ASSERT REQUEST TO SEND
        BIS     #TXENA,(R3)     ; ENABLE TRANSMITTER
        MOVB    TIMS-TTHRD(R5),TIME-TTHRD(R5) ; START TIMER

;+
; **-STCTS-WAIT FOR CLEAR TO SEND STATE PROCESSING
;
;-

STCTS:  BIT     #CTS,-4(R3)     ; IS CLEAR TO SEND UP ?
        BNE     STSYN           ; IF NE, YES - START SYNC TRAIN
        MOV     #STCTS,-(R5)    ; SET STATE FOR CTS
        MOV     #$PADB,R4       ; SET ADDRESS OF PAD BUFFER
        MOV     #TSOM,-(SP)     ; SET TSOM, CLEAR TEOM
        BR      TEXT1           ; FINISH IN COMMON CODE

;+
; **-STSYN-SYNC TRAIN REQUIRED STATE PROCESSING
;
;-

STSYN:  MOV     #STDAT,-(R5)    ; SET STATE FOR DATA
        MOV     #$SYNB,R4       ; SET ADDRESS OF SYNC BUFFER
        MOV     #TSOM,-(SP)     ; SET TSOM, CLEAR TEOM
        BR      TEXT0           ; FINISH IN COMMON CODE

;+
; **-STCRC-SEND CRC STATE PROCESSING
;
;-
        .ENABL  LSB

STCRC:  BIS     #TEOM,2(R3)     ; SEND CRC
        CALL    TPOST           ; POST COMPLETION AND SET UP NEXT CCB
        BNE     10$             ; IF NE, NOTHING MORE TO SEND
        MOV     #STDAT,-(R5)    ; ASSUME NEXT STATE IS SEND SYNC'S
        BIT     #CF.SYN,C.FLG-C.BUF(R4) ; ARE SYNC'S REQUIRED ?
        BEQ     20$             ; IF EQ, NO - LEAVE ASSUMED STATE
        MOV     #STSYN,(R5)     ; ELSE CHANGE STATE TO SEND SYNC'S
        BR      20$             ; WAIT FOR CRC TO BE SENT

10$:    MOV     #STIDL,-(R5)    ; SET STATE TO IDLE
        BIC     #TXENA,(R3)     ; SHUT DOWN TRANSMITTER
20$:                            ;

;+
```

```
; **-WAITI-WAIT FOR INTERRUPT
;
;-

WAITI:  MOV     #1,TCNT-TSTAT(R5) ; WAIT FOR ONE INTERRUPT
        MOVB    TIMS-TSTAT(R5),TIME-TSTAT(R5) ; START TIMER
        BR      TEXT2             ; FINISH IN COMMON CODE

;+
; **-STIDL-IDLE STATE PROCESSING
;
;-

STIDL:  BIC     #RTS,-4(R3)       ; DROP REQUEST TO SEND
        TST     -(R5)             ;
30$:
        CLRB    TIME-TSTAT(R5)    ; CLEAR TIMER
        BR      TEXT3             ; FINISH IN COMMON CODE

        .DSABL  LSB
;+
; **-TUNST-TRANSMIT DATA UNDER RUN STATE
;
;       RETURN ALL TRANSMIT BUFFERS TO HIGHER LEVEL
;-
TUNST:  ADD     #-TTHRD,R5        ;;TIMEOUT EXPECTS DDM LINE TABLE POINTER
        CLRB    (R5)                      ;;RESET TIMER
        CALL    DPTIM                     ;;FAKE A TIMEOUT TO RETURN BUFFERS
        MOV     #STIDL,TSEC-TSTAT(R5)     ;;SET STATE TO IDLE
        BR      TEXT3                     ;;TAKE COMMON EXIT
;+
; **-STDAT-DATA STATE PROCESSING
;
;-

STDAT:  MOV     (R5),R4           ; GET ADDRESS OF FLAGS WORD FROM THREAD
        ADD     #C.FLG-C.STS,(R5) ; UPDATE THREAD POINTER
        TST     (R4)+             ; LAST BUFFER THIS CCB ? (BIT 15 SET)
        BPL     10$               ; IF PL, NO
        CALL    TPOST             ; POST COMPLETION AND SET UP NEXT CCB
10$:    MOV     #STDAT,-(R5)      ; ASSUME DATA CONTINUES
        BIT     #CF.EOM,C.FLG-C.BUF(R4) ; SEND CRC FOLLOWING THIS BUFFER ?
        BEQ     20$               ; IF EQ, NO - LEAVE ASSUMED STATE
        MOV     #STCRC,(R5)       ; ELSE CHANGE STATE FOR CRC TO BE SENT
20$:    CLR     -(SP)             ; CLEAR TSOM, CLEAR TEOM

;+
; **-TEXT0-COMMON EXIT ROUTINES
; **-TEXT1-
; **-TEXT2-
; **-TEXT3-
;
;-

TEXT0:  MOVB    TIMS-TSTAT(R5),TIME-TSTAT(R5) ; START TIMER
TEXT1:  ADD     #TCSR-TSTAT+2,R5 ; POINT TO CURRENT BUFFER CELL

        .IFT

        MOV     (R4)+,(R5)+       ; COPY RELOCATION BIAS

        .IFF

        TST     (R4)+             ; SKIP OVER RELOCATION BIAS IN CCB
```

```
        .IFTF

        MOV     (R4)+,(R5)+     ; COPY VIRTUAL ADDRESS
        MOV     (R4),(R5)       ;   AND THE BYTE  COUNT

        .IFT

        MOV     -4(R5),KISAR6   ; MAP TO DATA BUFFER

        .IFTF

        BISB    @-2(R5),(SP)    ; BUILD CHARACTER TO OUTPUT
        INC     -2(R5)          ; UPDATE VIRTUAL ADDRESS
        MOV     (SP)+,2(R3)     ; OUTPUT CHARACTER AND FLAGS
TEXT2:  BIS     #TXINT,(R3)     ; ENABLE TRANSMITTER INTERRUPTS
TEXT3:  MOV     (SP)+,R3        ; RESTORE R3
        .IFT
```

GLOSSARY

Asynchronous Transmission
Transmission in which time intervals between transmitted characters may be of unequal length. Transmission is controlled by start and stop elements at the beginning and end of each character. Also called start-stop transmission.

BDIN
Data Input on the LSI-II bus.

BDOUT
Data Output on the LSI-II bus.

BIAKI
Interrupt Acknowledge.

Bit-Stuff Protocol
Zero insertion by the transmitter after any succession of five continuous ones designed for bit-oriented protocols such as IBM's Synchronous Data Link Control (SDLC).

Bits per Second (b/s)
Bit transfer rate per unit of time.

BIRQ
Interrupt Request priority level for LSI-11 bus.

BRPLY
LSI-11 Bus Reply. BRPLY is asserted in response to BDIN or BDOUT.

BSYNC
Synchronize – asserted by the bus master device to indicate that it has placed an address on the bus.

Buffer
Storage device used to compensate for a difference in the rate of data flow when transmitting data from one device to another.

BWTBT
Write Byte.

CCITT
Comite Consultatif Internationale de Telegraphie et Telephonie – An international consultative committee that sets international communications usage standards.

Control and Status Registers (CSRs)
Communication of control and status information is accomplished through these registers.

Cyclic Redundancy Check (CRC)
An error detection scheme in which the check character is generated by taking the remainder after dividing all the serialized bits in a block of data by a predetermined binary number.

Data Link Escape (DLE)
A control character used exclusively to provide supplementary line control signals (control character sequences or DLE sequences). These are 2-character sequences where the first character is DLE. The second character varies according to the function desired and the code used.

Data-Phone DIGITAL Service (DDS)
A communicaitons service of the Bell System in which data is transmitted in digital rather than analog form, thus eliminating the need for modems.

DIGITAL Data Communications Protocol (DDCMP)
DIGITAL's standard communications protocol for character-oriented protocol.

Direct Memory Access (DMA)
Permits I/O transfer directly into or out of memory without passing through the processor's general registers.

Electronic Industries Association (EIA)
A standards organization specializing in the electrical and functional characteristics of interface equipment.

Full-Duplex (FDX)
Simultaneous 2-way independent transmission in both directions.

Field-Replaceable Unit (FRU)
Refers to a faulty unit not to be repaired in the field. Unit is replaced with a good unit and faulty unit is returned to predetermined location for repair.

Half-Duplex (HDX)
An alternate, one-way-at-a-time independent transmission.

LARS
Field Service Labor Activity Reporting System.

Non-Processor Request (NPR)
Direct memory access-type transfers, (see DMA).

Protocol
A formal set of conventions governing the format and relative timing of message exchange between two communicating processes.

RS-232-C
EIA standard single-ended interface levels to modem.

RS-422-A
EIA standard differential interface levels to modem.

RS-423-A
EIA standard single-ended interface levels to modem.

RS-449

EIA standard connections for RS-422-A and RS-423-A to modem interface.

Synchronous Transmission

Transmission in which the data characters and bits are transmitted at a fixed rate with the transmitter and receiver synchronized.

V.35

(CCITT Standard) – Differential current mode-type signal interface for high-speed modems.

DPV11 SERIAL SYNCHRONOUS INTERFACE TECHNICAL MANUAL
EK-DPV11-TM-002

Reader's Comments

Your comments and suggestions will help us in our continuous effort to improve the quality and usefulness of our publications.

What is your general reaction to this manual? In your judgement is it complete, accurate, well organized, well written, etc? Is it easy to use? _____

What features are most useful? _____

What faults or errors have you found in the manual? _____

Does this manual satisfy the need you think it was intended to satisfy? _____

Does it satisfy *your* needs? _____ Why? _____

☐ Please send me the current copy of the *Technical Documentation Catalog*, which contains information on the remainder of DIGITAL's technical documentation.

Name _____ Street _____
Title _____ City _____
Company _____ State/Country _____
Department _____ Zip _____

Additional copies of this document are available from:

Digital Equipment Corporation
444 Whitney Street
Northboro, MA 01532

Attention: Printing and Circulation Services (NR2/M15)
Customer Services Section

Order No. **EK-DPV11-TM-002**

Fold Here

Do Not Tear – Fold Here and Staple

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MERRIMACK, NH

POSTAGE WILL BE PAID BY ADDRESSEE

Digital Equipment Corporation
Educational Services Development & Publishing
Continental Blvd. (MK1/2M26)
Merrimack, N.H. 03054

digital

digital equipment corporation

Printed in U.S.A.


pdp11
digital

EK-DPV11-UG-001

DPV11 serial synchronous interface user guide

digital equipment corporation ● merrimack, new hampshire

1st Edition, August 1980

Printed in U.S.A.

This document was set on DIGITAL's DECset-8000 computerized typesetting system.

CONTENTS

CONTENTS (Cont)

ILLUSTRATIONS

ILLUSTRATIONS (Cont)

TABLES

PREFACE

This manual is intended to provide an introduction to the DPV11 Interface and present the information required by the user for configuration, installation and operation.

It contains the following categories of information.

- General description including features, specifications, and configurations
- Installation
- Programming

The manual also contains four appendixes which include diagnostic information, integrated circuit descriptions, and programming examples.

The DPV11 Field Maintenance Print Set (MP00919) contains useful additional information.

CHAPTER 1
INTRODUCTION

1.1 SCOPE

This chapter contains introductory information about the DPV11. It includes a general description, and a brief overview of the DPV11 operation, features, general specifications, and configurations.

1.2 DPV11 GENERAL DESCRIPTION

The DPV11 is a serial synchronous line interface for connecting an LSI-11 bus to a serial synchronous modem that is compatible with EIA RS-232-C interface standards and EIA RS-423-A and RS-422-A electrical standards. EIA RS-422-A compatibility is provided for use in local communications only (timing and data leads only). The DPV11 is intended for character-oriented protocols such as BISYNC, byte count-oriented protocols such as DDCMP, or bit-oriented data communication protocols such as SDLC. The DPV11 does not provide automatic error generating and checking for BISYNC.

The DPV11 consists of one double-height module and may be connected to an EIA RS-232-C modem by a BC26L-25 (RS-232-C) cable.

The DPV11 is a bus request device only and must rely on the system software for service. Interrupt control logic generates requests for the transfer of data between the DPV11 and the LSI-11 memory by means of the LSI-11 bus. (Figure 1-1 shows the DPV11 system.)

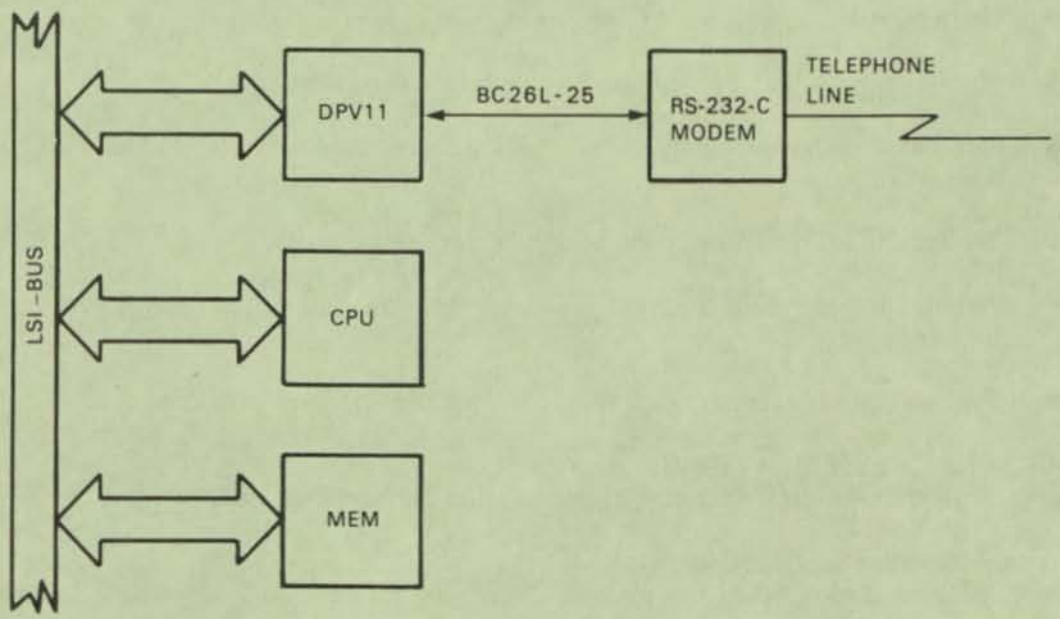


Figure 1-1 DPV11 System

1.3 DPV11 OPERATION

The DPV11 is a double-buffered program interrupt interface that provides parallel-to-serial conversion of data to be transmitted and serial-to-parallel conversion of received data. The DPV11 can operate at speeds up to 56K b/s.* It has five 16-bit registers which can be accessed in word or byte mode. These registers are assigned a block of four contiguous LSI-11 bus word addresses that start on a boundary with the low-order three bits being zeros. This block of addresses is jumper-selectable and may be located anywhere between 160000_8 and 177776_8. Two of these registers share the same address. One is accessed during a read from the address, the other during a write to the address. For a detailed description of each of the five registers, refer to Chapter 3. These registers are used for status and control information as well as data buffers for both the transmitter and receiver portions of the DPV11.

1.4 DPV11 FEATURES

Features of the DPV11 include:

- Full-duplex or half-duplex operation
- Double-buffered transmitter and receiver
- EIA RS-232-C compatibility
- All EIA RS-449 Category I modem control
- Partial Category II modem control to include incoming call, test mode, remote loopback, and local loopback
- Program interrupt on transitions of modem control signals
- Operating speeds up to 56K b/s (may be limited by software or CPU memory)
- Software-selectable diagnostic loopback
- Operation with bit-, byte count-, or character-oriented protocols
- Internal cyclic redundancy check (CRC) character generation and checking (not usable with BISYNC)
- Internal bit-stuff and detection with bit-oriented protocols.
- Programmable sync character, sync insertion, and sync stripping with byte count-oriented protocols.
- Recognition of secondary station address with bit-oriented protocols.

1.5 GENERAL SPECIFICATIONS

This paragraph contains environmental, electrical, and performance specifications for the DPV11.

1.5.1 Environmental Specifications

The DPV11 is designed to operate in a Class C environment as specified by DEC Standard 102 (extended).

Operating Temperature	5° C (41° F) to 60° C (140° F)
Relative Humidity	10% to 90% with a max. wet bulb temperature of 28° C (82° F) and a min. dew point of 2° C (36° F)

*The actual speed realized may be significantly less because of limitations imposed by the software and/or CPU memory refresh.

1.5.2 Electrical Specifications

The DPV11 requires the following voltages from the LSI-11 bus for proper operation.

+12 V at 0.30 A max. (0.15 A typical)
+5 V at 1.2 A max. (0.92 A typical)

The interface includes a charge pump to generate a negative voltage required to power the RS-423-A drivers.

The DPV11 presents 1 ac load and 1 dc load to the LSI-11 bus.

1.5.3 Performance Parameters

Performance parameters for the DPV11 are listed as follows.

Operating Mode	Full or half-duplex
Data Format	Synchronous BISYNC, DDCMP, and SDLC
Character Size	Program-selectable (5–8 bits with character-oriented protocols and 1–8 bits with bit-oriented protocols)
Max. Configuration	16 DPV11 modules per LSI-11 bus
Max. Distance	15 m (50 ft) for RS-232-C. 61 m (200 ft) for RS-423-A/RS-422-A (Distance is directly dependent on speed, and 200 ft is a suggested average. See RS-449 specification for details.)
Max. Serial Data Rates	56K b/s (May be less because of software and memory refresh limitations.)

1.6 DPV11 CONFIGURATIONS

There are two DPV11 configurations, the DA and the DB.

DPV11-DA
Unbundled version consists of:
- M8020 module
- DPV11 Maintenance Reference Card (EK-DPV11-CG)

DPV11-DB
Bundled version consists of:
- M8020 module
- H3259 turn-around connector
- BC26L-25 cable
- DPV11 User Manual (EK-DPV11-UG)
- DPV11 Maintenance Reference Card (EK-DPV11-CG)
- LIB kit (ZJ314-RB)
- Field Maintenance Print Set (MP00919)

Turn-around connectors, cables and documentation may be purchased separately.

1.7 EIA STANDARDS OVERVIEW (RS-449/RS-232-C)

The most common interface standard used in recent years has been the RS-232-C. However, this standard has serious limitations for use in modern data communication systems. The most critical limitations are in speed and distance.

For this reason, RS-449 standard has been developed to replace RS-232-C. It maintains a degree of compatibility with RS-232-C to accommodate an upward transition to RS-449.

The most significant difference between RS-232-C and RS-449 is in the electrical characteristics of signals used between the data communication equipment (DCE) and the data terminal equipment (DTE). The RS-232-C standard uses only unbalanced circuits, while the RS-449 uses both balanced and unbalanced electrical circuits. The specifications for the types of electrical circuits supported by RS-449 are contained in EIA standards RS-422-A for balanced circuits and RS-423-A for unbalanced circuits. These new standards permit much greater transmission speed and will allow greater distance between DTE and DCE. The maximum transmission speeds supported by RS-422-A and RS-423-A circuits vary with cable length; the normal speed limits are 20K b/s for RS-423-A and 2M b/s for RS-422-A, both at 61 m (200 feet).

Another major difference between RS-232-C and RS-449 is that additional leads are needed to support the balanced interface circuits and some new circuit functions. Two new connectors have been specified to accommodate these new leads. One connector is a 37-pin Cinch used in applications requiring secondary channel functions. Some of the new circuits added in RS-449 support local and remote loopback testing, and stand-by channel selection.

CHAPTER 2
INSTALLATION

2.1 INTRODUCTION

This chapter provides all the information necessary for a successful installation and subsequent check-out of the DPV11. Included are instructions for unpacking and inspection, pre-installation, installation and verification of operation.

2.2 UNPACKING AND INSPECTION

The DPV11 is packaged in accordance with commercial packing practices. Remove all packing material and verify that the following are present.

M8020 module
H3259 turn-around connector
BC26L-25 cable
DPV11 User Manual (EK-DPV11-UG)
LIB kit (ZJ314-RB)
Field Maintenance Print Set (MP00919)

Inspect all parts carefully for cracks, loose components or other obvious damage. Report damages or shortages to the shipper immediately, and notify the DIGITAL representative.

2.3 PRE-INSTALLATION REQUIREMENTS

Table 2-1 (Configuration Sheet) provides a convenient, quick reference for configuring jumpers.

Table 2-1 Configuration Sheet

(W1–W2) Driver Attenuation Jumper

Driver	Normal* Configuration	Alternate* Option	Description
Terminal Timing	W1 to W2	Not connected	Bypasses attenuation resistor. Jumper must be removed for certain modems to operate properly.

(W3–W11) Interface Selection Jumpers

Input Signals	Normal* Configuration	Alternate* Option	Description
SQ/TM (PCSCR-5)	W5 to W6		Signal quality
		W7 to W6	Test mode
DM (DSR) (RXCSR-9)	Not connected	W10 to W9	Data mode return for RS-422-A

*Normal configuration is typically RS-423-A compatible. Alternate option is typically RS-422-A compatible.

Table 2-1 Configuration Sheet (Cont)

(W3-W11) Interface Selection Jumpers (Cont)

Output Signals	Normal* Configuration	Alternate* Option	Description
SF/RL (RXCSR-0)	W3 to W4		Select frequency
		W5 to W3	Remote loopback
Local Loopback	W8 to W9	Not connected	Local loopback
	Not connected	W8 to W11	Local loopback (alternate pin)

(W12–W17) Receiver Termination Jumpers

Receiver	Normal* Configuration	Alternate* Option	Description
Receive Data	Not connected	W12 to W13	Connects terminating resistor for RS-422-A compatibility
Send Timing	Not connected	W14 to W15	
Receive Timing	Not connected	W16 to W17	

(W18–W23) Clock Jumpers

Function	Normal* Configuration	Alternate* Option	Description
NULL MODEM CLK	W20 to W18		Sets NULL CLK MODEM CLK to 2 kHz.
		W21 to W18	Sets NULL MODEM CLK to 50 kHz.
Clock Enable	W19 to W21 W22 to W23	W19 to W21 W22 to W23	Always installed except for factory testing.

(W24–W28) Data Set Change Jumpers

Modem Signal Name	Normal* Configuration	Alternate* Option	Description
Data Mode (DSR)	W26 to W24	Not connected	Connects the DSCNG flip-flop to the respective modem status signal for transition detection.
Clear to Send	W26 to W25	Not connected	
Incoming Call	W26 to W27	Not connected	Note: W26 is input to DSCNG flip-flop
Receiver Ready (Carrier Detect)	W26 to W28	Not connected	

*Normal configuration is typically RS-423-A compatible. Alternate option is typically RS-422-A compatible.

Table 2-1 Configuration Sheet (Cont)

Device Address Jumpers

GND	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3
W29	W31	W30	W36	W33	W32	W39	W38	W37	W34	W35

NOTE

The address to which the DPV11 is to respond is daisy-chain jumpered to W29 (GND).

Vector Address Jumpers

D8	D7	D6	D5	D4	D3	Source
W43	W42	W41	W40	W44	W45	W46

NOTE

Vector address to be asserted is daisy-chain jumpered to W46.

NOTE

Table 2-1 shows the recommended normal and alternate jumpering schemes. Any deviation from these will cause diagnostics to fail and require restrapping for full testing and verification. It is recommended that customer configurations that vary from this scheme not be contractually supported.

Prior to installing the DPV11, perform the following tasks.

1. Verify that the following modem interface wire-wrap jumpers are installed (Figure 2-1).

   W26 to W25 to W24 to W28 to W27
   W22 to W23 and W19 to W21
   W18 to W20
   W5 to W6
   W3 to W4
   W8 to W9
   W1 to W2

   This is the normal/RS-423-A shipped configuration. Some of these jumpers may be changed when the module is connected to external equipment for a specific application. The NULL MODEM CLK is set to 2 kHz as shipped.

2. Based on the LSI-11 bus floating vector scheme or user requirements, determine the vector address for the specific DPV11 module being installed and configure W40 through W46 accordingly (Table 2-2).

3. Based on the LSI-11 bus floating address scheme or user requirements, determine the device address range for the DPV11 module and configure W30 through W39 accordingly (Table 2-3). Devices may be physically addressed starting at 160000 and continuing through 177776; however, there may be some software restrictions. The normal addressing convention is as shown in Table 2-3.

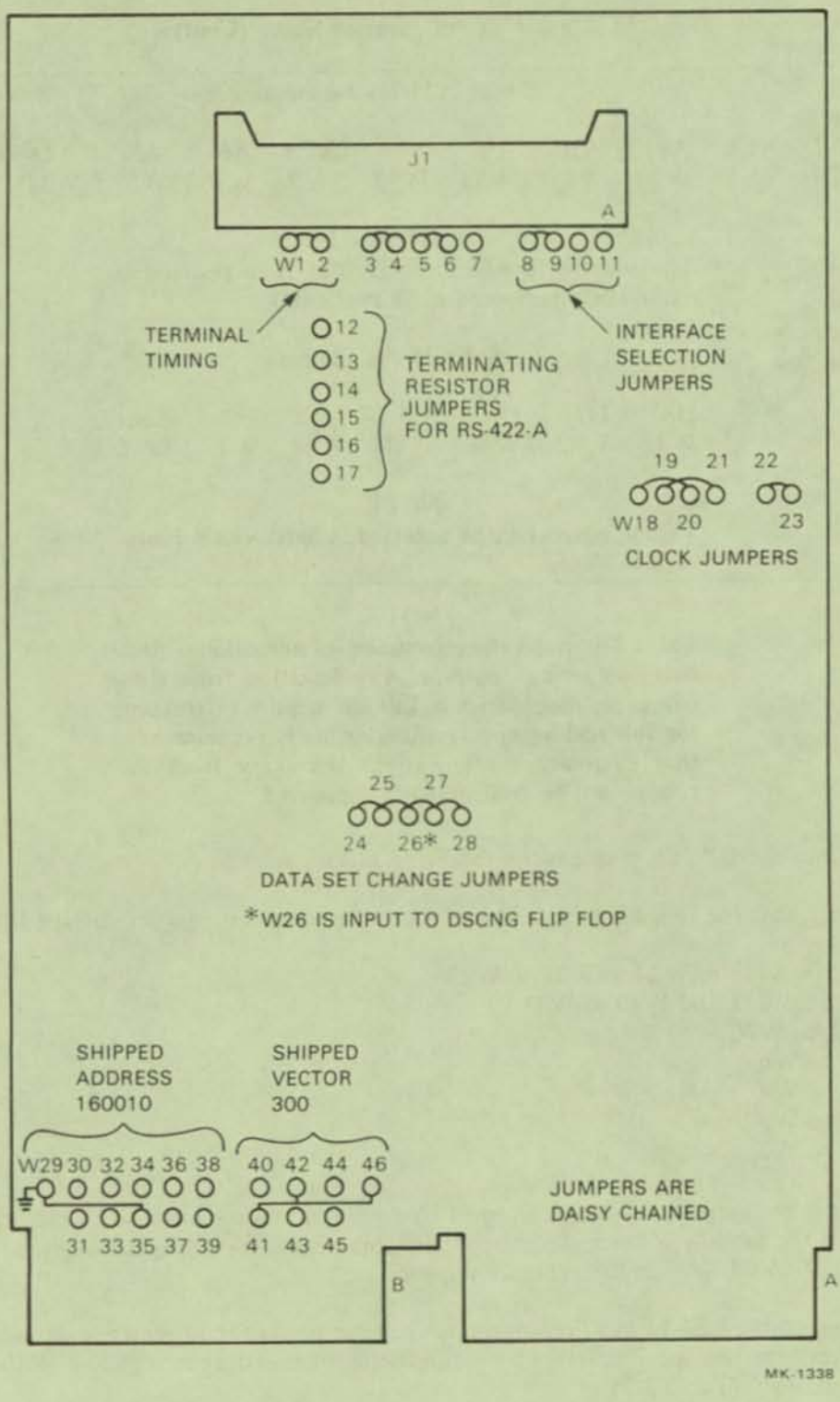


Figure 2-1 DPV11 Jumper Locations

Table 2-2 Vector Address Selection

DPV11 (M8020) VECTOR ADDRESSING

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

JUMPER NUMBER	W43	W42	W41	W40	W44	W45	VECTOR ADDRESS
		X	X				300
		X	X			X	310
		X	X		X		320
		X	X		X	X	330
		X	X	X			340
		X	X	X		X	350
		X	X	X	X		360
		X	X	X	X	X	370
	X						400

	X		X				500

	X	X					600

	X	X	X				700

"X" INDICATES A CONNECTION TO W46.
W46 IS THE SOURCE JUMPER FOR THE VECTOR ADDRESS
JUMPERS ARE DAISY CHAINED.

MK-1341

Table 2-3 Device Address Selection

DPV11-XX (M8020) DEVICE ADDRESSING

MSB LSB

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

JUMPER NUMBER	W31	W30	W36	W33	W32	W39	W38	W37	W34	W35	DEVICE ADDRESS
										X	760010
									X		760020
									X	X	760030
								X			760040
								X		X	760050
								X	X		760060
								X	X	X	760070
							X				760100

						X					760200

						X	X				760300

					X						760400

					X		X				760500

					X	X					760600

					X	X	X				760700

				X							761000

			X								762000

			X	X							763000

		X									764000

"X" INDICATES A CONNECTION TO W29. W29 IS TIED TO GROUND. JUMPERS ARE DAISY CHAINED.

MK-1339

2.4 INSTALLATION

The DPV11 can be installed in any LSI-11 bus-compatible backplane such as H9270. LSI-11 configuring rules must be followed. Proceed with the installation as follows. For additional information refer to *PDP-11/03 User Manual* EK-LSI11-TM or *LSI-11 Installation Guide* EK-LSI11-IG.

1. Configure the address and vector jumpers at this time if they have not been previously done (Paragraph 2.3).

WARNING
Turn all power OFF.

2. Connect the female Berg connector on the BC26L-25 cable to J1 on the M8020 module † and plug the module into a dual LSI-11 bus slot of the backplane.

CAUTION

Insert and remove modules slowly and carefully to avoid snagging module components on the card guides.

3. Connect the H3259† turn-around connector to the EIA connection on the BC26L-25 cable. The jumper W1 on the H3259 turn-around connector must be removed.

4. Perform resistance checks from backplane pin AA2 (+5 V) to ground and from AD2 (+12 V) to ground to ensure that there are no shorts on the M8020 module or backplane.

5. Turn system power on.

6. Check the voltages to ensure that they are within the specified tolerances (Table 2-4). If voltages are not within specified tolerances, replace the associated regulator (H780 P.S.)

Table 2-4 Voltage Requirements

Voltage	Max.	Min.	Backplane Pin
+5 V	+5.25	+4.75	AA2
+12 V	12.75	+11.25	AD2

2.4.1 Verification of Hardware Operation

The M8020 module is now ready to be tested by running the CVDPV* diagnostic. Additional information on the DPV11 diagnostics is contained in Appendix A. Proceed as follows.

NOTE

The * represents the revision level of the diagnostic.

1. Load and run CVDPV*. Three consecutive error-free passes of this test is the minimum requirement for a successful run. If this cannot be achieved, check the following.

   Board seating
   Jumper connections
   Cable connection
   Test connector

   If a successful run is still unachievable, corrective maintenance is required.

2. Load and run the DEC/X11 System Exerciser configured to test the number of DPV11s in the system.

   Each DEC/X11 CXDPV module will test up to eight consecutively addressed DPV11s.

   CXDPV uses a software switch register. Refer to the *DEC/X11 Cross-Reference* (AS-F055C-MC) for switch register utilization.

† If a BC26L-25 cable and H3259 turn-around connector are not available, an on-board test connector (H3260) can be ordered separately. See Paragraph 2.5.

The DEC/X11 System Exerciser is designed to achieve maximum contention with all devices that make up the system configuration. It is within this environment that the CXDPV module runs. Its intent is to isolate DPV11s which adversely affect the system operation.

For information on configuring and running the DEC/X11 System Exerciser, refer to *DEC/X11 User Manual* (AS-F0503B-MC) and *DEC-X11 Cross Reference* (AS-F055C-MC).

2.4.2 Connection to External Equipment/Link Testing

The DPV11 is now ready for connection to external equipment.

If the DPV11 is being connected to a synchronous modem, remove the H3259 connector and install the EIA connection of the BC26L-25 cable into the connector on the modem.

Configure jumpers W1–W28 in accordance with operating requirements (Table 2-1).

Load and run DCLT (CVCLH*) if a full link is available. This will check the final configuration and isolate failures to the CPU, the communications link, or the modem.

If the connection to external equipment uses RS-422-A, the user must provide the cable and test support.

2.5 TEST CONNECTORS

The only test connector provided with the DPV11 is the H3259 turn-around connector (Figure 2-2). Table 2-5 and Figure 2-3 show the relationship between pin numbers, signal names and register bits when the H3259 is connected by means of the BC26L-26 cable to the M8020 module.

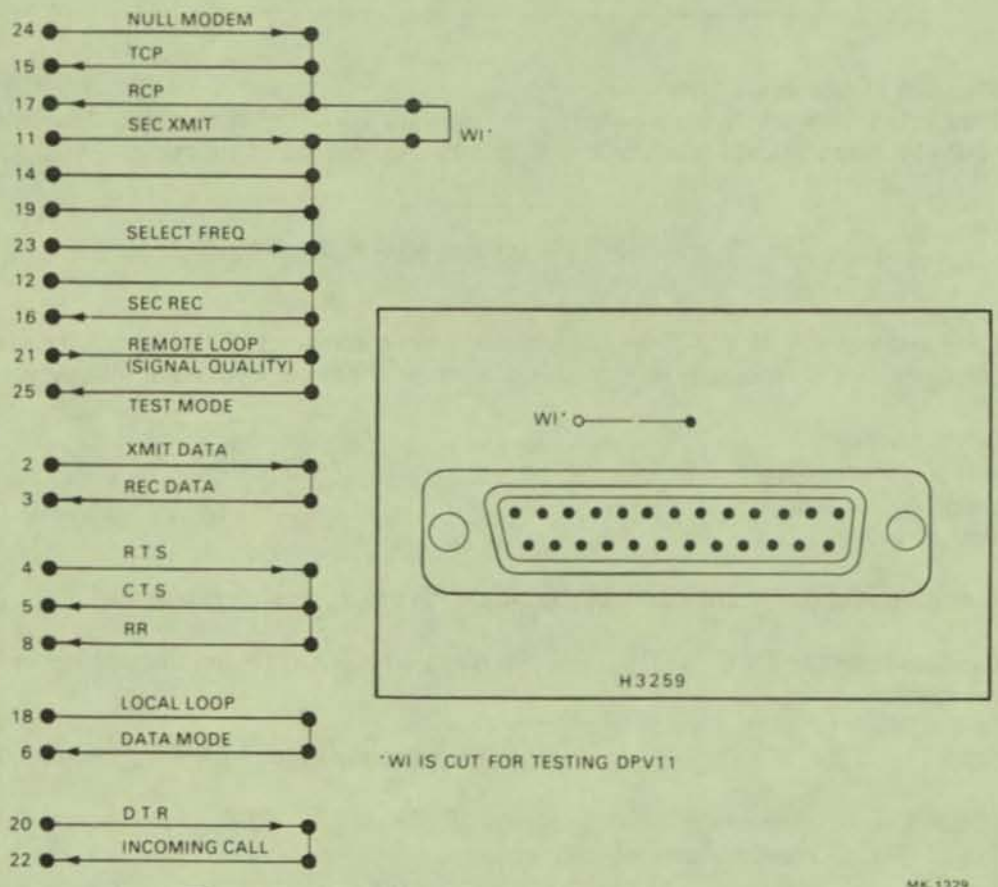


Figure 2-2 H3259 Turn-Around Test Connector

Table 2-5 H3259 Test Connections

From		To			
Signal Name	Pin No. H3259	Pin No. J1	Pin No. J1	Pin No. H3259	Signal Name
SEND DATA	2	F	J	3	RECEIVE DATA
REQUEST TO SEND (RTS) (RXCSR-2)	4	V	BB&T	5&8	CLEAR TO SEND (CTS)(RXCSR-13), RECEIVER READY (RR) (RXCSR-12)
LOCAL LOOPBACK (LL) (RXCSR-3)	18	U	Z	6	DATA MODE (DM) (RXCSR-9)
SELECT FREQ/REMOTE LOOPBACK (SF/RL) (RXCSR-0)	23/21	RR/MM	MM/C	21/25	SIGNAL QUALITY/ TEST MODE (SQ/TM) (PCSCR-5)
NULL MODEM	24	L	N&R	15&17	RCV CLOCK TX CLOCK
DATA TERMINAL READY (DTR) (RXCSR-1)	20	DD	X	22	INCOMING CALL (IC) (RXCSR-14)

The following accessories are available for interfacing and may be ordered separately.

- BC26L-X cable. Available in lengths of .3, 1.8, 2.4, 3.0, 3.6, 6.1, and 7.6 meters (1, 6, 8, 10, 12, 20 and 25 feet). When ordering, the dash number indicates the desired cable length in feet; e.g., BC26L-25 or BC26L-1.
- H3259 cable turn-around connector
- H856 Berg connector. Includes H856 Berg connector and 40 pins. Crimping tools are available from:

  Berg Electronics, Inc.
  New Cumberland, PA 17070

- H3260 on-board test connector (includes RS-422-A testing)

The H3260 on-board test connector (Figure 2-4) may be used to test the M8020 circuitry in its entirety. RS-422-A circuitry is not tested with the H3259 cable turn-around connector. The H3260 on-board test connector is shipped configured for testing RS-422-A. It may be configured to test RS-422-A or RS-423-A as follows.

RS-422-A	**RS-423-A**
W1–W2 out	W1–W2 installed
W3–W6 installed	W3–W6 out

The connector is installed into J1 with the jumper side up.

Since the H3260 on-board test connector does not test the cable, it is recommended that the DPV11 be tested with a turn-around connector at the modem end of the cable if possible.

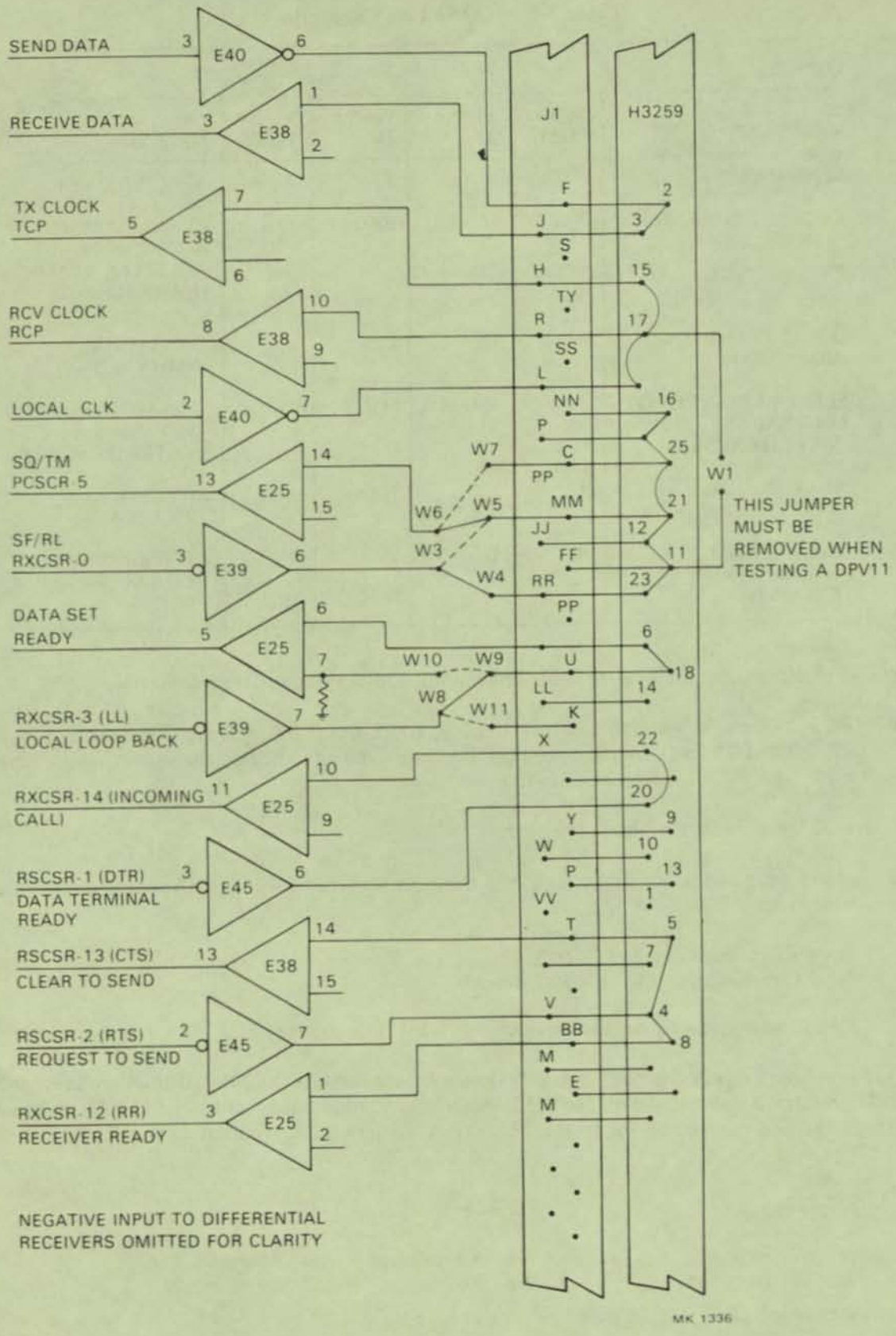


Figure 2-3 RS-423-A with H3259 Test Connector

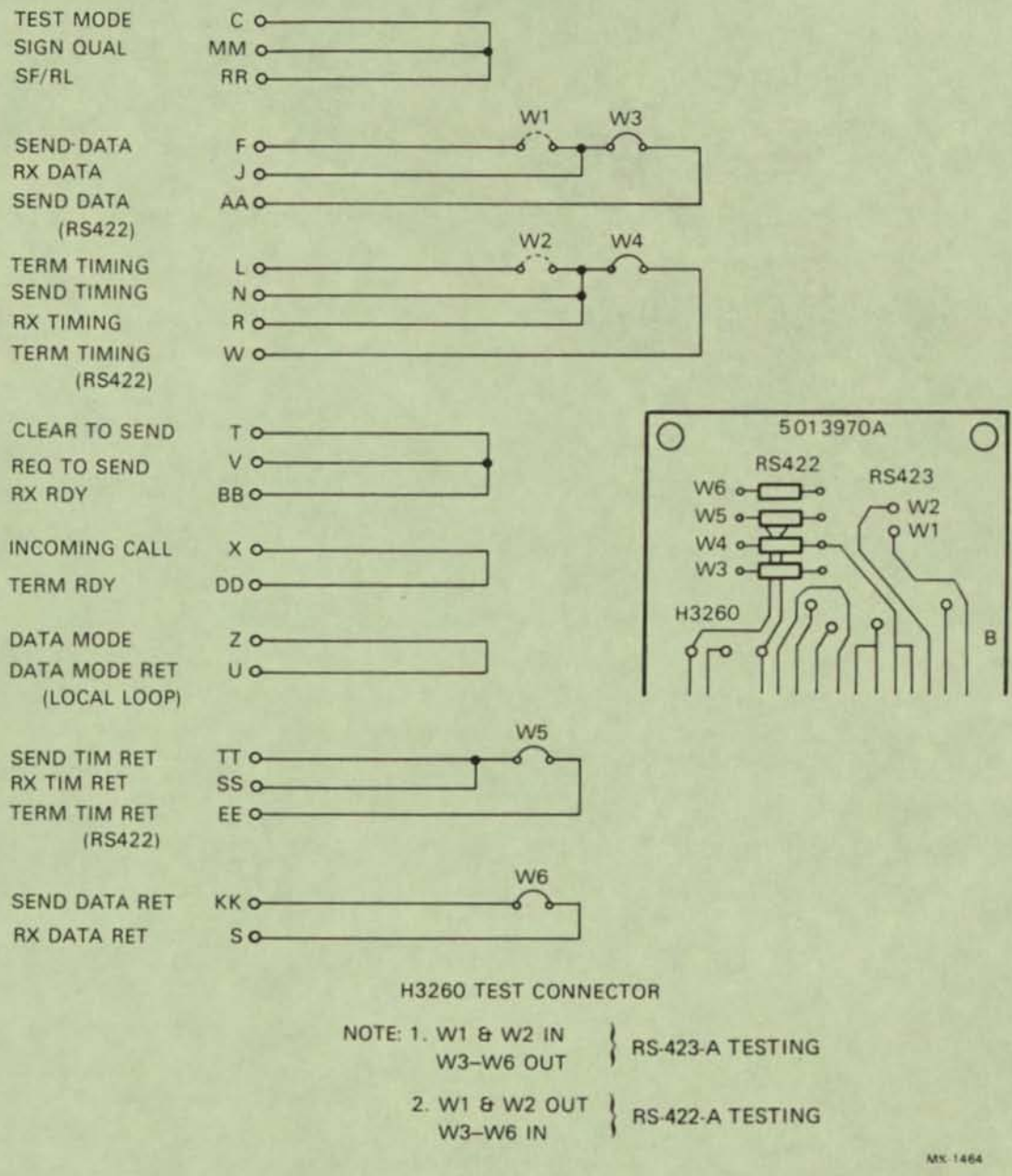


Figure 2-4 H3260 On-Board Test Connector

CHAPTER 3
REGISTER DESCRIPTIONS AND PROGRAMMING INFORMATION

3.1 INTRODUCTION

This chapter describes the bit assignments and programming considerations for the DPV11. Some typical start and receive sequences for both bit- and character-oriented protocols are included.

3.2 DPV11 REGISTERS AND DEVICE ADDRESSES

The five registers used in the DPV11 are shown in Table 3-1. Note that two of the registers (PCSAR and RDSR) have the same address. This does not constitute a conflict, however, because the PCSAR is a write-only register and the RDSR is a read-only register. These five registers occupy eight contiguous byte addresses which begin on a boundary where the low-order three bits are zero, and can be located anywhere between 160000_8 and 177776_8.

Table 3-1 DPV11 Registers

Register Name	Mnemonic	Address	Comments
Receive Control and Status	RXCSR	16xxx0	Word or byte* addressable. Read/write.
Receive Data and Status	RDSR**	16xxx2	Word or byte* addressable. Read-only.
Parameter Control Sync/Address	PCSAR**	16xxx2	Word or byte addressable. Write-only.†
Parameter Control and Character Length	PCSCR‡	16xxx4	Word or byte addressable. Read/write.
Transmit Data and Status	TDSR**	16xxx6	Word or byte addressable. Read/write.

* Reading either byte of these registers, clears data and certain status bits in other bytes. See Paragraphs 3.3.1 and 3.3.2.

** Registers contained within the USYNRT.

† It is not possible to do bit set or bit clear instructions on this register.

‡ The high byte of this register is internal to the USYNRT.

The DPV11 uses a universal-synchronous receiver/transmitter (USYNRT) chip which accounts for a large portion of the DPV11's functionality. The USYNRT provides complete serialization, deserialization and buffering of data to and from the modem.

Most of the DPV11 registers are internal to the USYNRT. Only the receiver control and status register (RXCSR) and the low byte of the parameter control and character length register (PCSCR) are external.

NOTE

When using the special space sequence function, all registers internal to the USYNRT must be written in byte mode.

3.3 REGISTER BIT ASSIGNMENTS

Bit assignments for the five DPV11 registers are shown in Figure 3-1. Paragraphs 3.3.1–3.3.5 provide a description of each register using a bit assignment illustration and an accompanying table with a detailed description of each bit.

3.3.1 Receive Control and Status Register (RXCSR) (Address 16xxx0)

Figure 3-2 shows the format for the receive control and status register (RXCSR). Table 3-2 is a detailed description of the register. This register is external to the USYNRT.

NOTE

The RXCSR can be read in either word or byte mode. However, reading either byte resets certain status bits in both bytes.

3.3.2 Receive Data and Status Register (RDSR) (Address 16xxx2)

Figure 3-3 show the format for the receive data and status register (RDSR). It is a read-only register and shares its address with the parameter control sync/address register (PCSAR) which is write-only. Table 3-3 is a detailed description of the RDSR.

NOTE

The RDSR can be read in either word or byte mode. However, reading either byte resets data and certain status bits in both bytes of this register as well as bits 7 and 10 of the RXCSR.

3.3.3 Parameter Control Sync/Address Register (PCSAR) (Address 16xxx2)

The parameter control sync/address register (PCSAR) is a write-only register which can be written in either byte or word mode. Figure 3-4 shows the format and Table 3-4 is a detailed description of the PCSAR. This register shares its address with the RDSR.

NOTE

Bit set (BIS) and bit clear (BIC) instructions cannot be executed on the PCSCR, since they execute using a read-modify-write sequence.

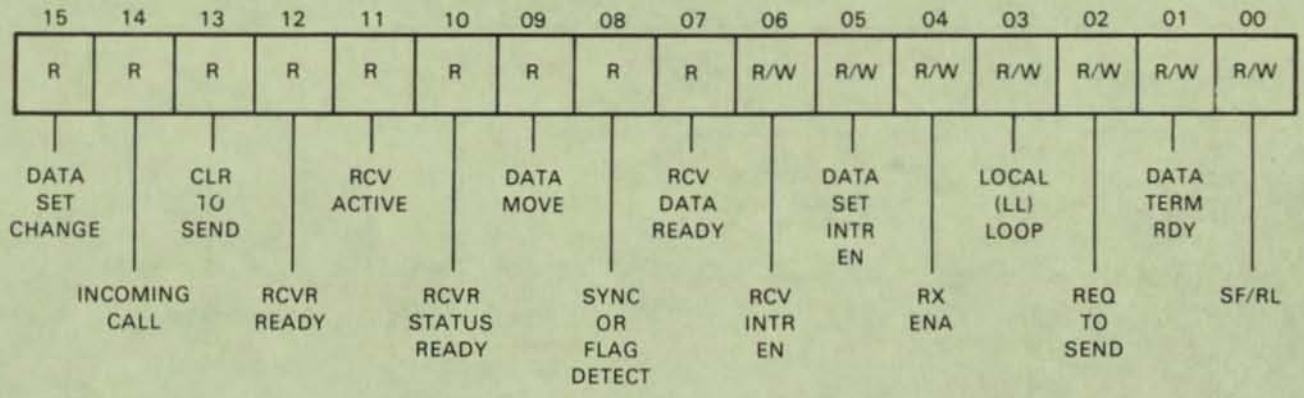
3.3.4 Parameter Control and Character Length Register (PCSCR) (Address 16xxx4)

The parameter control and character length register (PCSCR) can be read from or written into in either word or byte mode. The low byte of this register is external to the USYNRT and the high byte is internal. Figure 3-5 shows the format and Table 3-5 is a detailed description of the PCSCR.

3.3.5 Transmit Data and Status Register (TDSR) (Address 16xxx6)

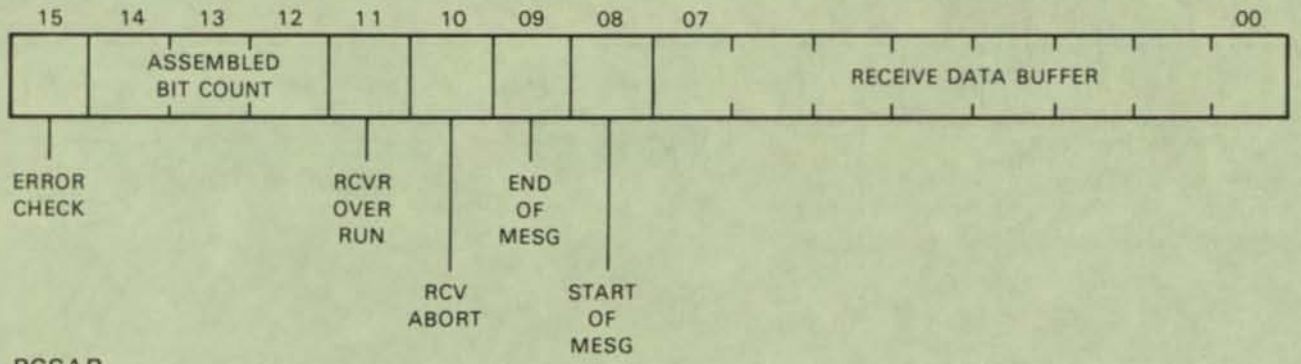
The format for the transmit data and status register (TDSR) is shown in Figure 3-6 and Table 3-6 is a detailed description. The TDSR is a read/write register which can be accessed in either word or byte mode with no restrictions. All bits can be read from or written into and are reset by Device Reset or Bus INIT except where noted.

RXCSR
16XXX0
READ/WRITE



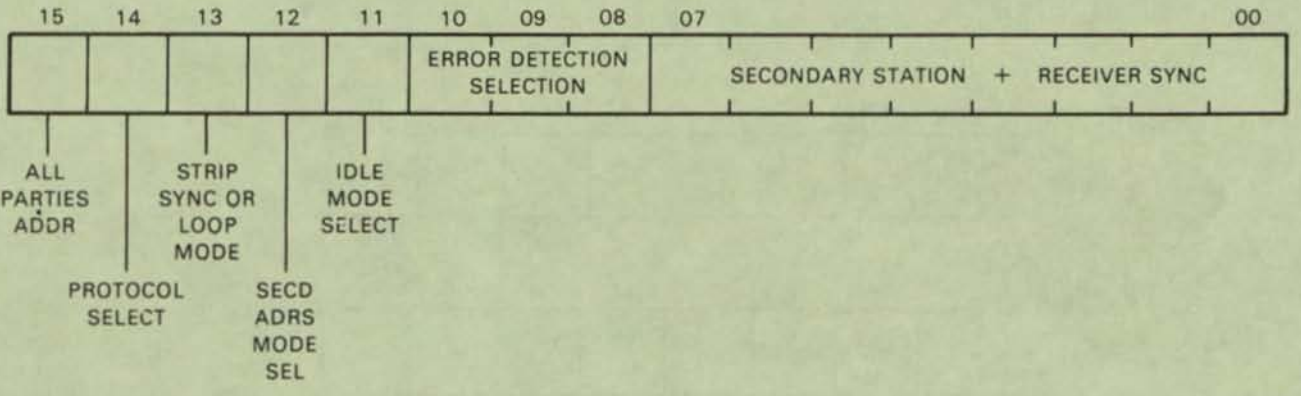
RDSR
16XXX2
READ ONLY

MK-1504



PCSAR
16XXX2
WRITE ONLY

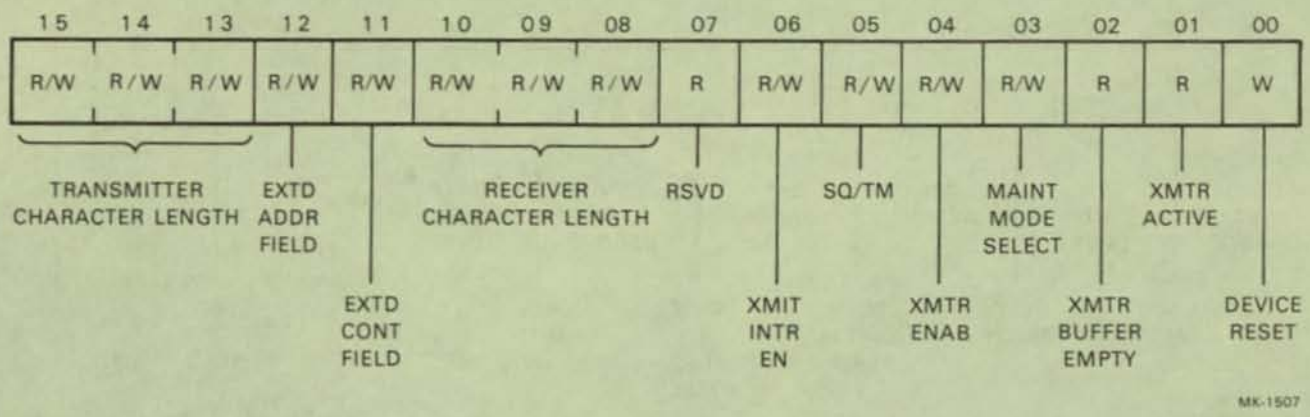
MK-1505



MK-1506

Figure 3-1 DPV11 Register Configurations and Bit Assignments (Sheet 1 of 2)

PCSCR
16XXX4
READ/WRITE



TDSR
16XXX6
READ/WRITE

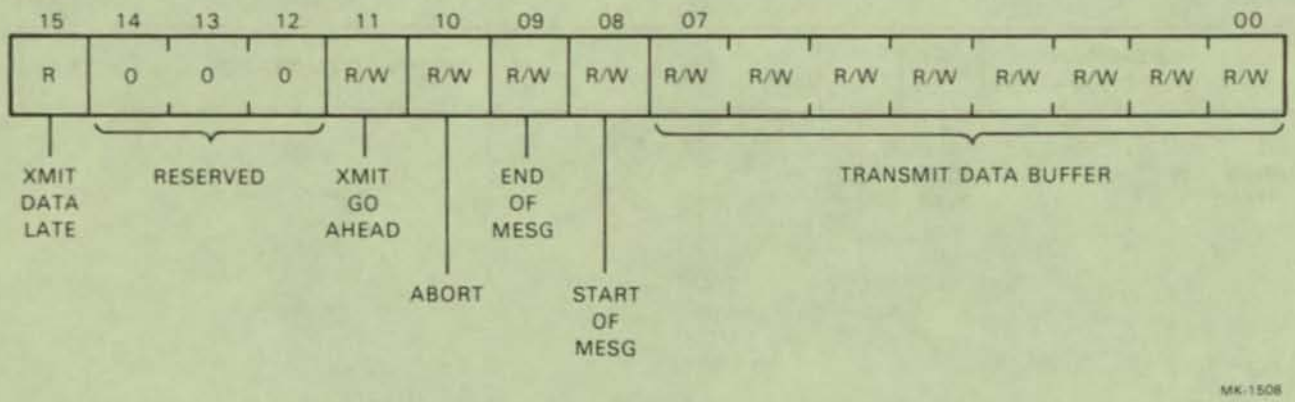


Figure 3-1 DPV11 Register Configurations and Bit Assignments (Sheet 2 of 2)

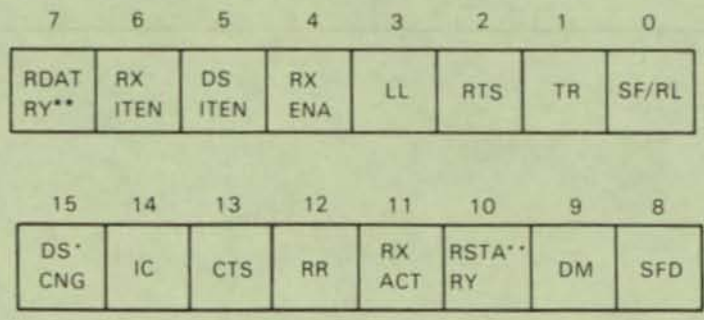


Figure 3-2 Receive Control and Status Register (RXCSR) Format

Table 3-2 Receive Control and Status Register (RXCSR) Bit Assignments

Bit	Name	Description
15	Data Set Change (DSCNG)	This bit is set when a transition occurs on any of the following modem control lines: Clear to Send Data Mode Receiver Ready Incoming Call Transition detectors for each of these four lines can be disabled by removing the associated jumper. Data Set Change is cleared by reading either byte of the RXCSR or by Device Reset or Bus INIT. Data Set Change causes a receive interrupt if DSITEN (bit 5) and RXITEN (bit 6) are both set.
14	Incoming Call (IC)	This bit reflects the state of the modem Incoming Call line. Any transition of this bit causes Data Set Change bit (bit 15) to be asserted unless the Incoming Call line is disabled by removing its jumper. This bit is read-only and cannot be cleared by software.
13	Clear to Send (CTS)	This bit reflects the state of the Clear to Send line of the modem. Any transition of this line causes Data Set Change (bit 15) to be set unless the jumper enabling the Clear to Send signal is removed. Clear to Send is a program read-only bit and cannot be cleared by software.
12	Receiver Ready (RR)	This bit is a direct reflection of modem Receiver Ready lead. It indicates that the modem is receiving a carrier signal. For external maintenance loopback, this signal must be high. If the line is open, RR is pulled high by the circuitry. Any transition of this bit causes Data Set Change (bit 15) to be asserted unless the jumper enabling the Receiver Ready signal is removed. Receiver Ready is a read-only bit and cannot be cleared by software.
11	Receiver Active (RXACT)	This bit is set when the USYNRT presents the first character of a message to the DPV11. It remains set until the receive data path of the USYNRT becomes idle. Receiver Active is cleared by any of the following conditions: a terminating control character is received in bit-oriented protocol mode; an off transition of Receiver Enable (RXENA) occurs; or Device Reset or Bus INIT is issued.

Table 3-2 Receive Control and Status Register (RXCSR) Bit Assignments (Cont)

Bit	Name	Description
		Receiver Active is a read-only bit which reflects the state of the USYNRT output pin 5.
10	Receiver Status Ready (RSTARY)	This bit indicates the availability of status information in the upper byte of the receive data and status register (RDSR). It is set when any of the following bits of the RDSR are set: Receiver End of Message (REOM); Receiver Overrun (RCV OVRUN); Receiver Abort or Go Ahead (RABORT); Error Check (ERRCHK) if VRC is selected. Receiver Status is cleared by any of the following conditions: reading either byte of the RDSR; clearing Receiver Enable (bit 4 of RXCSR); Device Reset, or Bus Init. When set, Receiver Status Ready causes a receive interrupt if Receive Interrupt Enable (bit 6) is also set. Receiver Status Ready is a read-only bit which reflects the state of USYNRT pin 7.
9	Data Mode (DM) (Data Set Ready)	This bit reflects the state of the Data Mode signal from the modem. When this bit is set it indicates that the modem is powered on and not in test, talk or dial mode. Any transition of this bit causes the Data Set Change bit (bit 15) to be asserted unless the Data Mode jumper has been removed. Data Mode is a read-only bit and cannot be cleared by software.
8	Sync or Flag Detect (SFD)	This bit is set for one clock time when a flag character is detected with bit-oriented protocols, or a sync character is detected with character-oriented protocols. SFD is a read-only bit which reflects the state of USYNRT pin 4.
7	Receive Data Ready (RDATRY)	This bit indicates that the USYNRT has assembled a data character and is ready to present it to the processor. If this bit becomes set while Receiver Interrupt Enable (bit 6) is set, a receive interrupt request will result. Receive Data Ready is reset when either byte of RDSR is read, Receiver Enable (bit 4) is cleared, or Device Reset or Bus INIT is issued. RDATRY is a read-only bit which reflectes the state of USYNRT pin 6.

Table 3-2 Receive Control and Status Register (RXCSR) Bit Assignments (Cont)

Bit	Name	Description
6	Receiver Interrupt Enable (RXITEN)	When set, this bit allows interrupt requests to be made to the receiver vector whenever RDATRY (bit 7) becomes set. The conditions which cause the interrupt request are the assertion of Receive Data Ready (bit 7), Receive Status Ready (bit 10), or Data Set Change (bit 15) if DSITEN (bit 5) is also set. RXITEN is a program read/write bit and is cleared by Device Reset or Bus INIT.
5	Data Set Interrupt Enable (DSITEN)	This bit, when set along with RXITEN, allows interrupt requests to be made to the receiver vector whenever Data Set Change (bit 15) becomes set. DSITEN is a program read/write bit and is cleared by Device Reset or Bus INIT.
4	Receiver Enable (RXENA)	This bit controls the operation of the receive section of the USYNRT. When this bit is set, the receive section of the USYNRT is enabled. When it is reset the receive section is disabled. In addition to disabling the receive section of the USYNRT, resetting bit 4 reinitializes all but two of the USYNRT receive registers. The two registers not reinitialized are the character length selection buffer and the parameter control register.
3	Local Loopback (LL)	Asserting this bit causes the modem connected to the DPV11 to establish a data loopback test condition. Clearing this bit restores normal modem operation. Local Loopback is program read/write and is cleared by Device Reset or Bus request to Send is program read/write and is cleared by Device Reset or Bus INIT.
2	Request to Send (RTS)	Setting this bit asserts the Request to Send signal at the modem interface. Request to Send is program read/write and is cleared by Device Reset or Bus INIT.
1	Terminal Ready (TR) (Data Terminal Ready)	When set, this bit asserts the Terminal Ready signal to the modem interface. For auto dial and manual call origination, it maintains the established call. For auto answer, it allows handshaking in response to a Ring signal.

Table 3-2 Receive Control and Status Register (RXCSR) Bit Assignments (Cont)

Bit	Name	Description
0	Select Frequency or Remote Loopback (SF/RL)	This bit can be wire-wrap jumpered to function as either select frequency or remote loopback. When jumpered as select frequency (W3 to W4), setting this bit selects the modem's higher frequency band for transmission to the line and the lower frequency band for reception from the line. The clear condition selects the lower frequency for transmission and the higher frequency for reception. When jumpered for remote loopback (W5 to W3), this bit, when asserted, causes the modem connected to the DPV11 to signal when a remote loopback test condition has been established in the remote modem. SF/RL is program read/write and is cleared by Device Reset or Bus INIT.

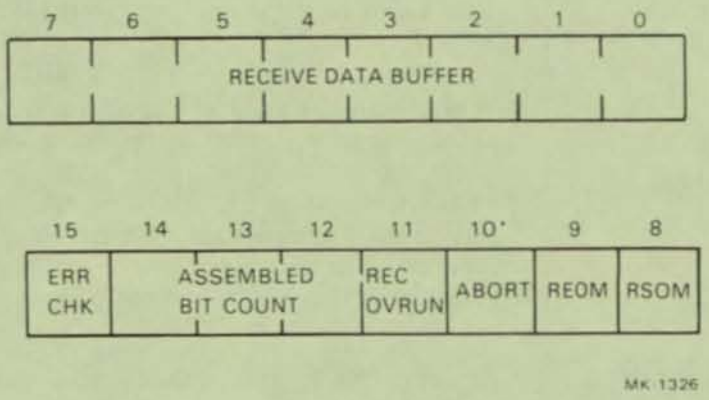


Figure 3-3 Receive Data and Status Register (RDSR) Format

Table 3-3 Receive Data and Status Register (RDSR) Bit Assignments

Bit	Name	Description
15	Error Check (ERR CHK)	This bit when set, indicates a possible error. It is used in conjunction with the error detection selection bits of the parameter control sync/address register (bits 8–10) to indicate either an error or an all zeros state of the CRC register. With bit-oriented protocols, ERR CHK indicates that a CRC error has occurred. It is set when the Receive End of Message bit (RDSR bit 9) is set. With character-oriented protocols ERR CHK is asserted with each data character if all zeros are in the CRC register. The processor must then determine if this indicates an error-free

Table 3-3 Receive Data and Status Register (RDSR) Bit Assignments (Cont)

Bit	Name	Description
		message or not. If VRC parity is selected, this bit is set for every character which has a parity error. ERR CHK is cleared by reading the RDSR, clearing RXENA (RXCSR bit 4), Device Reset or Bus INIT.
14–12	Assembled Bit Count (ABC)	Used only with bit-oriented protocols, these bits represent the number of valid bits in the last character of a message. They are all zeros unless the message ends on an unstated boundary. The bits are encoded to represent valid bits as shown below. **14 13 12 Number of Valid Bits** 0 0 0 All bits are valid 0 0 1 One valid bit 0 1 0 Two valid bits 0 1 1 Three valid bits 1 0 0 Four valid bits 1 0 1 Five valid bits 1 1 0 Six valid bits 1 1 1 Seven valid bits These bits are presented simultaneously with the last bits of data and are cleared by reading the RDSR or by resetting RXENA (bit 4 of RXCSR).
11	Receiver Overrun (RCV OVRUN)	This bit is used to indicate that an overrun situation has occurred. Overrun exists when the data buffer (bits 0–7 of RDSR) has not been serviced within one character time. As a general rule, the overrun is indicated when the last bit of the current character has been received into the shift register of the USYNRT and the data buffer is not yet available for a new character. Two factors exist which modify this general rule and apply only to bit-oriented protocols. The first factor is the number of bits inserted into the data stream for transparency. For each bit inserted during the formatting of the current character, the controller's maximum response time is increased by one clock cycle. The second factor is the result of termination of the current message. When this occurs, the data of the terminated message which is within the USYNRT is not overrunable. If an attempt is made to displace this data by the reception of a subsequent message, the data of the subsequent message is lost until the data of the prior message has been released.

Table 3-3 Receive Data and Status Register (RDSR) Bit Assignments (Cont)

Bit	Name	Description
10	Receiver Abort or Go Ahead (RABORT)	This bit is used only with bit-oriented protocols and indicates that either an abort character or a go-ahead character has been received. This is determined by the Loop Mode bit (PCSAR bit 13). If the Loop Mode bit is clear, RABORT indicates reception of an abort character. If the Loop Mode bit is set, RABORT indicates a go-ahead character has been received. The setting of RABORT causes Receiver Status Ready (bit 10 of RXCSR) to be set. RABORT is reset when the RDSR is read or when Receiver Enable (bit 4 of RXCSR) is reset. The abort character is defined to be seven or more contiguous one bits appearing in the data stream. Reception of this bit pattern when Loop Mode is clear causes the receive section of the USYNRT to stop receiving and set RSTARY (bit 10 of RXCSR). The abort character indicates abnormal termination of the current message. The go-ahead character is defined as a zero bit followed by seven consecutive one bits. This character is recognized as a normal terminating control character when the Loop Mode bit is set. If Loop Mode is cleared this character is interpreted as an abort character.
9	Receiver End of Message (REOM)	This bit is used only with bit-oriented protocols and is asserted if Receiver Active (bit 11 of RXCSR) is set and a message is terminated either normally or abnormally. When REOM becomes set, it sets RSTARY (bit 10 of RXCSR). REOM is cleared when RDSR is read or when Receive Enable (bit 4 of RXCSR) is reset.
8	Receiver Start of Message (RSOM)	Used only with bit-oriented protocols. This bit is presented to the processor along with the first data character of a message and is synchronized to the last received flag character. Setting of RSOM does not set RSTARY (RXCSR bit 10). RSOM is cleared by Device Reset, Bus INIT, resetting Receiver Enable (RXCSR bit 4), or the next transfer into the Receive Data buffer (low byte of RDSR).
7–0	Receive Data Buffer	The low byte of the RDSR is the Receive Data buffer. The serial data input to the USYNRT is assembled and transferred to the low byte of the RDSR for presentation to the processor. When the RDSR receives data, Receive Data Ready (bit 7 of RXCSR) becomes set to indicate that the RDSR has data to be picked up. If this data is not read within one character time, a data overrun occurs. The characters in the Receive Data buffer are right-justified with bit 0 being the least significant bit.

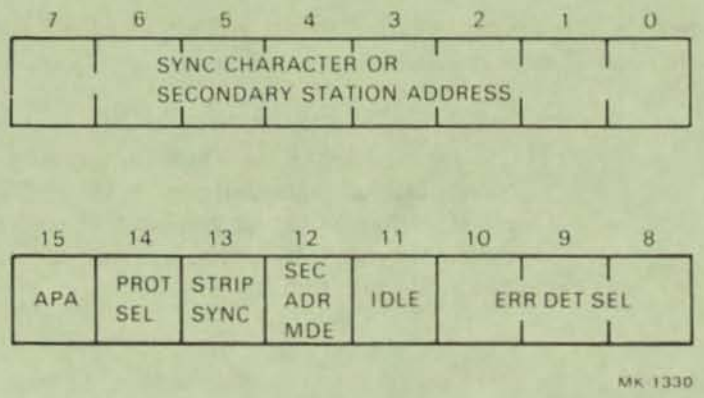


Figure 3-4 Parameter Control Sync/Address Register (PCSAR) Format

Table 3-4 Parameter Control Sync/Address Register (PCSAR) Bit Assignments

Bit	Name	Description
15	All Parties Addressed (APA)	This bit is set when automatic recognition of the All Parties Addressed character is desired. The All Parties Addressed character is eight bits of ones with necessary bit stuffing so as not to be confused with the abort character. Recognition of this character is done in the same way as the secondary station address (see bit 12 of this register) except that the broadcast address is essentially hardwired within the receive data path. The logic inspects the address character of each frame for the broadcast address. When the broadcast address is recognized, the USYNRT makes it available and sets Receiver Start of Message (bit 8 of RDSR). If the broadcast address is not recognized, one of two possible actions occurs. 1. If the Secondary Address Select mode bit (bit 12) is set, a test of the secondary station address is made. 2. If bit 12 is not set or the secondary station address is not recognized, the receive section of the USYNRT renews its search for synchronizing control characters.
14	Protocol Select (PROT SEL)	This bit is used to select between character- and byte count-oriented or bit-oriented protocols. It is set for character- and byte count-oriented protocols and reset for bit-oriented protocols.
13	Strip Sync or Loop Mode (STRIP SYNC)	This bit serves the following two functions. 1. Strip Sync (character-oriented protocols) – In character-oriented protocols, all sync characters after the initial synchronization are deleted from the message and not included in the CRC computation if this bit is set. If it is cleared, all sync characters remain in the message and are included in the CRC computation.

Table 3-4 Parameter Control Sync/Address Register (PCSAR) Bit Assignments (Cont)

Bit	Name	Description
		2. Loop Mode (bit-oriented protocols) – With bit-oriented protocols, this bit is used to control the method of termination. If it is set, either a flag or go-ahead character can cause a normal termination of a message. If it is cleared, only a flag character can cause a normal termination.
12	Secondary Address Mode (SEC ADR MDE)	This bit is used with bit-oriented protocols when automatic recognition of the secondary station address is desired. If it is set, the station address of the incoming message is compared with the address stored in the low byte of this register. Only messages prefixed with the correct secondary address are presented to the processor. If the addresses do not compare, the receive section of the USYNRT goes back to searching for flag or go-ahead characters. When SEC ADR MDE is cleared, the receive section of the USYNRT recognizes all incoming messages.
11	Idle Mode Select (IDLE)	This bit is used with both bit- and character-oriented protocols. With bit-oriented protocols, IDLE is used to select the type of control character issued when either Transmit Abort (bit 10 of TDSR) is set or a data underrun error occurs. If IDLE is set, flag characters are issued. If IDLE is clear, abort characters are issued. With character-oriented protocols, IDLE is used to control the method in which initial sync characters are transmitted and the action of the transmit section of the USYNRT when an underrun error occurs. IDLE is cleared to cause sync characters from the low byte of PCSAR to be transmitted. When IDLE is set, the transmit data output is held asserted during an underrun error and at the end of a message.
10–8	Error Detection Selection (ERR DEL SEL)	These bits are used to determine the type of error detection used on received and transmitted messages. In bit-oriented protocols, the selection is independent of character length. In character- and byte count-oriented protocols, CRC error detection is usable only with 8-bit character lengths. The maximum character length for VRC is seven. The bits are encoded as follows. **10 9 8 CRC Polynomial** 0 0 0 $x^{16}+x^{12}+x^5+1$ (CRC CCITT) (Both CRC data registers in the transmit and receive sections are set to all ones prior to the computation.) 0 0 1 $x^{16}+x^{12}+x^5+1$ (CRC CCITT) (Both CRC data registers set to all zeros.)

Table 3-4 Parameter Control Sync/Address Register (PCSAR) Bit Assignments (Cont)

Bit	Name	Description			
		0	1	0	Not used
		0	1	1	$x^{16}+x^{15}+x^2+1$ (CRC 16) (Both CRC registers set to all zeros.)
		1	0	0	Odd VRC Parity (A parity bit is attached to each transmitted character.) Should be used only in character-oriented protocols.
		1	0	1	Even VRC parity (Resembles odd VRC except that an even number of bits are generated.)
		1	1	0	Not used.
		1	1	1	All error detection is inhibited.
7–0	Sync Character or Secondary Address	The low byte of PCSAR is used as either the sync character for character-oriented protocols or as the secondary station address for bit-oriented protocols. The bits are right-justified with the least significant bit being bit 0.			

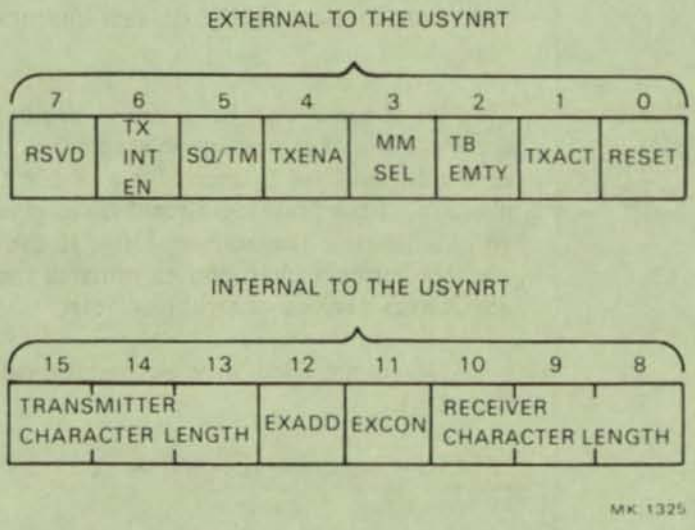


Figure 3-5 Parameter Control and Character Length Register (PCSCR) Format

Table 3-5 Parameter Control and Character Length Register (PCSCR) Bit Assignments

Bit	Name	Description
15–13	Transmitter Character Length	These bits can be read or written and are used to determine the length of the characters to be transmitted. They are encoded to set up character lengths as follows. **15 14 13 Character Length** 0 0 0 Eight bits per character 1 1 1 Seven bits per character 1 1 0 Six bits per character 1 0 1 Five bits per character (bit-oriented protocol only) 1 0 0 Four bits per character (bit-oriented protocol only) 0 1 1 Three bits per character (bit-oriented protocol only) 0 1 0 Two bits per character (bit-oriented protocol only) 0 0 1 One bit per character (bit-oriented protocol only) These bits can be changed while the transmitter is active, in which case the new character length is assumed at the completion of the current character. This field is set to a character length of eight by Device Reset or Bus INIT. When VRC error detection is selected, the default character length is eight bits plus parity.
12	Extended Address Field (EXADD)	This bit is used with bit-oriented protocols and affects the address portion of a message in receiver operations. When it is set, each address byte is tested for a one in the least significant bit position. If the least significant bit is zero, the next character is an extension of the address field. If the least significant bit is one, the current character terminates the address field and the next character is a control character. EXADD is not used with Secondary Address Mode (bit 12 of PCSAR). EXADD is read/write and is reset by Device Reset or Bus INIT.
11	Extended Control Field (EXCON)	This bit is used with bit-oriented protocols and affects the control character of a message in receiver operations. When EX-

Table 3-5 Parameter Control and Character Length Register (PCSCR) Bit Assignments (Cont)

Bit	Name	Description
		CON is set it extends the control field from one 8-bit byte to two 8-bit bytes. EXCON is not used with Secondary Address Mode (bit 12 of PCSAR) EXCON is read/write and is reset by Device Reset or Bus INIT.
10–8	Receiver Character Length	These bits are used to determine the length of the characters to be received. They are encoded to set up character lengths as follows. **10 9 8 Character Length** 0 0 0 Eight bits per character 1 1 1 Seven bits per character 1 1 0 Six bits per character 1 0 1 Five bits per character 1 0 0 Four bits per character (bit-oriented protocols only) 0 1 1 Three bits per character (bit-oriented protocols only) 0 1 0 Two bits per character (bit-oriented protocols only) 0 0 1 One bit per character (bit-oriented protocols only)
7	Reserved	Not used by the DPV11
6	Transmit Interrupt Enable (TXINTEN)	When set, this bit allows a transmitter interrupt request to be made to the transmitter vector when Transmit Buffer Empty (TBEMTY) is asserted. Transmit Interrupt Enable (TXINTEN) is read/write and is cleared by Device Reset or Bus INIT.
5	Signal Quality or Test Mode (SQ/TM)	This bit can be wire-wrap jumpered to function as either Signal Quality or Test Mode. When jumpered for signal quality (W5 to W6), this bit reflects the state of the signal quality line from the modem. When asserted, it indicates that there is a low probability of errors in the received data. When clear it indicates that there is a high probability of errors in the received data.

Table 3-5 Parameter Control and Character Length Register (PCSCR) Bit Assignments (Cont)

Bit	Name	Description
		When jumpered for the test mode (W6 to W7), this bit indicates that the modem has been placed in a test condition when asserted. The modem test condition could be established by asserting Local Loopback (bit 3 of RXCSR), Remote Loopback (bit 0 of RXCSR) or other means external to the DPV11. When SQ/TM is clear, it indicates that the modem is not in test mode and is available for normal operation. SQ/TM is program read-only and cannot be cleared by software.
4	Transmitter Enable (TXENA)	This bit must be set to initiate the transmission of data or control information. When this bit is cleared, the transmitter will revert back to the mark state once all indicated sequences have been completed. TXENA should be cleared after the last data character has been loaded into the transmit data and status register (TDSR). Transmit End of Message (bit 9 of TDSR) should be asserted when TXENA is reset (if it is to be asserted at all) and remain asserted until the transmitter enters the idle mode. TXENA is connected directly to USYNRT pin 37. It is a read/write bit and is reset by Device Reset or Bus INIT.
3	Maintenance Mode Select (MM SEL)	When this bit is asserted, it causes the USYNRT's serial output to be internally connected to the USYNRT's serial input. The serial send data output line from the interface is asserted and the receive data serial input is disabled. Send timing and receive timing to the USYNRT are disabled and replaced with a clock signal generated on the interface. The clock rate is either 49.152K b/s or 1.9661K b/s depending on the position of a jumper on the interface board. Maintenance mode allows diagnostics to run in loopback without disconnecting the modem cable. MM SEL is a read/write bit and is cleared by Device Reset or Bus INIT. When it is cleared, the interface is set for normal operation.
2	Transmitter Buffer Empty (TBEMTY)	This bit is asserted when the transmit data and status register (TDSR) is available for new data or control information. It is also set after a Device Reset or Bus INIT. The TDSR should be loaded only in response to TBEMTY being set. When the TDSR is written into, TBEMTY is cleared. If TBEMTY becomes set while Transmit Interrupt Enable (bit 6 of PCSCR) is set, a transmit interrupt request results. TBEMTY reflects the state of USYNRT pin 35.

Table 3-5 Parameter Control and Character Length Register (PCSCR) Bit Assignments (Cont)

Bit	Name	Description
1	Transmitter Active (TXACT)	This bit indicates the state of the transmit section of the USYNRT. It becomes set when the first character of data or control information is transmitted. TXACT is cleared when the transmitter has nothing to send or when Device Reset or Bus INIT is issued. TXACT reflects the state of USYNRT pin 34.
0	Device Reset (RESET)	When a one is written to this bit all components of the interface are initialized. It performs the same function as Bus INIT with respect to this interface. Modem Status (Data Mode, Clear to Send, Receiver Ready, Incoming Call, Signal Quality or Test Mode) is not affected. RESET is write-only; it cannot be read by software.

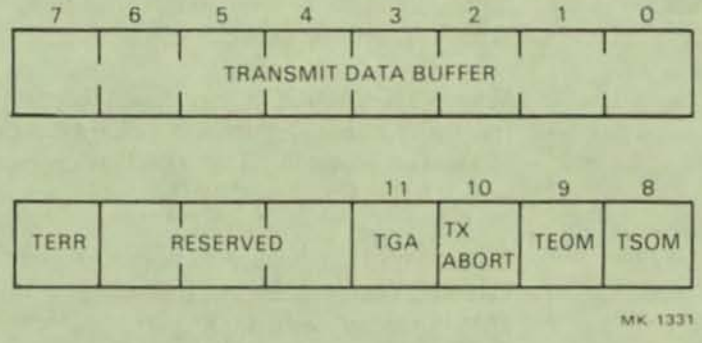


Figure 3-6 Transmit Data and Status Register (TDSR) Format

Table 3-6 Transmit Data and Status Register (TDSR) Bit Assignments

Bit	Name	Description
15	Transmitter Error (TERR)	This is a read-only bit which becomes asserted when the Transmitter Buffer Empty (TBEMTY) indication has not been serviced for more than one character time. When TERR occurs in bit-oriented protocols, the transmit section of the USYNRT generates an abort or flag character based on the state of the IDLE bit (PCSAR bit 11). If IDLE is set, a flag character is sent. If it is reset, an abort character is sent. When TERR occurs in character-oriented protocols, the state of the IDLE bit again determines the result. If IDLE is set, the transmit serial output is held in the MARK condition. If it is cleared, a sync character is transmitted.

Table 3-6 Transmit Data and Status Register (TDSR) Bit Assignments (Cont)

Bit	Name	Description
		TERR is cleared when TSOM (TDSR bit 8) becomes set or by Device Reset or Bus INIT. Clearing Transmitter Enable (PCSCR bit 4) does not clear TERR and TERR is not set with Transmit End of Message.
14–12	Reserved	Not used by the DPV11
11	Transmit Go Ahead (TGA)	This bit, when asserted, modifies the bit pattern of the control character initiated by either Transmit Start of Message (TSOM) or Transmit End of Message (TEOM). TSOM or TEOM normally causes a flag character to be sent. If TGA is set, a go-ahead character is sent in place of the flag character. TGA is only used with bit-oriented protocols.
10	Transmit Abort (TXABORT)	This bit is used only with bit-oriented protocols to abnormally terminate a message or to transmit filler information used to establish data link timing. When TXABORT is asserted, the transmitter automatically transmits either flag or abort characters depending on the state of the IDLE mode bit. If IDLE is cleared, abort characters are sent. If IDLE is set, flag characters are sent.
9	Transmit End of Message (TEOM)	This control bit is used to normally terminate a message in bit-oriented protocol. It also terminates a message in character-oriented protocols when CRC error detection is used. As a secondary function, it is used in conjunction with the Transmit Start of Message (TSOM) bit to transmit a SPACE SEQUENCE. Refer to the TSOM bit description (bit 8 of this register) for information regarding this sequence. With bit-oriented protocols, asserting this bit causes the CRC information to be transmitted, if CRC is enabled, followed by flag or go-ahead characters depending on the state of the Transmit Go Ahead (TGA) bit. See bit 11 of this register. With character-oriented protocols, asserting this bit causes CRC information, if CRC is enabled, to be transmitted followed by either sync characters or a MARK condition depending on the state of the IDLE bit. If IDLE is cleared, sync characters are transmitted. The character following the CRC information is repeated until the transmitter is disabled or the TEOM bit is cleared. A subsequent message may be initiated while the transmit section of the USYNRT is active. This is accomplished by clearing the TEOM bit and supplying new message data without setting

Table 3-6 Transmit Data and Status Register (TDSR) Bit Assignments (Cont)

Bit	Name	Description
		the Transmit Start Of Message bit. However, the CRC character for the prior message must have completed transmission.
8	Transmit Start of Message (TSOM)	This bit is used with either bit- or character-oriented protocols. As long as it remains asserted, flag characters (bit-oriented protocols) or sync characters (character-oriented protocols) are transmitted. With bit-oriented protocols, a space sequence (byte mode only) of 16 zero bits can be transmitted by asserting TSOM and TEOM simultaneously provided the transmitter is in the idle state and Transmit Enable is cleared. This should not be done during the transfer of data, and must only be done in byte mode. **NOTE** **When using the special space sequence function, all registers internal to the USYNRT must be written in byte mode.** Normally at the completion of each sync, flag, go-ahead or Abort character, the TBEMTY indication is asserted. This allows the software to count the number of transmitted characters. In certain applications, the software may elect to ignore the service of the Transmitter Buffer Empty (TBEMTY) indication. Normally during data transfers, this would cause a transmit data late error. The TSOM bit asserted suppresses this error and provides the necessary synchronization to automatically transmit another flag, go-ahead or sync character.
7–0	Transmit Data Buffer	Data from the processor to be transmitted on the serial output line is loaded into this byte of the TDSR when Transmitter Buffer Empty (TBEMTY) is asserted. If the transmitter buffer is not loaded within one character time, an underrun error occurs. The characters are right-justified, with bit 0 being the least significant bit.

3.4 DATA TRANSFERS

Paragraphs 3.4.1 and 3.4.2 discuss receive and transmit data transfers as they relate to the system software.

3.4.1 Receive Data

Serial data to be presented to the DPV11 from the modem enters the receiver circuit and is presented to the USYNRT. Recognition by the USYNRT of a control character initiates the transfer. When a transfer has been initiated, a character is assembled by the USYNRT and then placed in the low byte of the receive data and status register (RDSR) when it is available. If the RDSR is not available, the transfer is delayed until the previous character has been serviced. This must take place before the next character is fully assembled or an overrun error exists. Refer to the description of bit 11 in Table 3-3 for more details on Receiver Overrun.

Servicing of the RDSR is the responsibility of the system software in response to the Receive Data Ready (RDATRY) signal. This signal is asserted when a character has been transferred to the RDSR. The setting of RDATRY would also cause a receive interrupt request if Receive Interrupt Enable (RXITEN) is set. The software's response to RDATRY is to read the contents of the RDSR. At the completion of this operation, the new information is loaded into the RDSR and RDATRY is reasserted. This operation continues until terminated by some control character. The upper byte of the RDSR contains status and error indications which the software can also read.

The DPV11 will handle data in bit-, byte count- or character-oriented protocols.

With bit-oriented protocol, only flag characters are used to initiate the transfer of a message. Information inserted into the data stream for transparency or control is deleted before it is presented to the RDSR. This means that only data characters are available to the software. The first two characters of every message or frame are defined to be 8-bit characters and the USYNRT will handle them as such regardless of the programmed character length. All subsequent data is formatted in the selected character length. When CRC error detection is selected, the received CRC check characters are not presented to the software, but the error indication will be presented if an error has been detected.

If the secondary address mode is implemented, the first received data character must be the selected address. If this is not the case, the USYNRT will renew its search for flag or go-ahead characters. Refer to the description of bit 12 of the PCSAR in Table 3-4.

With byte count- or character-oriented protocols, two consecutive sync characters are required to synchronize the transfer of data. The sync characters used in the message must be the same as the sync character loaded by the software into the low byte of the parameter control sync/address register (PCSAR). If leading sync characters subsequent to the initial two syncs are to be deleted from the data stream, the Strip Sync bit (bit 13) must also be set in the upper byte of the PCSAR. The character length of the data to be received should also be set in bits 8, 9, and 10 of the parameter control and character length register (PCSCR). Sync characters and data must have the same character length and only characters of the selected length will be presented to the receive buffer. Sync characters following the initial two will be presented to the buffer and included in the CRC computation unless the Strip Sync bit is set. If vertical redundancy check (VRC) parity checking is selected, the parity bit itself is deleted from the character before it is presented to the buffer.

3.4.2 Transmit Data

System software loads information to be transmitted to the modem into the transmit data and status register (TDSR). This does not ordinarily include error detection or control character information. Loading of the TDSR occurs in response to the Transmitter Buffer Empty (TBEMTY) signal from the USYNRT. The character length of information to be transmitted is established by the software when it loads the transmit character length register (bits 13, 14, and 15 of the PCSCR). The default length of eight is assigned when the transmit character length register equals zero. The length of characters presented to the TDSR should not exceed the assigned character length. When the information in the TDSR is transmitted, the TBEMTY signal is again asserted to request another character. The setting of TBEMTY also causes a transmit interrupt request if Transmit Interrupt Enable is set.

Byte count- or character-oriented protocols require the transmission of synchronizing information normally referred to as sync characters. The sync characters can be transmitted when Transmit Start of Message (TDSR bit 8) is set. This happens in one of two ways depending on the state of the IDLE bit (PCSAR bit 11). When the IDLE bit is cleared, the sync character is taken directly from the common sync register (PCSAR bits 7–0). The sync register would have been previously loaded by the software. If the IDLE bit is set, the sync character must be loaded into the TDSR by the software when it is to be transmitted. If multiple sync characters are to be transmitted, the TDSR must only be loaded with the first one of the sequence. This character will be transmitted until data information is loaded into the TDSR. The TBEMTY signal is asserted at the end of each sync character but the TSOM signal allows it to be ignored without causing a data late error.

With bit-oriented protocols, the USYNRT automatically generates control characters as initiated by the software and inserts necessary information into the data stream to maintain transparency.

Typical programming examples in bit- and byte count-oriented protocols appear in Appendix D.

3.5 INTERRUPT VECTORS

The DPV11 generates two vector addresses, one for receive data and modem control and the other for transmit data.

The receive and modem control interrupt has priority over the transmit interrupt and is enabled by setting bit 6 (RXITEN) of the receiver control and status register (RXCSR).

If bit 6 of the RXCSR is set, a receiver interrupt may occur when any one of the following signals is asserted.

- Receive Data Ready (RDATRY)
- Receive Status Ready (RSTARY)
- Data Set Change (DAT SET CH)

The signal DAT SET CH only causes an interrupt if bit 5 (DSITEN) of the RXCSR is also set.

It is possible that a data set change interrupt could be pending while a receiver interrupt is being serviced, or the opposite could be true. In either case, the hardware ensures that both interrupt requests are recognized.

NOTE

The modem status change circuit interprets any pulse of two microseconds or greater duration as a data set change. This ensures that all legitimate transitions of modem status will be detected. However, on a poor line, noise may be interpreted as a data set change. Software written for the DPV11 must account for this possibility.

A transmitter interrupt request occurs if Transmit Interrupt Enable (TXINTEN) is set when Transmit Buffer Empty (TBEMTY) becomes asserted.

APPENDIX A
DIAGNOSTIC SUPERVISOR SUMMARY

A.1 INTRODUCTION

The PDP-11 diagnostic supervisor is a software package that performs the following functions.

- Provides run-time support for diagnostic programs running on a PDP-11 in stand-alone mode
- Provides a consistent operator interface to load and run a single diagnostic program or a script of programs
- Provides a common programmer interface for diagnostic development
- Imposes a common structure upon diagnostic programs
- Guarantees compatibility with various load systems such as APT, ACT, SLIDE, XXDP+, ABS Loader
- Performs nondiagnostic functions for programs, such as console I/O, data conversion, test sequencing, program options

A.2 VERSIONS OF THE DIAGNOSTIC SUPERVISOR

File Name	Environment
HSAA **.SYS	XXDP+
HSAB **.SYS	APT
HSAC **.SYS	ACT/SLIDE
HSAD **.SYS	Paper Tape (Absolute Loader)

In the above file names, "**" stands for revision and patch level, such as "A0".

A.3 LOADING AND RUNNING A SUPERVISOR DIAGNOSTIC

A supervisor-compatible* diagnostic program may be loaded and started in the normal way, using any of the supported load systems. Using XXDP+ for example, the program CVDPVA.BIN is loaded and started by typing .R CVDPVA.

The diagnostic and the supervisor will automatically be loaded as shown in Figure A-1 and the program started. The program types the following message.

```
DRS LOADED
DIAG.RUN-TIME SERVICES
CVDPV-A-0
```

* To determine if diagnostics are supervisor-compatible, use the List command under the Setup utility (see Paragraph A.5.).

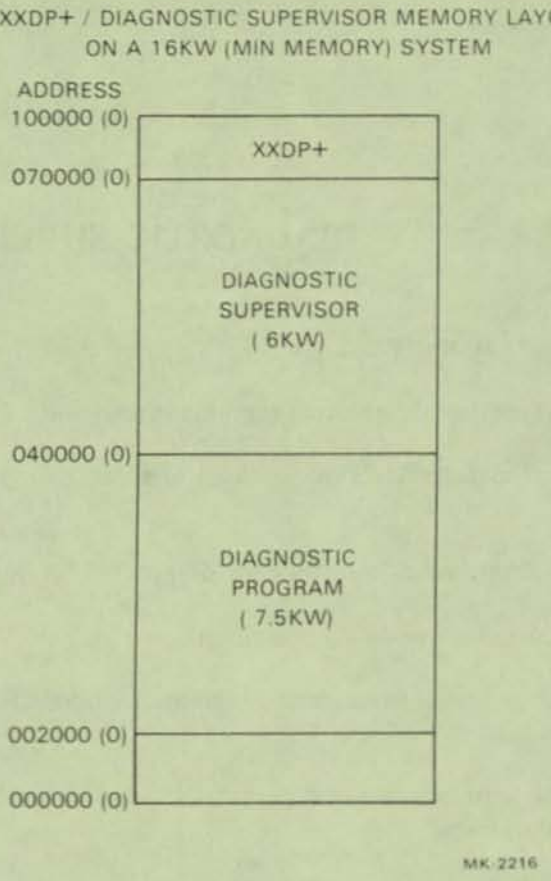


Figure A-1 Typical XXDP+/Diagnostic Supervisor Memory Layout

```
DIAGNOSTIC TESTS
UNIT IS DPV11
DR>
```

DR> is the prompt for the diagnostic supervisor routine. At this point a supervisor command must be entered (the supervisor commands are listed in Paragraph A.4).

Five Steps to Run a Supervisor Diagnostic

1. Enter Start command.

   When the prompt DR> is issued, type:

   STA/PASS:1/FLAGS:HOE <CR>

   The switches and flags are optional.

2. Enter number of units to be tested.

   The program responds to the Start command with:

   # UNITS?

   At this point enter the number of devices to be tested.

3. Answer hardware parameter questions.

   After the number of devices to be tested has been entered, the program responds by asking a number of hardware questions. The answers to these questions are used to build hardware parameter tables in memory. A series of questions is posed for each device to be tested. A "Hardware P-Table" is built for each device.

4. Answer software parameter questions.

   When all the "Hardware P-Tables" are built, the program responds with:

   CHANGE SW?

   If other than the default parameters are desired for the software, type Y. If the default parameters are desired, type N.

   If you type Y, a series of software questions will be asked and the answers to these will be entered into the "Software P-Table" in memory. The software questions will be asked only once, regardless of the number of units to be tested.

5. Diagnostic execution.

   After the software questions have been answered, the diagnostic begins to run.

   What happens next is determined by the switch options selected with the Start command, or errors occurring during execution of the diagnostic.

A.4 SUPERVISOR COMMANDS

The supervisor commands that may be issued in response to the DR> prompt are as follows.

- Start – Starts a diagnostic program.
- Restart – When a diagnostic has stopped and control is given back to the supervisor, this command restarts the program from the beginning.
- Continue – Allows a diagnostic to continue running from where it was stopped.
- Proceed – Causes the diagnostic to resume with the next test after the one in which it halted.
- Exit – Transfers control to the XXDP+ monitor.
- Drop – Drops units specified until an Add or Start command is given.
- Add – Adds units specified. These units must have been previously dropped.
- Print – Prints out statistics if available.
- Display – Displays P-Tables.
- Flags – Used to change flags.
- ZFLAGS – Clears flags.

All of the supervisor commands except Exit, Print, Flags, and ZFLAGS can be used with switch options.

A.4.1 Command Switches

Switch options may be used with most supervisor commands. The available switches and their function are as follows.

- ./TESTS: – Used to specify the tests to be run (the default is all tests). An example of the tests switch used with the Start command to run tests 1 through 5, 19, and 34 through 38 would be:

  DR> START/TESTS : 1-5 : 19 : 34-38 <CR>

- ./PASS: – Used to specify the number of passes for the diagnostic to run. For example:

  DR> START/PASS : 1

  In this example, the diagnostic would complete one pass and give control back to the supervisor.

- ./EOP: – Used to specify how many passes of the diagnostic will occur before the end of pass message is printed (the default is one).

- ./UNITS: – Used to specify the units to be run. This switch is valid only if N was entered in response to the CHANGE HW? question.

- ./FLAGS: – Used to check for conditions and modify program execution accordingly. The conditions checked for are as follows.

  :HOE –Halt an error (transfers control back to the supervisor)

  :LOE – Loop on error

  :IER – Inhibit error reports

  :IBE – Inhibit basic error information

  :IXE – Inhibit extended error information

  :PRI – Print errors on line printer

  :PNT – Print the number of the test being executed prior to execution

  :BOE – Ring bell on error

  :UAM – Run in unattended mode, bypass manual intervention tests

  :ISR – Inhibit statistical reports

  :IOU – Inhibit dropping of units by program

A.4.2 Control/Escape Characters Supported

The keyboard functions supported by the diagnostic supervisor are as follows.

- CONTROL C (↑C) – Returns control to the supervisor. The DR> prompt would be typed in response to CONTROL C. This function can be typed at any time.

- CONTROL Z (↑Z) – Used during hardware or software dialogue to terminate the dialogue and select default values.
- CONTROL O (↑O) – Disables all printouts. This is valid only during a printout.
- CONTROL S (↑S) – Used during a printout to temporarily freeze the printout.
- CONTROL Q (↑Q) – Resumes a printout after a CONTROL S.

A.5 THE SETUP UTILITY

Setup is a utility program that allows the operator to create parameters for a supervisor diagnostic prior to execution. This is valid for either XXDP+ or ACT/SLIDE environments. Setup asks the hardware and software questions and builds the P-Tables.

The following commands are available under Setup.

List – list supervisor diagnostics
Setup – create P-Tables
Exit – return control to the supervisor

The format for the List command is:

LIST DDN:FILE.EXT

Its function is to type the file name and creation date of the file specified if it is a revision C or later supervisor diagnostic. If no file name is given, all revision C or later supervisor diagnostics are listed. The default for the device is the system device, and wild cards are accepted.

The format for the Setup command is:

SETUP DDN:FILE.EXT=DDN:FILE.EXT

It reads the input file specified and prompts the operator for information to build P-Tables. An output file is created to run in the environment specified. File names for the output and input files may be the same. The output and input device may be the same. The default for the device is the system device and wild cards are not accepted.

APPENDIX B
USYNRT DESCRIPTION

5025 Universal Synchronous Receiver/Transmitter (USYNRT)
The data paths of the USYNRT provide complete serialization, deserialization and buffering. Output signals are provided to the USYNRT controller to indicate the state of the data paths, the command fields or recognition of extended address fields. These tasks must be performed by the USYNRT controller.

The USYNRT is a 40-pin dual-in-line package (DIP). Figure B-1 is a terminal connection (identification) diagram.

Data port bits DP07:DP00 are dedicated to service four 8-bit wide registers. Bits DP15:DP08 service either control information or status registers. The PCSCR register is reserved. (See Figure B-2.)

Purchase Specification 2112517-0-0 provides a detailed description of the 5025 USYNRT.

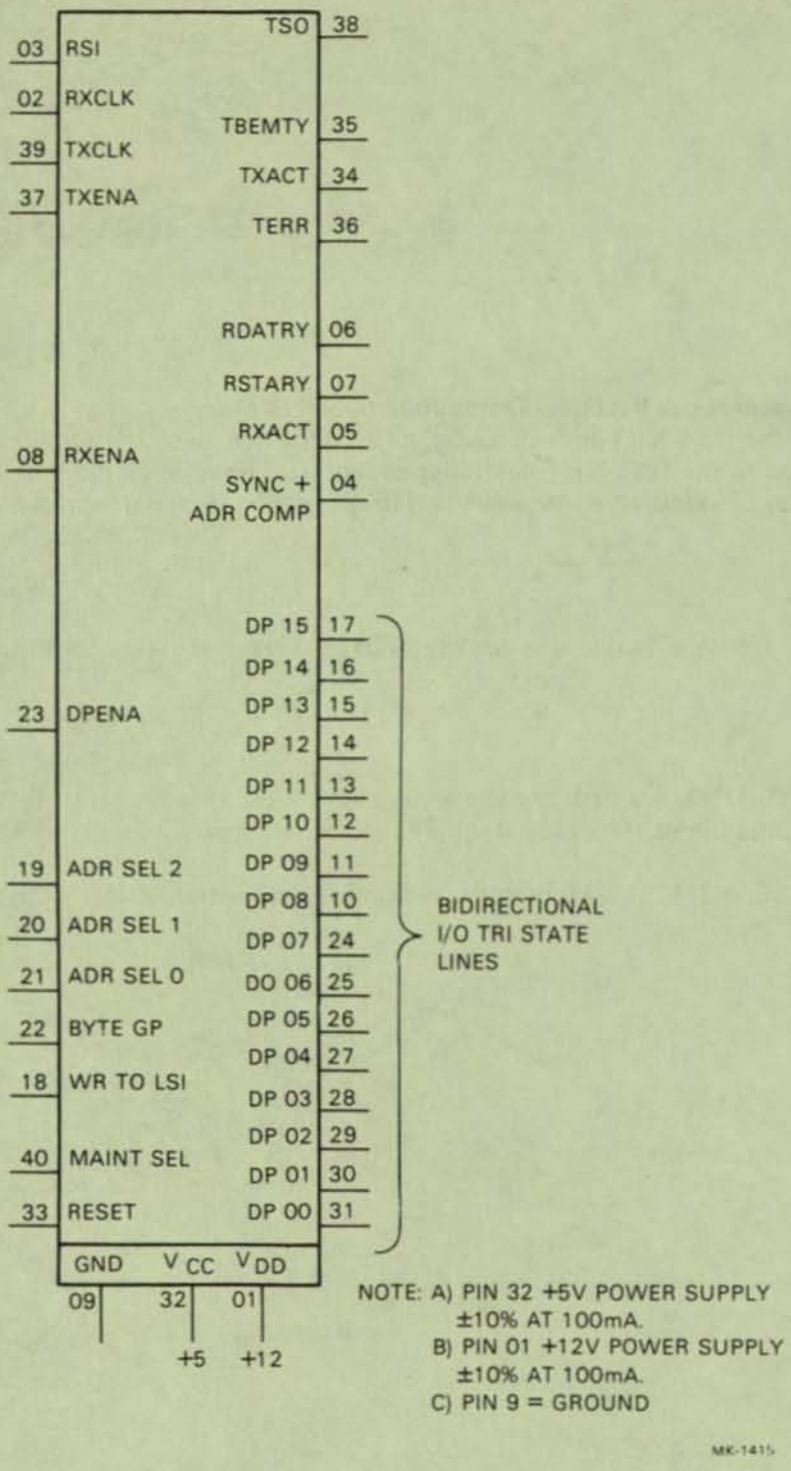


Figure B-1 Terminal Connection Identification Diagram (2112517-0-0 Variation)

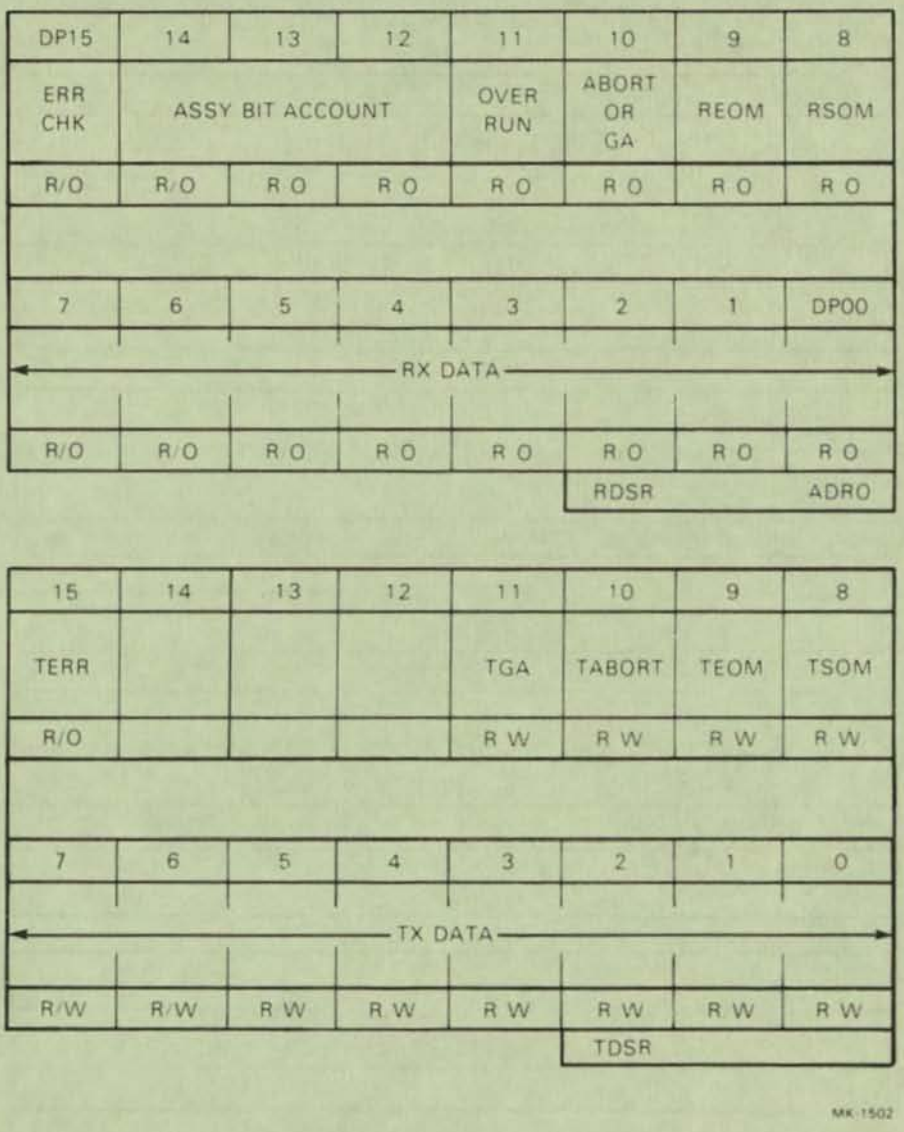


Figure B-2 5025 Internal Register Bit Map (2112517-0-0 Variation) (Sheet 1 of 2)

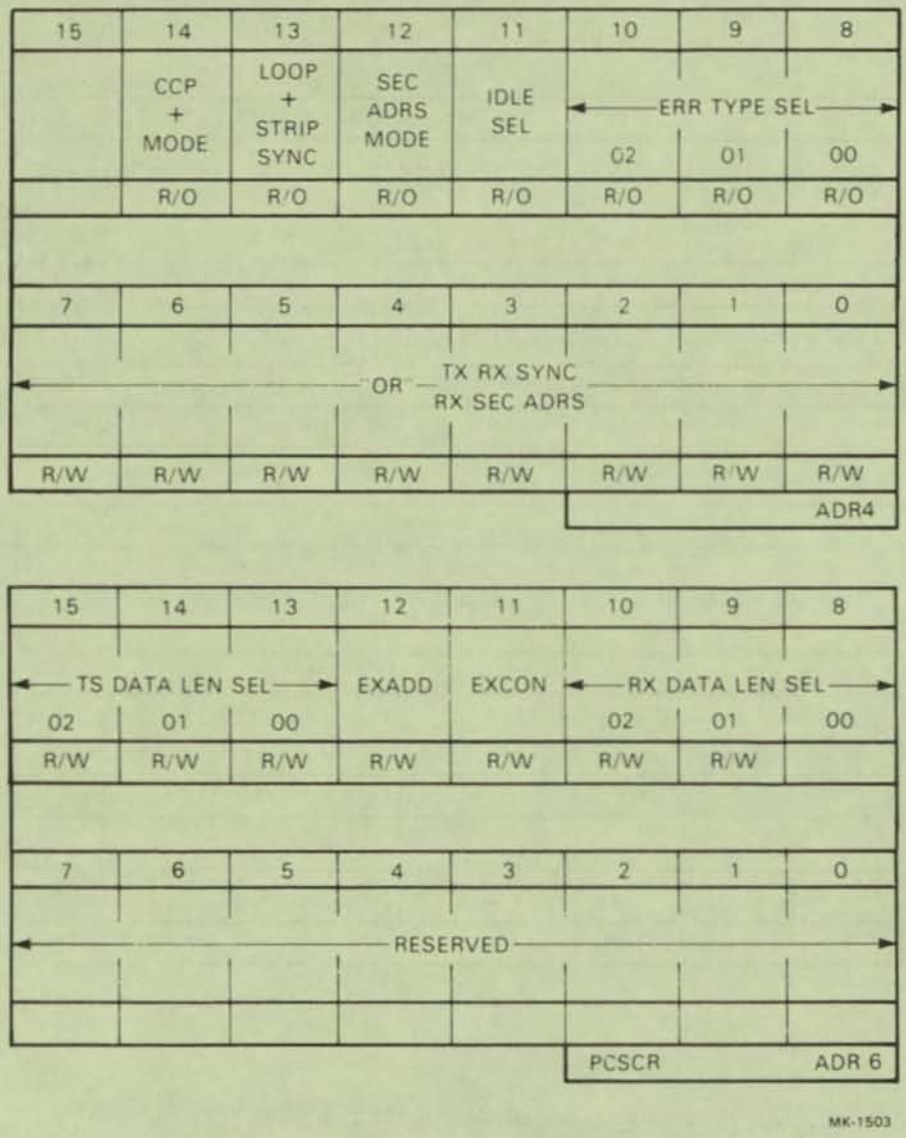


Figure B-2 5025 Internal Register Bit Map (2112517-0-0 Variation) (Sheet 2 of 2)

APPENDIX C
IC DESCRIPTIONS

C.1 GENERAL

This appendix contains data on the LSI-11 chips and some of the unusual ICs used by the DPV11. The other ICs are common, widely-used logic devices. Detailed specifications on these chips are readily available, and hence are not included here.

C.2 DC003 INTERRUPT CHIP

The interrupt chip is an 18-pin DIP device. It provides the circuits to perform an interrupt transaction in a computer system that uses a "pass-the-pulse" type arbitration scheme. The device provides two interrupt channels labeled A and B, with the A section at a higher priority than the B section. Bus signals use high-impedance input circuits or high-drive open-collector outputs, which allow the device to directly attach to the computer system bus. Maximum current required from the V_{cc} supply is 140 mA.

Figure C-1 is a simplified logic diagram of the DC003 IC. Table C-1 describes the signals and pins of the DC003.

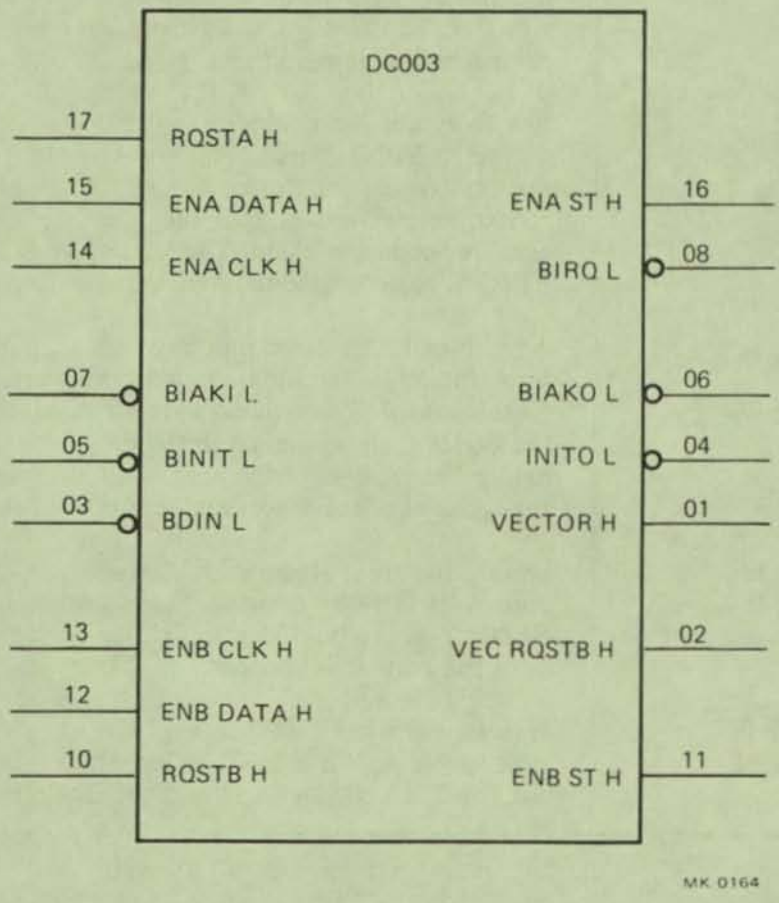


Figure C-1 DC003 Logic Symbol

Table C-1 DC003 Pin/Signal Descriptions

Pin	Signal	Description
1	VECTOR H	Interrupt Vector Gating Signal – This signal gates the appropriate vector address onto the bus and forms the bus signal BRPLY L. Not used in the DPV11.
2	VEC RQSTB H	Vector Request B Signal – When asserted, this signal indicates RQST B service vector address is required. When negated, it indicates RQST A service vector address is required. VECTOR H is the gating signal for the entire vector address; VEC RQST B H is normally bit 2 of the address.
3	BDIN L	Bus Data In – THE BDIN signal always precedes a BIAK signal.
4	INITO L	Initialize Out – This is the buffered BINIT L signal used in the device interface for general initialization.
5	BINIT L	Bus Initialize – When asserted, this signal brings all drive lines to their negated state (except INITO L).
6	BIAKO L	Bus Interrupt Acknowledge – This signal is the daisy-chained signal that is passed by all devices not requesting interrupt service (see BIAKI L). Once passed by a device, it must remain passed until a new BAIKI L is generated.
7	BIAKI L	Bus Interrupt Acknowledge – This signal is the processor's response to BIRQ L true. This signal is daisy-chained such that the first requesting device blocks the signal propagation while nonrequesting devices pass the signal on as BIAKO L to the next device in the chain. The leading edge of BIAKI L causes BIRQ L to be unasserted by the requesting device.
8	BIRQ L	Asynchronous Bus Interrupt Request – The request is generated by a true RQST signal along with the associated true Interrupt Enable signal. The request is removed after the acceptance of the BDIN L signal and on the leading edge of the BAIKI L signal, or the removal of the associated interrupt enable, or due to the removal of the associated request signal.
17 10	RQSTA H RQSTB H	Device Interrupt Request Signal – When asserted with the enable A/B flip-flop asserted, this signal causes the assertion of BIRQ L on the bus. This signal line normally remains asserted until the request is serviced.
16 11	ENA ST H ENB ST H	Interrupt Enable – This signal indicates the state of the interrupt enable A/B internal flip-flop which is controlled by the signal line ENA/B DATA H and the ENA/B CLK H clock line.

Table C-1 DC003 Pin/Signal Descriptions (Cont)

Pin	Signal	Description
15 12	ENA DATA H ENB DATA H	Interrupt Enable Data – The level on this line, in conjunction with the ENA/B CLK H signal, determines the state of the internal interrupt enable A flip-flop. The output of this flip-flop is monitored by the ENA/B ST H signal.
14 13	ENA CLK H ENB CLK H	Interrupt Enable Clock – When asserted (on the positive edge), interrupt enable A/B flip-flop assumes the state of the ENA/B DATA H signal line.

C.3 DC004 PROTOCOL CHIP

The protocol chip is a 20-pin DIP device that functions as a register selector, providing the signals necessary to control data flow into and out of up to four word registers (8 bytes). Bus signals can directly attach to the device because receivers and drivers are provided on the chip. An RC delay circuit is provided to slow the response of the peripheral interface to data transfer requests. The circuit is designed such that if tight tolerance is not required, then only an external 1K $\times$20 percent resistor is necessary. External RCs can be added to vary the delay. Maximum current required from the V_{cc} supply is 120 mA.

Figure C-2 is a simplified logic diagram of the DC004 IC. Signal and pin definitions for the DC004 are shown in Table C-2.

C.4 DC005 BUS TRANSCEIVER CHIP

The 4-bit transceiver is a 20-pin DIP, low-power Schottky device for primary use in peripheral device interfaces, functioning as a bidirectional buffer between a data bus and peripheral device logic. In addition to the isolation function, the device also provides a comparison circuit for address selection and a constant generator, useful for interrupt vector addresses. The bus I/O port provides high-impedance inputs and high-drive (70 mA) open-collector outputs to allow direct connection to a computer's data bus. On the peripheral device side, a bidirectional port is also provided, with standard TTL inputs and 20 mA tri-state drivers. Data on this port is the logical inversion of the data on the bus side.

Three address jumper inputs are used to compare against three bus inputs and to generate the signal MATCH. The MATCH output is open-collector, which allows the output of several transceivers to be wire-ANDed to form a composite address match signal. The address jumpers can also be put into a third logical state that disconnects that jumper from the address match, allowing for "don't care" address bits. In addition to the three address jumper inputs, a fourth high-impedance input line is used to enable/disable the MATCH output.

Three vector jumper inputs are used to generate a constant that can be passed to the computer bus. The three inputs directly drive three of the bus lines, overriding the action of the control lines.

Two control signals are decoded to give three operational states: receive data, transmit data, and disable.

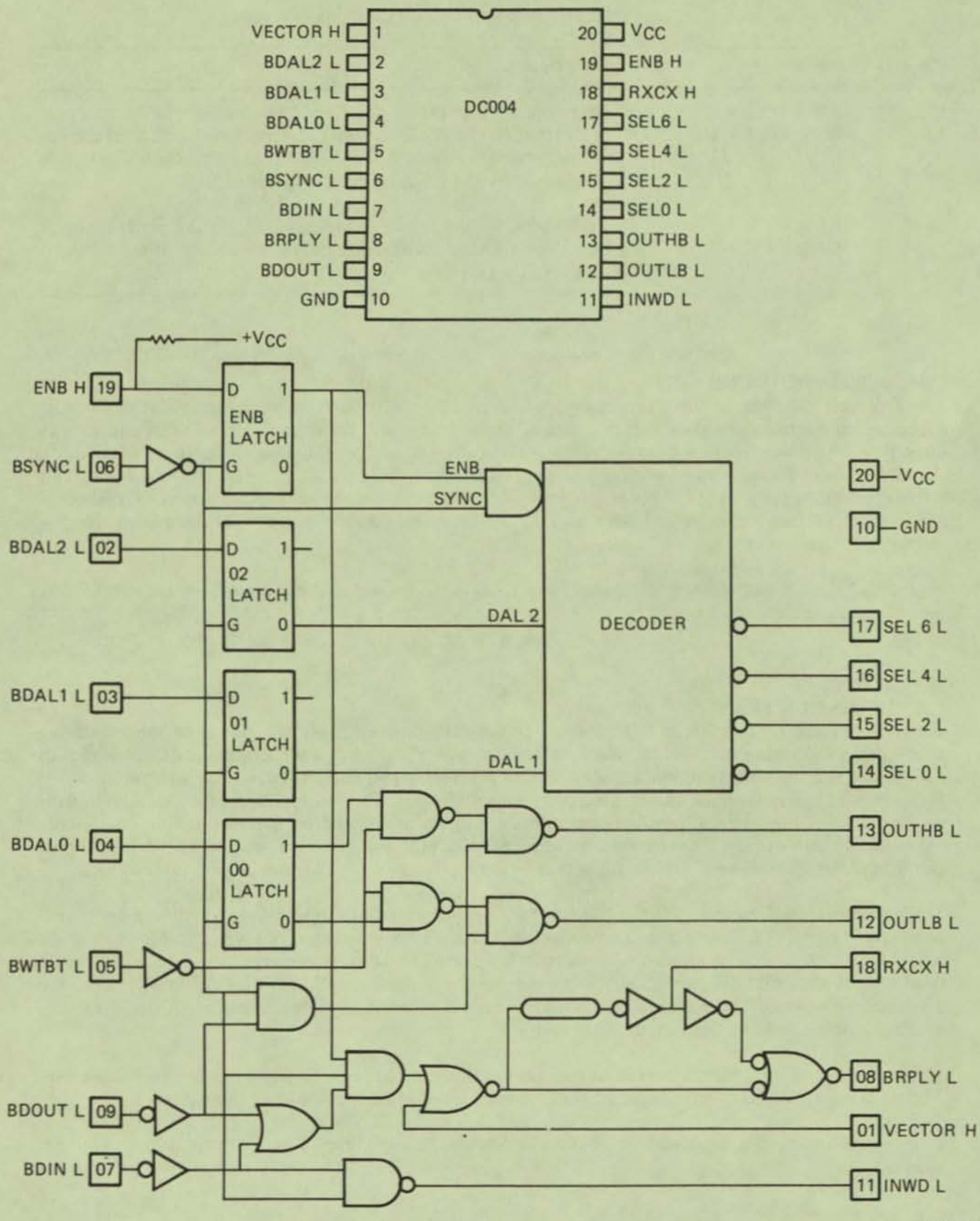


Figure C-2 DC004 Simplified Logic Diagram

Table C-2 DC004 Pin/Signal Descriptions

Pin	Signal	Description
1	VECTOR H	Vector – This input causes BRPLY L to be generated through the delay circuit. Independent of BSYNC L and ENB H.
2 3 4	BDAL2 L BDAL1 L BDAL0 L	Bus Data Address Lines – These signals are latched at the assert edge of BSYNC L. Lines 2 and 1 are decoded for the select outputs; line 0 is used for byte selection.
5	BWTBT L	Bus Write/Byte – While the BDOUT L input is asserted, this signal indicates a byte or word operation: asserted = byte, unasserted = word. Decoded with BDOUT L and latched BDAL0 L, BWTBT L is used to form OUTLB L and OUTHB L.
6	BSYNC L	Bus Synchronize – At the assert edge of this signal, address information is trapped in four latches. While unasserted, this signal disables all outputs except the vector term of BRPLY L.
7	BDIN L	Bus Data In – This is a strobing signal to effect a data input transaction. BDIN L generates BRPLY L through the delay circuit and INWD L.
8	BRPLY L	Bus Reply – This signal is generated through an RC delay by VECTOR H, and strobed by BDIN L or DBOUT L, and BSYNC L and latched ENB H.
9	BDOUT L	Bus Data Out – This is a stobing signal to effect a data output transaction. Decoded with BWTBT L and BDALO, it is used to form OUTLB L and OUTHB L. BDOUT L generates BRPLY L through the delay circuit.
11	INWD L	In Word – Used to gate (read) data from a selected register onto the data bus. It is enabled by BSYNC L and strobed by BDIN L.
12 13	OUTLB L OUTHB L	Out Low Byte, Out High Byte – Used to load (write) data into the lower, higher, or both bytes of a selected register. It is enabled by BSYNC L and the decode of BWTBT L and latched BDAL0 L. It is strobed by BDOUT L.
14 15 16 17	SEL0 L SEL2 L SEL4 L SEL6 L	Select Lines – One of these four signals is true as a function of BDAL2 L and BDAL1 L if ENB H is asserted at the assert edge of BYSNC L. They indicate that a word register has been selected for a data transaction. These signals never become asserted except at the assertion of BSYN L (then only if ENB H is asserted at that time) and, once asserted, are not negated until BSYNC L is negated.
18	RXCX H	External Resistor Capacitor Node – This node is provided to vary the delay between the BDIN L, BDOUT L, and VECTOR H inputs and BRPLY L output. The external resistor should be tied to V_{cc} and the capacitor to ground. As an output, it is the logical inversion of BRPLY L.

Table C-2 DC004 Pin/Signal Descriptions (Cont)

Pin	Signal	Description
19	ENB H	Enable – This signal is latched at the asserted edge of BSYNC L and is used to enable the select outputs and the address term of BRPLY L.

Maximum current required from the V_{cc} supply is 100 mA.

Figure C-3 is a simplified logic diagram of the DC005 IC. Signal and pin definitions for the DC005 are shown in Table C-3.

C.5 26LS32 QUAD DIFFERENTIAL LINE RECEIVER

The 26LS32 line receiver is a 16-pin DIP device. Terminal connections are shown in Figure C-4.

C.6 8640 UNIBUS RECEIVER

The 8640 is a quad 2-input NOR. Its equivalent circuit is shown in Figure C-5.

C.7 8881 NAND

The 8881 is a quad 2-input NAND. The schematic and pin identifications are shown in Figure C-6.

C.8 9636A DUAL LINE DRIVER

The 9636A is an 8-pin DIP device specified to satisfy the requirements of EIA standards RS-423-A and RS-232-C. Additionally, it satisfies the requirements of CCITT V.28, V.10 and the federal standard FIPS 1030.

The output slew rates are adjustable by a single external resistor connected from pin 1 to ground.

The logic diagram and terminal identification are shown in Figure C-7.

C.9 9638 DUAL DIFFERENTIAL LINE DRIVER

The 9638 is an 8-pin DIP device specified to satisfy the requirements of EIA RS-422-A and CCITT V.11 specifications.

The logic diagram and terminal identification are shown in Figure C-8.

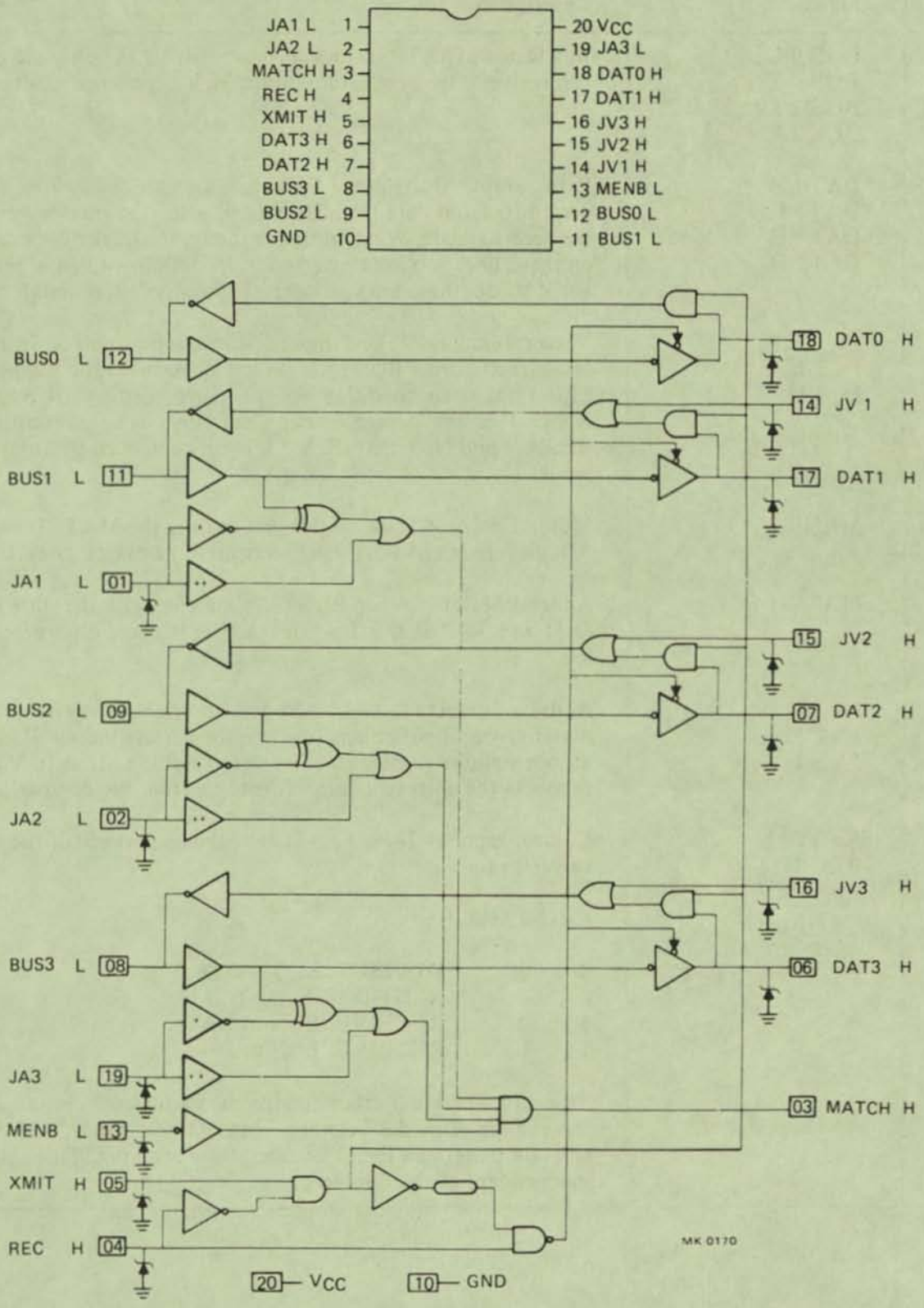
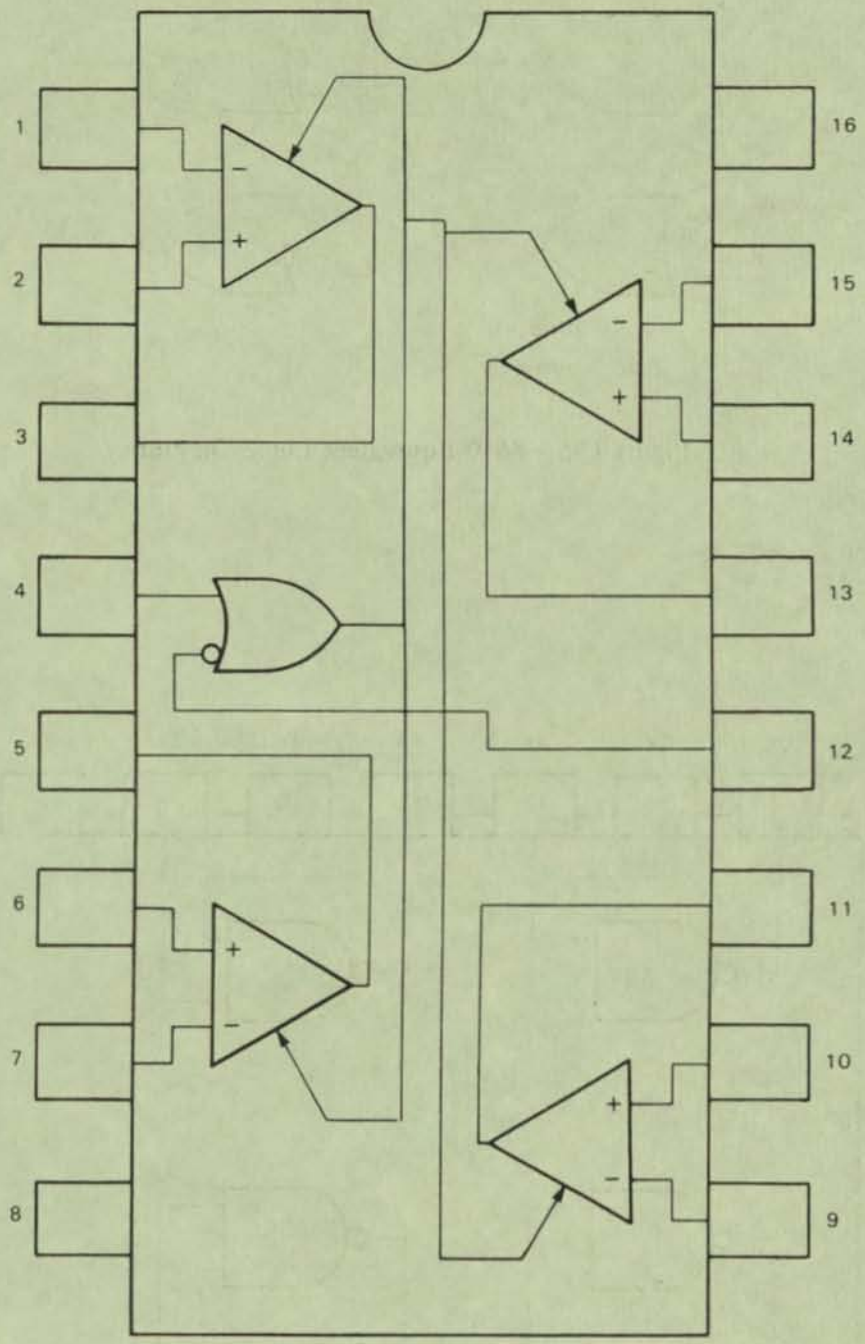


Figure C-3 DC005 Simplified Logic Diagram

Table C-3 DC005 Pin/Signal Descriptions

Pin	Signal	Description
12 11 9 8	BUS 0 L BUS 1 L BUS 2 L BUS 3 L	Bus Data – This set of four lines constitutes the bus side of the transceiver. Open-collector output; high-impedance inputs. Low = 1.
18 17 7 6	DAT0 H DAT1 H DAT2 H DAT3 H	Peripheral Device Data – These four tri-state lines carry the inverted received data from BUS (3:0) when the transceiver is in the receive mode. When in transmit data mode, the data carried on these lines is passed inverted to BUS (3:0). When in the disabled mode, these lines go open (high impedance). High = 1.
14 15 16	JV 1 H JV 2 H JV 3 H	Vector Jumpers – These inputs, with internal pull-down resistors, directly drive BUS (3:1). A low or open on the jumper pin causes an open condition on the corresponding BUS pin if XMIT H is low. A high causes a one (low) to be transmitted on the BUS pin. Note that BUS 0 L is not controlled by any jumpr input.
13	MENB L	Match Enable – A low on this line enables the MATCH output. A high forces MATCH low, overriding the match circuit.
3	MATCH H	Address Match – When BUS (3:1) matches with the state of JA (3:1) and MENB L is low, this output is open; otherwise, it is low.
1 2 19	JA 1 L JA 2 L JA 3 L	Address Jumpers – A strap to ground on these inputs allows a match to occur with a one (low) on the corresponding BUS line; an open allows a match with a zero (high); a strap to V_{cc} disconnects the corresponding address bit from the comparison.
5 4	XMIT H REC H	Control Inputs – These lines control the operational of the transceiver as follows. REC XMIT 0 0 DISABLE: BUS and DAT open 0 1 XMIT DATA: DAT to BUS 1 0 RECEIVE: BUS to DAT 1 1 RECEIVE: BUS to DAT To avoid tri-state overlap conditions, an internal circuit delays the change of modes between Transmit data mode, and delays tri-state drivers on the DAT lines from enabling. This action is independent of the disable mode.



NOTE: PIN 1 IS MARKED FOR ORIENTATION. NUMBERS INDICATED DENOTE TERMINAL NUMBERS.

TERMINAL IDENTIFICATION

1. INPUT A	16. POSITIVE SUPPLY VOLTAGE (V_{CC})
2. INPUT A	15. INPUT B
3. OUTPUT A	14. INPUT B
4. ENABLE	13. OUTPUT B
5. OUTPUT C	12. $\overline{\text{ENABLE}}$
6. INPUT C	11. OUTPUT D
7. INPUT C	10. INPUT D
8. GROUND	9. INPUT D

MK 1340

Figure C-4 26LS32 Terminal Connection Diagram and Terminal Identification

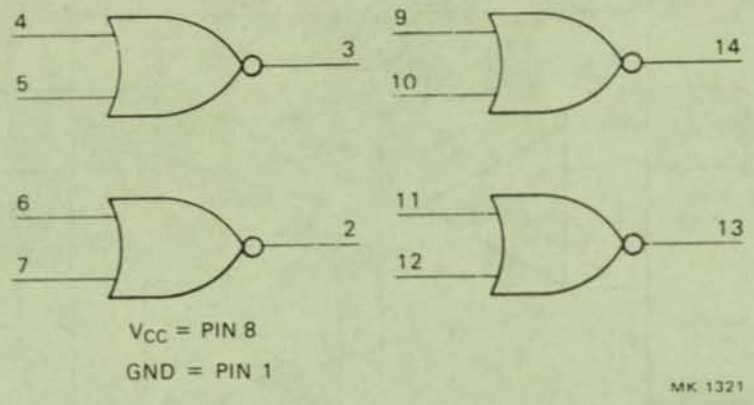


Figure C-5 8640 Equivalent Logic Diagram

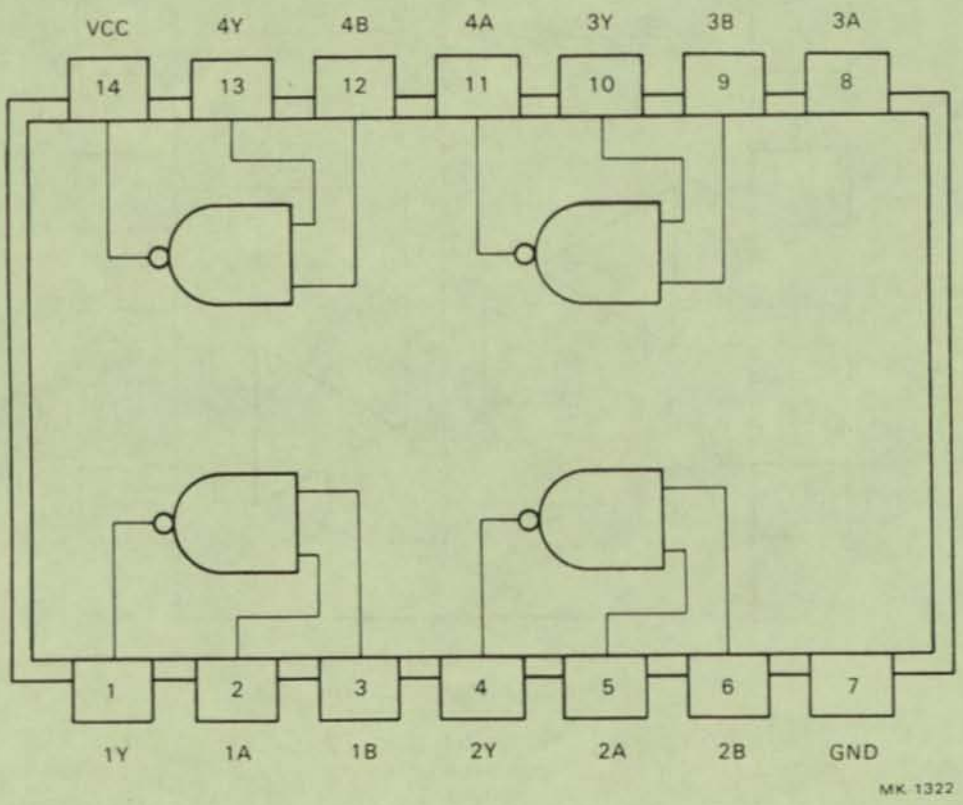
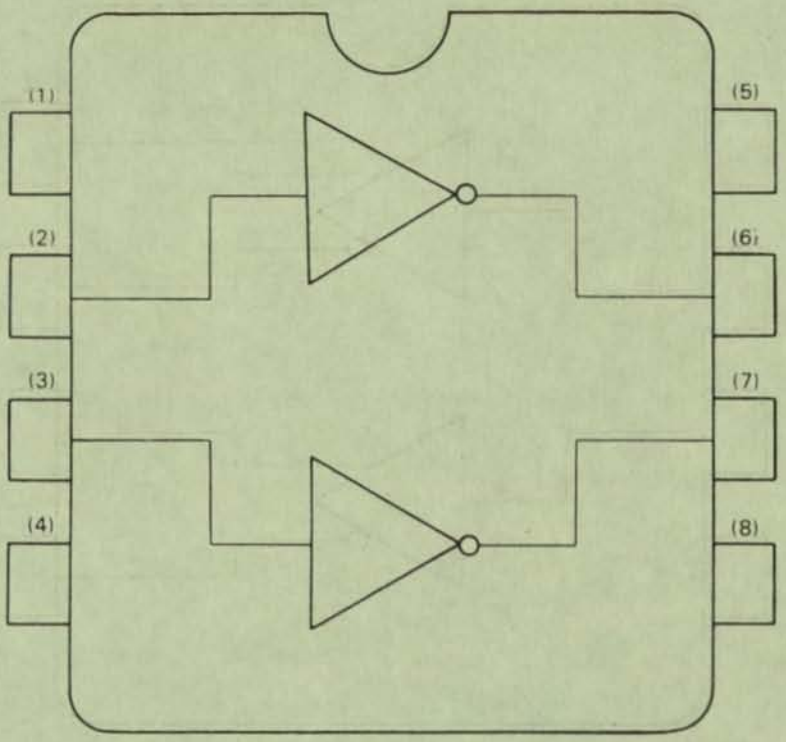


Figure C-6 8881 Pin Identification



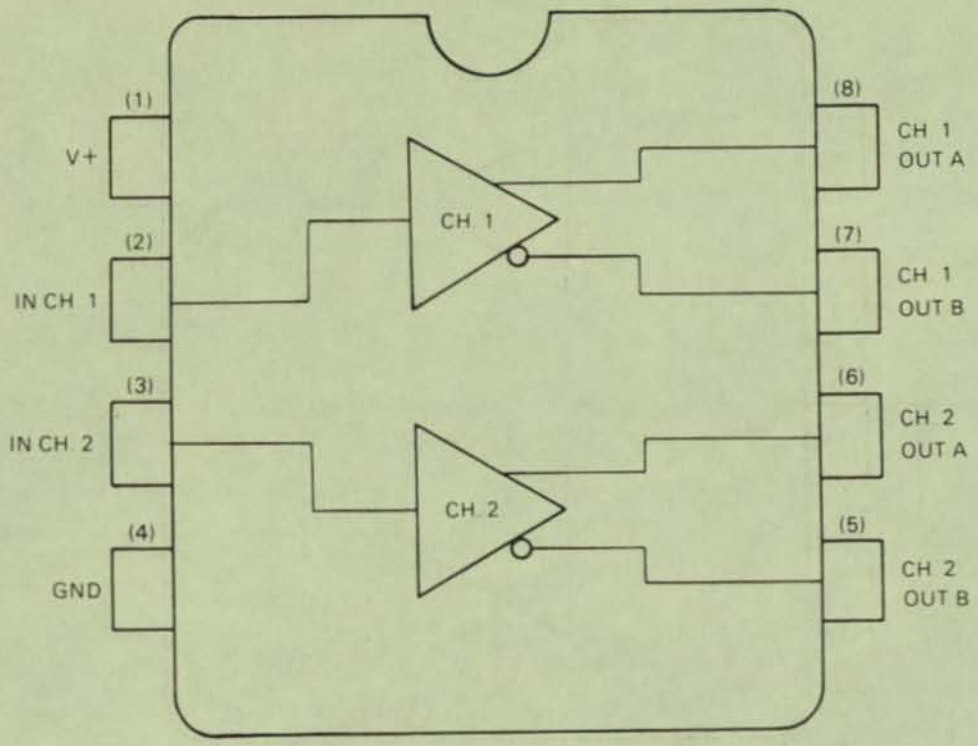
NOTE NUMBERS IN ( ) DENOTE TERMINAL NUMBERS

TERMINAL IDENTIFICATION

(1) WAVESHAPE CONTROL (RISE AND FALL TIME)
(2) INPUT A
(3) INPUT B
(4) POWER AND SIGNAL GROUND
(5) NEGATIVE SUPPLY VOLTAGE
(6) OUTPUT B
(7) OUTPUT A
(8) POSITIVE SUPPLY VOLTAGE (V_{CC})

MK-1323

Figure C-7 9636A Logic Diagram and Terminal Identification



NOTE: NUMBERS IN ( ) DENOTE TERMINAL NUMBERS

TERMINAL IDENTIFICATION

1. POSITIVE SUPPLY VOLTAGE
2. CHANNEL 1 INPUT
3. CHANNEL 2 OUTPUT
4. SUPPLY AND SIGNAL GROUND
5. CHANNEL 2 INVERTED OUTPUT
6. CHANNEL 2 NON INVERTED OUTPUT
7. CHANNEL 1 INVERTED OUTPUT
8. CHANNEL 1 NON INVERTED OUTPUT

MK 1324

Figure C-8 9638 Logic Diagram and Terminal Identification

APPENDIX D
PROGRAMMING EXAMPLES

Two examples are included in this appendix. The first is an example for bit-oriented protocols, and the second is an example for byte count-oriented protocols.

These are only examples and are not intended for any other purpose.

```
        .TITLE  DPV11  --  DPV-11 DDM FOR BIT ORIENTED PROTOCOLS
        .IDENT  /X00/
;
; COPYRIGHT (C) 1980 BY
; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
;
;       EXAMPLE OF AN APPLICATION RSX-11M BIT ORIENTED DPV-11 DEVICE DRIVER
;               *** NOTE - THIS IS NOT A RUNNING DRIVER
;
        .MCALL  HWDDF$,$INTSX,$INTXT,MDCDF$,CCBDF$,TMPDF$,ASYRET,SYNRET
        HWDDF$          ; DEFINE THE HARDWARE REGISTERS
        CCBDF$          ; DEFINE THE CCB OFFSETS
        MDCDF$          ; DEFINE THE MODEM CONTROL SYMBOLS
        TMPDF$          ; DEFINE LINE-TABLE TEMPLATE OPERATORS
;
;       DEVICE CHARACTERISTICS DEFINED IN  -D.DCHR-
;
DC.HDX  =  000001       ; HALF-DUPLEX LINE INDICATOR    (WORD #0)
DC.PRT  =  000007       ; PROTOCOL SELECTION FIELD      (WORD #1)
DC.MPT  =  000010       ; MULTI-POINT CONFIGURATION     (WORD #1)
DC.SEC  =  000020       ; MULTI-POINT SECONDARY MODE    (WORD #1)
DC.ADR  =  000040       ; STATION ADDRESS IS 16 BITS    (WORD #1)
DC.SPS  =  000013       ; SDLC PRIMARY STATION          (COMPOSITE)
DC.SSS  =  000033       ; SDLC SECONDARY STATION        (COMPOSITE)
;
;       DEVICE STATUS FLAGS DEFINED IN  -D.FLAG-
;
DD.ENB  ==  001         ; IF ZERO, LINE HAS BEEN ENABLED
DD.STR  ==  002         ; IF ZERO, LINE HAS BEEN STARTED
DD.EOM  ==  CF.EOM      ;  --(UNUSED)--
DD.SOM  ==  CF.SOM      ;  --(UNUSED)--
DD.ABT  ==  020         ; TRANSMIT ABORTED DUE TO UNDERRUN
DD.SYN  ==  CF.SYN      ; TRANSMIT SYNC-TRAIN REQUIRED
DD.TRN  ==  CF.TRN      ; TRANSMIT LINE TURN-AROUND REQUIRED
DD.ACT  ==  200         ; TRANSMITTER READY FOR NEXT FRAME
DD.DIS  ==  DD.ENB!DD.STR ; INITIAL STATUS = DISABLED, STOPPED
;
;       [ SEL 0 ]  --  MODEM CONTROL BITS
;
DSCHG   =  100000       ; DATA SET CHANGE
DSRING  =  040000       ; RING INDICATOR
DSCTS   =  020000       ; CLEAR TO SEND
DSCARY  =  010000       ; CARRIER INDICATOR
DSMODR  =  001000       ; MODEM READY
DSITEN  =  000040       ; DATA SET INTERRUPT ENABLE
DSLOOP  =  000010       ; DATA SET LOOPBACK
DSRTS   =  000004       ; REQUEST TO SEND
DSDTR   =  000002       ; DATA TERMINAL READY
DSSEL   =  000001       ; SELECT FREQUENCY OR REMOTE LOOPBACK
;
;       [ SEL 0 ]  --  RECEIVER CONTROL BITS
;
RXACT   =  004000       ; RECEIVER ACTIVE
RXSRDY  =  002000       ; RECEIVER STATUS READY
```

```
RXFLAG  =  000400        ; RECEIVER FLAG DETECT
RXDONE  =  000200        ; RECEIVER DONE
RXITEN  =  000100        ; RECEIVER INTERRUPT ENABLE
RXREN   =  000020        ; RECEIVER ENABLE
;
;       [ SEL 2 ]  --  RECEIVER STATUS INPUTS
;
RXERR   =  100000        ; RECEIVER CRC ERROR
RXABC   =  070000        ; RECEIVER ASSEMBLED BIT COUNT
RXBFOV  =  010000        ; RECEIVER BUFFER OVERFLOW (SOFTWARE ERROR)
RXOVRN  =  004000        ; RECEIVER DATA OVERRUN
RXABRT  =  002000        ; RECEIVED ABORT
RXENDM  =  001000        ; RECEIVED END OF MESSAGE
RXSTRM  =  000400        ; RECEIVED START OF MESSAGE
;
;       [ SEL 2 ]  --  MODE CONTROL OUTPUTS
;
DPAPA   =  100000        ; ALL PARTIES ADDRESSED
DPDECM  =  040000        ; DDCMP / BISYNC OPERATION
DPSTRP  =  020000        ; STRIP SYNC OR LOOP MODE
DPSECS  =  010000        ; SDLC / ADCCP SECONDARY STATION SELECT
DPIDLE  =  004000        ; IDLE MODE SELECT
DPCRC   =  3*400         ; USE CRC 16 ERROR DETECTION
DPADRC  =  000377        ; STATION ADDRESS OR SYNC CHARACTER
INPRM   =  DPSTRP!DPCRC  ; INITIAL STARTUP PARAMETERS
;
;       [ SEL 4 ]  --  TRANSMITTER STATUS AND CONTROL
;
TCLEN   =  160000        ; TRANSMIT CHARACTER LENGTH
EXADD   =  010000        ; EXTENDED ADDRESS FIELD
EXCON   =  004000        ; EXTENDED CONTROL FIELD
RCLEN   =  003400        ; RECEIVE CHARACTER LENGTH
TXITEN  =  000100        ; TRANSMITTER INTERRUPT ENABLE
TXREN   =  000020        ; TRANSMITTER ENABLE
TXMAI   =  000010        ; MAINTENANCE MODE SELECT
TXDONE  =  000004        ; TRANSMITTER DONE
TXACT   =  000002        ; TRANSMITTER ACTIVE
TXRES   =  000001        ; DEVICE RESET
;
;       [ SEL 6 ]  --  TRANSMITTER OUTPUT CONTROLS
;
TXLATE  =  100000        ; TRANSMITTER DATA LATE (UNDERRUN)
TXGO    =  004000        ; TRANSMITTER GO AHEAD
TXABRT  =  002000        ; TRANSMITTER ABORT
TXENDM  =  001000        ; TRANSMIT END OF MESSAGE
TXSTRM  =  000400        ; TRANSMIT START OF MESSAGE
;
; PROCESS DISPATCH TABLE
;
$DXPTB::
        .WORD   $SDASX          ; TRANSMIT ENABLE
        .WORD   $SDASR          ; RECEIVE  ENABLE (ASSIGN BUFFER)
        .WORD   $SDKIL          ; KILL I/O ENABLE
        .WORD   $SDCTL          ; CONTROL  ENABLE
```

```
        .WORD   $SDTIM          ; TIME OUT

        .SBTTL  $SDPRI  --  RECEIVE INTERRUPT SERVICE ROUTINE
;+
; FUNCTION:
;
;       THE DEVICE INTERRUPT IS VECTORED BY THE HARDWARE TO THE
;       DEVICE LINE TABLE. THE '$SDPRI' LABEL IS ENTERED VIA A
;       CALLING SEQUENCE IN THE LINE TABLE AT OFFSET 'D.RXIN'.
;
; ON ENTRY:
;
;         R5  = ADDRESS OF 'D.RDBF' IN THE LINE TABLE
;       0(SP) = SAVED R5
;       2(SP) = INTERRUPTED PC
;       4(SP) = INTERRUPTED PS
;
; OUTPUTS:
;
;       R5 = ADDRESS OF 'D.RDB2' IN THE LINE TABLE
;       D.RVAD = RECEIVER STATUS BITS FROM CSR [SEL 2]
;-

$SDPRI::
        MOV     R3,-(SP)        ;;; SAVE REGISTERS
        MOV     R4,-(SP)        ;;; ...
        MOV     @(R5)+,R4       ;;; GET CHARACTER AND FLAGS
        BIC     #RXABC,R4       ;;; DON'T WORRY ABOUT ASSEMBLED BIT COUNT
    .IF DF M$$MGE
        MOV     KISAR6,-(SP)    ;;; SAVE CURRENT MAP
        MOV     (R5)+,KISAR6    ;;; MAP TO DATA BUFFER
    .IFTF
        DEC     (R5)+           ;;; DECREMENT BUFFER BYTE COUNT
        BMI     DPRBO           ;;; BUFFER OVERFLOW - POST COMPLETE

        MOV     2(R5),R3        ;;; GET CSR+2 ADDRESS
        BIT     #RXSRDY,-(R3)   ;;; ERROR OR END-OF-MESSAGE ?
        BNE     DPRCP           ;;; YES - POST RECEIVE COMPLETE

        MOVB    R4,@(R5)+       ;;; STORE CHARACTER IN RECEIVE BUFFER
    .IFT
        MOV     (SP)+,KISAR6    ;;; RESTORE PREVIOUS MAPPING
    .IFTF
        INC     -(R5)           ;;; ADVANCE BUFFER ADDRESS
        MOV     (SP)+,R4        ;;; RESTORE REGISTERS
        MOV     (SP)+,R3        ;;; ...
        $INTXT                  ;;; EXIT THE INTERRUPT


DPRBO:                          ;;; BUFFER OVERRUN HAS OCCURRED
        BIS     #RXBFOV,R4      ;;; SET (SOFTWARE) ERROR INDICATOR

DPRCP:                          ;;; END-OF-MESSAGE OR ERROR INDICATION
    .IFT
```

```
        MOV     (SP)+,KISAR6    ;;; RESTORE PREVIOUS MAPPING
    .ENDC
        MOV     R4,(R5)+        ;;; SAVE STATUS FLAGS IN 'D.RVAD'
        MOV     (R5)+,R4        ;;; GET CSR+2 ADDR + POINT TO 'D.RPRI'
        BIC     #RXITEN,-(R4)   ;;; CLEAR RECEIVER INTERRUPT ENABLE
        MOV     (SP)+,R4        ;;; RESTORE R4 SO '$INTSV' IS HAPPY
        MOV     (SP)+,R3        ;;; AND R3
        $INTSX                  ;;; DO A TRICKY $INTSV (R5 SAVED BUT NOT R4)
;
;       CHECK FOR ERRORS, POST RECEIVE COMPLETE, ASSIGN NEW BUFFER
;
        MOV     R3,-(SP)                ;; SAVE AN ADDITIONAL REGISTER
        MOV     (R5),R4                 ;; CCB ADDRESS TO R4 (R5 POPPED)
        ADD     #D.RCNT-D.RCCB,R5       ;; BACK UP TO THE RESI DUAL COUNT
        SUB     (R5)+,C.CNT1(R4)        ;; COMPUTE RECEIVED FRAME BYTE COUNT
        CLR     R3                      ;; SET R3 FOR COMPLETION STATUS

        BIC     #61777,(R5)+    ;; ANY ERRORS REPORTED ?
        BEQ     40$             ;; NO -- POST RECEIVE COMPLETE O.K.
        ASR     -(R5)           ;; SHIFT ERROR INDICATORS...
        ASR     (R5)+           ;; ...TWO PLACES RIGHT
        ASRB    -(R5)           ;; SHIFT 'RXABRT' INTO C-BIT
        MOVB    (R5)+,R3        ;; USE INDICATORS AS TABLE INDEX
        MOV     RCVERR-2(R3),R3 ;; R3 NOW = CCB STATUS FLAGS
        BCC     40$             ;; FRAME NOT ABORTED - POST COMPLETE
        INC     D.RABT-D.RDB2(R5) ;COUNT NUMBER OF ABORTED FRAMES
        CALL    RBFUSE          ;; RE-INITIALIZE WITH THE SAME BUFFER
        BR      60$             ;; RE-ENABLE INTERRUPTS FOR NEXT FRAME

40$:    BIS     C.STS(R4),R3    ;; INCLUDE RE-SYNC STATUS, IF ANY
        MOV     R3,-(SP)        ;; SAVE STATUS REPORTED TO DLC
        CALL    $DDRCP          ;; POST RECEIVE COMPLETE
        MOV     (SP)+,R3        ;; RECOVER COMPLETION STATUS
        CALL    RBFSET          ;; ASSIGN NEW CCB TO THE RECEIVER
        BCS     DREXIT          ;; FAILED - LEAVE RECEIVER INACTIVE
        TST     R3              ;; WAS AN ERROR REPORTED TO DLC ?
        BMI     DRCLRA          ;; YES - DISABLE RCVR FOR RE-SYNC

60$:    MOV     -(R5),R3        ;; RECEIVER CSR [SEL 2] TO R3
        BIS     #RXITEN,-(R3)   ;; RE-ENABLE RECEIVER INTERRUPTS
DREXIT:
        MOV     (SP)+,R3        ;; RESTORE REGISTER R3
        RETURN                  ;; EXIT TO THE SYSTEM

;+
; DRCLRA:
;
;       MOMENTARILY RESET 'RXREN' FLAG IN ORDER TO FORCE RECEIVER
;       RE-SYNCHRONIZATION.  THIS IS REQUIRED FOR ANY ERROR WHICH
;       TERMINATES THE RECEIVE OPERATION IN MID-FRAME.
;
; ON ENTRY:
;
;       R5 = ADDRESS OF 'D.RCCB' IN THE LINE TABLE
```

```
;       R4 = ADDRESS OF 'C.STS' IN THE NEWLY-ASSIGNED CCB
;      (SP)= SAVED R3 VALUE
;-
DRCLRA:
        MOV     -(R5),R3                ;; RCVR CSR ADDRESS [SEL 2] TO R3
        BIC     #RXREN,-(R3)            ;; RESET RCVR ENABLE FOR RE-SYNC
        BIS     #CS.RSN,(R4)            ;; SET RE-SYNC IN CCB 'C.STS'
        BIS     #RXREN!RXITEN,(R3)      ;; RE-ENABLE THE RECEIVER
        BR      DREXIT                  ;; RESTORE R3 AND EXIT

        .SBTTL  $SDPTI  --  TRANSMIT INTERRUPT SERVICE ROUTINE
;+
; FUNCTION:
;
;       THE DEVICE INTERRUPT IS VECTORED BY THE HARDWARE TO THE
;       DEVICE LINE TABLE. THE '$SDPTI' LABEL IS ENTERED VIA A
;       CALLING SEQUENCE IN THE LINE TABLE AT OFFSET 'D.TXIN'.
;       ONCE FRAME TRANSMISSION IS INITIATED, EACH INTERRUPT IS
;       HANDLED BY THE ROUTINE ADDRESSED VIA THE 'D.TSPA' WORD.
;
; ON ENTRY:
;
;         R5  = ADDRESS OF 'D.TCSR' IN THE LINE TABLE
;       0(SP) = SAVED R5
;       2(SP) = INTERRUPTED PC
;       4(SP) = INTERRUPTED PS
;
; ON EXIT:
;
;       R5 = ADDRESS OF 'D.TCCB' IN THE LINE TABLE
;-

$SDPTI::
        MOV     R4,-(SP)            ;;; SAVE R4
        MOV     (R5)+,R4            ;;; GET TRANSMITTER CSR ADDRESS
        TST     (R4)+               ;;; POINT TO [SEL 6] + TEST UNDERRUN
        JMP     @(R5)+              ;;; GO TO CORRECT STATE PROCESSOR
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
;       CURRENT STATE =         MONITOR CSR FOR 'CLEAR TO SEND'     ;
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
TISCTS:
        BIT     #DSCTS,-6(R4)               ;;; IS 'CLEAR TO SEND' ACTIVE YET ?
        BNE     TISIFL                      ;;; YES - START TO SEND THE FRAME
        BITB    #DD.SYN,D.FLAG-D.TCNT(R5)   ;;; SYNC-TRAIN REQUIRED ?
        BEQ     TISIFX                      ;;; NO -- SEND FLAGS UNTIL 'CTS'
        MOV     #TXSTRM!TXENDM,(R4)         ;;; START + END SENDS SYNC STRING
        BR      TISEXT
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
;       CURRENT STATE =         SEND INITIAL FRAME 'FLAG'           ;
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
TISIFL:
        MOV     #TISTRT,-(R5)   ;;; NEXT STATE = SEND ADDRESS BYTE
TISIFX:
        MOV     #TXSTRM,(R4)    ;;; SEND AN SDLC FLAG CHARACTER
```

```
        BR      TISEXT

;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
;       CURRENT STATE =         SEND ADDR BYTE FOLLOWING 'FLAG' ;
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
TISTRT:
        DEC     (R5)                            ;;; DECREMENT COUNT FOR ADDR BYTE
        MOV     D.TADC-D.TCNT(R5),(R4)          ;;; SEND ADDR, CLEAR 'TXSTRM'
        MOV     #TISDAT,-(R5)                   ;;; NEXT STATE = DATA TRANSFER
        BR      TISEXT
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
;       CURRENT STATE =         TRANSFER FRAME DATA BYTES       ;
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
TISDAT:
        BMI     TISLAT          ;;; UNDERRUN - ABORT AND RE-TRANSMIT
        DEC     (R5)+           ;;; DECREMENT DATA BYTE COUNT
        BMI     TISEND          ;;; ALL DONE - SEND END-MSG SEQUENCE
    .IF DF M$$MGE
        MOV     KISAR6,-(SP)    ;;; SAVE CURRENT MAPPING
        MOV     (R5)+,KISAR6    ;;; MAP TO THE TRANSMIT BUFFER
    .IFTF
        INC     (R5)            ;;; ADVANCE THE BUFFER ADDRESS
        MOVB    @(R5)+,(R4)     ;;; NEXT CHARACTER TO BE SENT
    .IFT
        MOV     (SP)+,KISAR6    ;;; RESTORE PREVIOUS MAPPING
    .ENDC

TISEXT:
        MOV     (SP)+,R4        ;;; COMMON LEVEL-7 INTERRUPT EXIT
        $INTXT                  ;;; RESTORE R4
                                ;;; EXIT INTERRUPT SERVICE
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
;       CURRENT STATE =         DATA BYTE-COUNT EXHAUSTED       ;
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
TISEND:
        MOV     #TXENDM,(R4)            ;;; TRANSMIT END-OF-MSG SEQUENCE
        INC     -(R5)                   ;;; ADJUST R5 AND CLEAR 'D.TCNT'
        MOV     #TISFLG,-(R5)           ;;; NEXT STATE = IDLE FLAGS (ASSUMED)
        ASLB    D.FLAG-D.TSPA(R5)       ;;; TEST FOR LINE TURN-AROUND
        BPL     TISEXT                  ;;; NO -- IDLE THE LINE WITH FLAGS
        MOV     #TISPAD,(R5)            ;;; YES - SEND PADS, THEN DISABLE
        BR      TISEXT
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
;       CURRENT STATE =         SEND 'ABORT' AS PAD AFTER 'FLAG';
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
TISPAD:
        CLRB    D.FLAG-D.TCNT(R5)       ;;; RESET THE DEVICE FLAG BYTE
        MOV     #TISCLR,-(R5)           ;;; NEXT STATE = SEND SECOND PAD
        MOV     #TXABRT,(R4)            ;;; SET 'TXABRT' TO SEND A PAD
        BR      TISEXT
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
;       CURRENT STATE =         SEND SECOND 'ABORT' AS PAD      ;
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
TISCLR:
        MOV     #TISRTS,-(R5)   ;;; NEXT STATE = DROP 'REQUEST TO SEND'
```

```
TISCLX:
        MOV     #TXABRT,(R4)    ;;; SETUP TO SEND ANOTHER 'ABORT' CHAR
        BIC     #TXREN,-(R4)    ;;; DISABLE THE TRANSMITTER
        BR      TISEXT

;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
;       CURRENT STATE =         DROP REQUEST TO SEND + EXIT             ;
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
TISRTS:
        BIT     #DC.HDX,D.DCHR-D.TCNT(R5) ;;; HALF-DUPLEX CHANNEL ?
        BEQ     TISDON                    ;;; NO -- LEAVE 'RTS' ACTIVE
        BIC     #DSRTS,-5(R4)             ;;; DROP 'REQUEST TO SEND' LINE
        BR      TISDON                    ;;; POST TRANSMIT COMPLETE
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
;       CURRENT STATE =         TRANSMITTER DATA UNDERRUN               ;
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
TISLAT:
        MOV     #TISDON,-(R5)             ;;; NEXT STATE = RE-TRANSMIT
        MOVB    #DD.ABT,D.FLAG-D.TSPA(R5) ;;; THIS FRAME WAS ABORTED
        INC     D.TURN-D.TSPA(R5)         ;;; COUNT THE ERROR EVENTS
        BR      TISCLX                    ;;; SEND PAD, DISABLE TRANSMITTER
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
;       CURRENT STATE =         IDLE FLAGS BETWEEN FRAMES               ;
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
TISFLG:
        MOV     #TXSTRM,(R4)              ;;; CLEAR 'TXENDM', IDLE FLAGS
        MOVB    #DD.ACT,D.FLAG-D.TCNT(R5) ;;; TRANSMITTER IS ACTIVE
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
;       CURRENT STATE =         POST COMPLETE OR RE-TRANSMIT            ;
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
TISDON:
        ADD     #D.TPRI-D.TCNT,R5 ;;; ADJUST LINE TABLE POINTER
        BIC     #TXITEN,-(R4)     ;;; DISABLE 'TXDONE' INTERRUPTS
        MOV     (SP)+,R4          ;;; RESTORE R4 FOR PRIORITY DROP
        $INTSX                    ;;; '$INTSV' W/O R4 SAVED (POPS R5)

        MOV     R3,-(SP)                    ;; SAVE AN ADDITIONAL REGISTER
        MOV     (R5),R4                     ;; ACTIVE CCB ADDRESS TO R4
        CLR     (R5)+                       ;; THIS CCB IS NO LONGER ACTIVE
        BITB    #DD.ABT,D.FLAG-D.TCBQ(R5)   ;; WAS THE FRAME ABORTED ?
        BNE     TRSTRT                      ;; YES - SETUP RE-TRANSMISSION
        TST     D.KCCB-D.TCBQ(R5)           ;; TRANSMIT KILL IN PROGRESS ?
        BNE     CKILLT                      ;; YES - RETURN CCB'S TO THE DLC
        CLR     R3                  ;; SET COMPLETION STATUS = SUCCESS
        CALL    $DDXMP              ;; POST TRANSMIT COMPLETE TO THE DLC
        MOV     (R5),R4             ;; FIRST CCB ON SECONDARY CHAIN
        BEQ     TREXIT              ;; NONE THERE - TRANSMITTER IDLE
        MOV     (R4),(R5)           ;; REMOVE CCB FROM SECONDARY CHAIN

;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
;       CURRENT STATE =         START UP FRAME TRANSMISSION             ;
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
TRSTRT:
        CLR     (R4)                        ;; CLEAR CCB LINKAGE WORD
```

```
        MOV     R4,-(R5)
        TST     -(R5)                   ;; SETUP AS THE ACTIVE CCB
        ADD     #C.FLG1,R4              ;; SKIP BACK OVER 'D.TPRI'
        BISB    (R4),D.FLAG-D.TPRI(R5)  ;; POINT TO THE CCB BUFFER FLAGS
        BICB    #DD.ABT,D.FLAG-D.TPRI(R5) ;; SAVE FLAGS FOR LEVEL-7 USE
                                        ;MAKE SURE 'ABORT' FLAG IS OFF
        MOV     -(R4),D.TCNT-D.TPRI(R5) ;; SET TRANSMIT BYTE COUNT
        CLR     -(R5)                   ;; INITIALIZE 'D.TADC' WORD
        MOV     -(R4),-(R5)             ;; SET TRANSMIT BUFFER ADDRESS
.IF DF M$$MGE
        MOV     -(R4),-(R5)             ;; SET TRANSMIT BUFFER RELOCATION
        MOV     KISAR6,-(SP)            ;; SAVE THE CURRENT APR6 MAPPING
        MOV     (R5)+,KISAR6            ;; MAP TO THE TRANSMIT BUFFER
.IFTF
        MOVB    @(R5)+,(R5)             ;; MOVE ADDRESS BYTE TO 'D.TADC'
.IFT
        MOV     (SP)+,KISAR6            ;; RESTORE PREVIOUS APR6 MAPPING
.ENDC

        ADD     #D.TSPA-D.TADC,R5       ;; BACK UP TO STATE PROCESSOR CELL
        TSTB    D.FLAG-D.TSPA(R5)       ;; IS THE TRANSMITTER READY NOW ?
        BPL     20$                     ;; NO -- ENABLE IT, THEN START

        MOV     #TISTRT,(R5)            ;; INITIAL STATE = SEND ADDR BYTE
        BR      40$                     ;; ENABLE INTERRUPTS AND EXIT

20$:    MOV     -2(R5),R3               ;; TRANSMITTER CSR [SEL 4] TO R3
        BIS     #DSRTS,-4(R3)           ;; ASSERT 'REQUEST TO SEND'
        BIS     #TXREN,(R3)+            ;; ENABLE THE TRANSMITTER

        MOV     #TISCTS,(R5)            ;; INITIAL STATE = WAIT FOR 'CTS'

40$:    BIS     #TXITEN,@-(R5)          ;; RE-ENABLE TRANSMIT INTERRUPTS

TREXIT:
        MOV     (SP)+,R3                ;; RESTORE R3 FROM ENTRY
        ASYRET                          ;; EXIT WHEREVER APPROPRIATE, ASYNC

;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
;       CURRENT STATE =         TRANSMIT KILL OR TIMEOUT                ;
;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -;
CKILLT:
        MOV     #CS.ERR!CS.ABO,-(SP) ;; TRANSMIT COMPLETION STATUS

CKTTMO:
        BIC     #TXREN,@D.TCSR-D.TCBQ(R5) ;; DISABLE TRANSMITTER
        MOV     (R5),(R4)       ;; ADD SECONDARY CHAIN TO PRIMARY
        CLR     (R5)+           ;; CLEAR SECONDARY CHAIN POINTER

20$:    MOV     (SP),R3         ;; COMPLETION STATUS TO R3
        MOV     (R4),-(SP)      ;; NEXT CCB ADDRESS TO STACK
        CLR     (R4)            ;; MAKE SURE LINK WORD IS ZERO
        CALL    $DDXMP          ;; POST A CCB COMPLETE W/ERROR
        MOV     (SP)+,R4        ;; NEXT CCB ADDRESS TO R4
```

```
        BNE     20$             ;; MORE TO GO - CONTINUE
        TST     (SP)+           ;; CLEAN STATUS OFF THE STACK

        MOV     (R5),R4         ;; KILL CCB ADDRESS TO R4
        BEQ     TREXIT          ;; NONE - RESTORE R3 AND EXIT
        CLR     (R5)            ;; KILL NO LONGER IN PROGRESS
        CLR     R3              ;; STATUS = SUCCESSFUL

        CMPB    #FC.KIL,C.FNC(R4) ;; KILL-I/O OR CONTROL FUNCTION ?
        BNE     40$               ;; CONTROL - POST IT COMPLETE
        CALL    $DDKCP            ;; POST KILL-I/O COMPLETE
        BR      TREXIT            ;; RESTORE R3 AND EXIT

40$:    CALL    $DDCCP          ;; POST CONTROL COMPLETE
        BR      TREXIT          ;; RESTORE R3 AND EXIT

        .SBTTL  $SDASX  --  TRANSMIT ENABLE ENTRY
;+
; FUNCTION:
;
;       '$SDASX' IS ENTERED (VIA THE DISPATCH TABLE) TO QUEUE A
;       CCB CONTAINING AN SDLC FRAME TO BE TRANSMITTED.  IF THE
;       TRANSMITTER IS BUSY, THE CCB IS QUEUED TO THE SECONDARY
;       CCB CHAIN.  IF NOT, THE TRANSMITTER IS ENABLED TO START
;       TRANSMITTING THE NEW FRAME.
;
; ON ENTRY:
;
;       R4 = ADDRESS OF TRANSMIT ENABLE CCB
;       R5 = ADDRESS OF DEVICE LINE TABLE
;       PS = PRIORITY OF CALLING DLC PROCESS
;
; ON EXIT:
;
;       ALL REGISTERS ARE UNPREDICTABLE
;-

$SDASX::
        MOV     R3,-(SP)        ;; SAVE R3 FOR EXIT VIA 'TRSTRT'
        MOV     D.TCSR(R5),R3   ;; TRANSMIT CSR ADDRESS [SEL 4] TO R3
        BIC     #TXITEN,(R3)    ;; DISABLE TRANSMITTER INTERRUPTS
        ADD     #D.TCCB,R5      ;; POINT TO ACTIVE CCB ADDRESS CELL

        TST     (R5)+           ;; IS THERE AN ACTIVE CCB ?
        BEQ     TRSTRT          ;; NO -- START UP THE TRANSMITTER
        MOV     R4,-(SP)        ;; SAVE POINTER TO FIRST CCB

20$:    MOV     R5,R4           ;; COPY THE CCB ADDRESS TO R4
        MOV     (R4),R5         ;; ADDRESS OF THE NEXT CCB TO R5
        BNE     20$             ;; LOOP UNTIL WE FIND THE END

        MOV     (SP)+,(R4)      ;; LINK NEW CCB TO END OF CHAIN
        CLR     @(R4)+          ;; MARK NEW END OF CCB CHAIN
        BIS     #TXITEN,(R3)    ;; RE-ENABLE TRANSMITTER INTERRUPTS
```

```
        BR      TREXIT          ;; RESTORE R3 AND EXIT

        .SBTTL  $SDASR  --  RECEIVE ENABLE AFTER BUFFER WAIT
;+
; FUNCTION:
;
;       THIS ROUTINE IS CALLED BY THE BUFFER POOL MANAGER WHEN
;       A BUFFER ALLOCATION REQUEST CAN BE SATISFIED, FOLLOWING
;       AN ALLOCATION FAILURE AND A CALL TO '$RDBWT'.
;
; ON ENTRY:
;
;       R4 = ADDRESS OF CCB AND RECEIVE BUFFER
;       R5 = ADDRESS OF DEVICE LINE TABLE
;
; ON EXIT:
;
;       R5 = ADDRESS OF 'D.RCCB' IN THE LINE TABLE
;       R4 = ADDRESS OF 'C.STS' IN THE CCB
;      (SP)= SAVED VALUE OF R3
;-

$SDASR::
        ADD     #D.RDB2,R5      ;; POINT TO SECOND RCVR-CSR WORD
        CALL    RBFUSE          ;; ASSIGN BUFFER TO THE RECEIVER
        BIS     #CS.BUF,(R4)    ;; PREV. ALLOC. FAILURE TO CCB 'C.STS'

        MOV     R3,-(SP)        ;; PUSH R3 FOR EXIT AT 'DREXIT', ABOVE
        JMP     DRCLRA          ;; RESET AND ACTIVATE THE RECEIVER

;+
; $SDSTR  --  START UP DEVICE AND LINE ACTIVITY
;-

$SDSTR::
        BITB    #DD.ENB,D.FLAG(R5)      ;; HAS THE LINE BEEN ENABLED ?
        BNE     60$                     ;; NO -- REJECT THE 'START'

        MOV     D.RDBF(R5),R3   ;; RECEIVER CSR ADDR [SEL 2] TO R3
        MOV     D.STN(R5),(R3)  ;; SET ADDRESS BYTE + OPERATING MODE
        BIS     #RXREN,-(R3)    ;; ENABLE THE RECEIVER

        MOV     R5,-(SP)        ;; SAVE LINE TABLE START ADDRESS
        ADD     #D.RDB2,R5      ;; ADJUST R5 FOR BUFFER ROUTINE
        CALL    RBFSET          ;; ASSIGN A RECEIVE CCB AND BUFFER
        BCS     20$             ;; FAILED - START THE TRANSMITTER
        BIS     #RXITEN,(R3)    ;; ENABLE RECEIVER INTERRUPTS

20$:    MOV     (SP)+,R5                ;; RECOVER LINE TABLE START
        CLRB    D.FLAG(R5)              ;; LINE HAS BEEN STARTED
        BIT     #DC.HDX,D.DCHR(R5)      ;; CHECK THAT ASSUMPTION
        BNE     CTLCMP                  ;; CORRECT - STARTUP COMPLETE
        BIS     #DSRTS,(R3)             ;; ASSERT 'REQUEST TO SEND' LINE
        BR      CTLCMP                  ;; ...AND POST START COMPLETE
```

```
60$:    MOV     #CS.ERR!CS.DIS,R3       ;; STATUS = LINE DISABLED
        BR      CTLERR                  ;; RETURN ERROR W/COMPLETION


DP.NOP:                                 ;; CONTROL FUNCTION = NO-OPERATION
CTLCMP:
        CLR     R3                      ;; STATUS = SUCCESSFUL
CTLERR:
        MOV     (SP)+,R4                ;; RECOVER SAVED R4 VALUE
        SYNRET                          ;; SYNCHRONOUS RETURN

        .SBTTL  $SDSTP  --  STOP DEVICE AND LINE ACTIVITY
;---------------------------------------------------------------;
;       'S T O P'   C O N T R O L   F U N C T I O N             ;
;---------------------------------------------------------------;
$SDSTP::
        MOV     D.RDBF(R5),R3   ;; RECEIVER CSR ADDR [SEL 2] TO R3
        MOV     #DSDTR,-(R3)    ;; DISABLE RECEIVER, LEAVE 'DSDTR' ACTIVE
        CLR     4(R3)           ;; DISABLE TRANSMITTER

        MOV     D.RCCB(R5),R4   ;; ACTIVE RECEIVE CCB TO R4
        BEQ     20$             ;; NONE THERE - SKIP IT
        CALL    $RDBRT          ;; RETURN BUFFER TO THE POOL

20$:    CLR     D.RCCB(R5)      ;; NO RECEIVE CCB ASSIGNED
        CLR     R4              ;; CLEAR R4 FOR PARAMETER USE
        BISB    D.SLN(R5),R4    ;; SET SYSTEM LINE NUMBER IN R4
        CALL    $RDBQP          ;; PURGE BUFFER WAIT QUEUE REQUESTS

        BISB    #DD.STR,D.FLAG(R5) ;; LINE IS NO LONGER STARTED
        TST     D.TCCB(R5)      ;; IS THERE AN ACTIVE TRANSMIT CCB ?
        BEQ     CTLCMP          ;; NO -- POST CONTROL COMPLETE

        MOV     (SP)+,D.KCCB(R5)    ;; SAVE THE CONTROL CCB FOR TIMEOUT
        MOVB    #1,(R5)             ;; MAKE SURE THE TIMER IS ACTIVE
        ASYRET                      ;; RETURN WITH ASYNCHRONOUS COMPLETION


        .SBTTL  $SDENB  --  ENABLE THE LINE AND DEVICE
;---------------------------------------------------------------;
;        E N A B L E   L I N E   A N D   D E V I C E            ;
;---------------------------------------------------------------;
$SDENB::
        MOV     D.RDBF(R5),R3   ;; RECEIVER CSR ADDRESS [SEL 2] TO R3
        BIS     #TXRSET,2(R3)   ;; RESET THE DEVICE (1-US SINGLE-SHOT)

        ADD     #D.DCHR+2,R5    ;; POINT TO CHARACTERISTICS WORD #1
        BIT     #DC.ADR,(R5)+   ;; 16-BIT STATION ADDRESS ?
        BEQ     20$             ;; NO -- SHOULD BE ALL SET
        SWAB    (R5)            ;; USE THE HIGH-ORDER BYTE IN DPV-11
20$:    BIC     #^C<DPADRC>,(R5) ;;CLEAR HIGH-ORDER BYTE OF 'D.STN' WORD
        BIS     #INPRM,(R5)     ;;SETUP INITIAL PARAMETERS
        BIC     #DC.ADR,-(R5)   ;; ADDRESS-SIZE NO LONGER SIGNIFICANT
```

```
        CMPB    #DC.SPS,(R5)    ;; SDLC PRIMARY-STATION MODE ?
        BEQ     40$             ;; YES - FLAGS ARE SETUP AS IS
        CMPB    #DC.SSS,(R5)    ;; SDLC SECONDARY-STATION MODE ?
        BNE     60$             ;; NO -- OPERATING MODE INVALID
        BIS     #DPSECS,2(R5)   ;; ENABLE STATION ADDRESS CHECKING

40$:    BIS     #DSDTR,-(R3)    ;; ASSERT 'DATA TERMINAL READY' LINE
        BICB    #DD.ENB,D.FLAG-D.DCHR-2(R5) ;; LINE IS ENABLED
        BR      CTLCMP          ;; POST CONTROL FUNCTION COMPLETE

60$:    MOV     #CS.ERR!CS.DEV,R3 ;; ERROR STATUS - INVALID PROTOCOL
        BR      CTLERR            ;; POST CONTROL COMPLETE WITH ERROR

        .SBTTL  $SDDIS  --  DISABLE THE LINE
;
$SDDIS::
        MOV     #CS.ERR!CS.ENB,R3         ;; ERROR CODE IF NOT STOPPED
        BITB    #DD.STR,D.FLAG(R5)        ;; IS LINE STATE CORRECT ?
        BEQ     CTLERR                    ;; NO -- REJECT THE DISABLE

        MOV     D.RDBF(R5),R3             ;; ADDRESS OF RECEIVER CSR [SEL 2]
        CLR     -(R3)                     ;; DISABLE RECEIVER + TURN DTR OFF
        MOVB    #DD.ENB!DD.STR,D.FLAG(R5) ;; LINE NO LONGER ENABLED
        BR      CTLCMP                    ;; CLEAR CARRY AND EXIT


        .SBTTL  $SDMSN  --  SENSE MODEM STATUS
;-----------------------------------------------------------------;
;             S E N S E   M O D E M   S T A T U S                 ;
;-----------------------------------------------------------------;
$SDMSN::
        CLR     R4              ;; CLEAR R4 FOR RETURN CODES
        MOV     D.RDBF(R5),R3   ;; ADDRESS OF RECEIVER CSR [SEL 2]

        BIT     #DSDSR,-(R3)    ;; IS THE DATA-SET READY ?
        BEQ     20$             ;; NO --
        BIS     #MC.DSR,R4      ;; YES - SET INDICATOR IN R4

20$:    BIT     #DSRING,(R3)    ;; IS THE PHONE RINGING ?
        BEQ     40$             ;; NO --
        BIS     #MC.RNG,R4      ;; YES - SET INDICATOR IN R4

40$:    BIT     #DSCARY,(R3)    ;; IS THERE CARRIER PRESENT ?
        BEQ     60$             ;; NO -- POST COMPLETE
        BIS     #MC.CAR,R4      ;; YES - SET INDICATOR IN R4

60$:    MOV     R4,(SP)         ;; RETURN RESULTS IN (SAVED) R4
        BR      CTLCMP          ;; POST CONTROL FUNCTION COMPLETE

        .END
```

```
        .TITLE  DPV - BYTE ORIENTED DPV-11 DEVICE DRIVER MODULE
        .IDENT  /X00/
;
; COPYRIGHT (C) 1980 BY
; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
;
;       EXAMPLE OF AN APPLICATION RSX-11M BYTE ORIENTED DPV-11 DEVICE DRIVER
;
        .MCALL  $INTSX,$INTXT,INHIB$,ENABL$
        .MCALL  CCBDF$,TMPDF$,$LIBCL
        .MCALL  MDCDF$
        .MCALL  CHADF$
        MDCDF$                  ; DEFINE MODEM CONTROL SYMBOLS
        CCBDF$                  ; DEFINE THE CCB OFFSETS
        TMPDF$                  ; DEFINE LINE TABLE OFFSET MACROS
        CHADF$                  ; DEFINE DEVICE CHARACTERISTICS
;
; LOCAL SYMBOL DEFINITIONS
;
; TRANSMITTER FLAGS
;
TINIT=  000010                  ; INITIAL TRANSMIT STATUS (HALF DUPLEX)
TXENA=  000020                  ; TRANSMIT ENABLE
TXINT=  000100                  ; TRANSMIT INTERRUPT ENABLE
TXACT=  000002                  ; TRANSMIT ACTIVE
TSOM=   000400                  ; TRANSMIT START OF MESSAGE
TEOM=   001000                  ; TRANSMIT END OF MESSAGE
;
; RECEIVE CSR FLAGS
;
RCVEN=  000020                  ; RECEIVE ENABLE
RXINT=  000100                  ; RECEIVE INTERRUPT ENABLE
CRC=    3*400                   ; RECEIVE CRC CHECK
SSYN=   020000                  ; STRIP SYNC
PROSEL= 040000                  ; PROTOCOL SELECTION (BYTE)
RINIT=  RXINT!RCVEN!DTR         ; INITIAL RECEIVE STATUS
INPRM=  SSYN!PROSEL!CRC         ; INITIALIZATION FLAGS
;
; MODEM STATUS FLAGS
;
RTS=    000004                  ; REQUEST TO SEND LEAD
CTS=    020000                  ; CLEAR TO SEND
DTR=    000002                  ; DATA TERMINAL READY
DSR=    001000                  ; DATA SET READY
RING=   040000                  ; RING INDICATOR
;
; DPV11 DEVICE DRIVER DISPATCH TABLE
;
$DPVTB::.WORD   DPASX           ; TRANSMIT ENABLE
        .WORD   DPASR           ; RECEIVE ENABLE (ASSIGN BUFFER)
        .WORD   DPKIL           ; KILL I/O
        .WORD   DPCTL           ; CONTROL INITIATION
        .WORD   DPTIM           ; TIME OUT
```

```
;+
; **-$DPVRI-DPV11 RECEIVE INTERRUPT SERVICE ROUTINE
;
; THE DEVICE INTERRUPT IS VECTORED TO THE DEVICE LINE TABLE
; BY THE HARDWARE AND THIS ROUTINE IS ENTERED BY A
; 'JSR R5,$DPVRI' INSTRUCTION AT THE BEGINNING OF THE LINE
; TABLE.
;
; INPUTS:
;
;       R5 = ADDRESS OF DEVICE LINE TABLE + 4
;       STACK:
;       0(SP) = SAVED R5
;       2(SP) = INTERRUPTED BIAS
;       4(SP) = INTERRUPTED PC
;       6(SP) = INTERRUPTED PS
;
; OUTPUTS:
;
; ETC.
;-

$DPVRI::
        MOV     R4,-(SP)        ;;; SAVE R4
        MOV     (R5)+,R4        ;;; GET ADDRESS OF RECEIVER DATA BUFFER
        MOV     (R4),R4         ;;; GET CHARACTER AND FLAGS
        BMI     DPRHO           ;;; ANY ERROR IS RECEIVER OVERRUN

        .IF DF  M$$MGE

        MOV     KISAR6,-(SP)    ;;; SAVE CURRENT MAP
        MOV     (R5)+,KISAR6    ;;; MAP TO DATA BUFFER

        .IFTF

        MOVB    R4,@(R5)+       ;;; STORE CHARACTER IN RECEIVE BUFFER

        .IFT

        MOV     (SP)+,KISAR6    ;;; RESTORE PREVIOUS MAPPING

        .ENDC

        DEC     (R5)            ;;; DECREMENT REMAINING BYTE COUNT
        BEQ     DPRCP           ;;; IF EQ RECEIVE COMPLETE
        INC     -(R5)           ;;; ADVANCE BUFFER ADDRESS
        MOV     (SP)+,R4        ;;; RESTORE REGISTERS
        $INTXT                  ;;; EXIT THE INTERRUPT

;
; EXCEPTIONAL RECEIVE SERVICE ROUTINES
;
; HARDWARE OVERRUN
;
```

```
        .ENABL  LSB

DPRH0:  ADD     #<RCNT-RDBF-2>,R5 ;;; POINT TO COUNT CELL
        MOV     #100001,RFLAG-RCNT(R5) ;;; SET FLAGS TO COMPLETE REQUEST AND
                                ;;; CLEAR RECEIVE ACTIVE ON EXIT
        MOV     #CS.ERR+CS.ROV,RSTAT-RCNT(R5) ;;; SET OVERRUN STATUS

;
; RECEIVE BYTE COUNT RUNOUT
;

DPRCP:  MOV     R4,(R5)+        ;;; SAVE CRC FLAG AND POINT TO PRIORITY
        MOV     RDBF-RPRI(R5),R4 ;;; GET RECEIVE DATA BUFFER ADDRESS
        BIC     #RXINT,-(R4)    ;;; CLEAR RECEIVER INTERRUPT ENABLE
        MOV     (SP)+,R4        ;;; RESTORE R4 SO '$INTSV' IS HAPPY
        $INTSX                  ;;; DO A TRICKY $INTSV (R5 PRESAVED BUT NOT R4)
        MOV     R3,-(SP)        ;;  SAVE AN ADDITIONAL REGISTER
        TST     (R5)+           ;;  POINT TO FLAGS WORD
        ASR     (R5)+           ;;  LOAD C-BIT FROM FLAGS (BIT 0)
        BCS     20$             ;;  IF CS DATA, POST COMPLETION
        MOV     (R5),R4         ;;  GET PRIMARY CCB ADDRESS
        .LIST MEB
        $LIBCL  HDRA-RPRIM,R5,$DDHAR,SAV ;; CALL DDHAR THROUGH LINE TABLE
        .NLIST  MEB
        ROR     -2(R5)          ;;  SAVE 'FINAL SEEN' IN FLAGS (BIT 15 SET)
        TST     R3              ;;  EXAMINE BYTE COUNT FOR THIS MESSAGE
        BMI     10$             ;;  IF MI AN INVALID HEADER RECEIVED
        BEQ     7$              ;;  IF EQ SET TO RECEIVE REST OF HEADER
        ADD     #2,R3           ;;  ACCOUNT FOR BCC IN CURRENT COUNT
        MOV     R3,RPCNT-RPRIM(R5) ;; SAVE DATA COUNT UNTIL HEADER CRC
                                ;;    IS CHECKED
7$:     MOV     #5,R3           ;; GET REMAINING HEADER
        INC     -(R5)           ;;  MARK DATA IN PROGRESS IN FLAGS (BIT 0 SET)
        ADD     R3,@-(R5)       ;;  INCLUDE CURRENT COUNT IN TOTAL COUNT
        ADD     #RCNT-RTHRD,R5  ;;  POINT TO CURRENT COUNT
        MOV     R3,(R5)         ;;  SET UP CURRENT BYTE COUNT
        INC     -(R5)           ;;  MOVE BUFFER ADDRESS PAST BCC

        .IF DF  M$$MGE

        MOV     -4(R5),R3       ;;  GET ADDRESS OF RECEIVE DATA BUFFER

        .IFF

        MOV     -(R5),R3        ;;  GET ADDRESS OF RECEIVE DATA BUFFER

        .ENDC

        BR      REXT0           ;;  FINISH IN COMMON CODE

;
; INVALID HEADER RECEIVED
;
```

```
10$:    BIT     #CS.MTL,R3          ;;  MESSAGE TOO LONG ?
        BNE     31$                 ;;  IF NE YES, POST COMPLETION
        MOV     (R5)+,R4            ;;  RECOVER PRIMARY CCB ADDRESS
        CALL    BUFUSE              ;;  SET UP THIS CCB AGAIN (CLEARS 'RSTAT')
        MOV     RDBF-RPRIM(R5),R3 ;; SET POINTER TO REC. DAT. BUFF.
        BR      40$                 ;;  CLEAR RECEIVE ACTIVE TO FORCE RESYNC

;
; POST COMPLETION ON RECEIVE COMPLETE
;
;       R5 =  POINTS TO PRIMARY CCB ADDRESS
;

20$:    TST     RCNT-RPRIM(R5) ;;  IS CRC ERROR FLAG SET ?
        BMI     25$                 ;;  IF MI, YES - CRC IS VALID
        MOV     #CS.ERR+CS.DCR,R3 ;;  ELSE SET CRC ERROR STATUS FOR DLC
        BR      31$                 ;; GO RETURN BUFFER
25$:    MOV     RPCNT-RPRIM(R5),RCNT-RPRIM(R5) ;; SET REMAINING COUNT
        BEQ     30$                 ;; NONE SO END OF MESSAGE
        ADD     RPCNT-RPRIM(R5),@RTHRD-RPRIM(R5) ;; SET TOTAL COUNT IN CCB
        SEC                         ;; FORCE C BIT
        ROL     RFLAG-RPRIM(R5) ;; PUT  Q SYNC BACK & MARK NON HEADER
        INC     RADD-RPRIM(R5)  ;; INCLUDE LAST CHAR IN BUFFER
        MOV     RDBF-RPRIM(R5),R3 ;; GET CSR FOR EXIT
        BR      REXT                ;; TAKE COMMON EXIT
30$:    CLR     R3                  ;; GET GOOD STATUS
31$:    MOV     (R5)+,R4            ;;  GET PRIMARY CCB ADDRESS
        BIS     (R5),R3             ;;  PICK UP ADDITIONAL STATUS
        CALL    $DDRCP              ;;  POST RECEIVE COMPLETION
        MOV     RDBF-RSTAT(R5),R3 ;;  GET ADDRESS OF RECEIVE DATA BUFFER
        CALL    BUFSET              ;;  SET UP NEXT RECEIVE BUFFER
        BCS     REXT1               ;;  IF CS NO BUFFER AVAILABLE TURN OFF RECEIVER
        BNE     40$                 ;;  IF NE CLEAR RECEIVE ACTIVE TO RESYNC
REXT:   CLR     RPCNT-RPRIM(R5) ;; RESET PARTIAL COUNT
REXT0:  BIS     #RXINT,-(R3)        ;;  ENABLE RECEIVER INTERRUPTS
REXT1:  MOV     (SP)+,R3            ;;  RESTORE R3
        RETURN                      ;;  RETURN TO SYSTEM

40$:                                ;;  REF LABEL
;
; CLEAR RECEIVE ACTIVE TO FORCE RESYNC
;
;       R3 = ADDRESS OF RECEIVE DAT BUFFER
;       R5 = ADDRESS OF 'RPRIM'
;

DPCRA:  CLR     -(R5)               ;;  CLEAR FLAGS WORD
        BIC     #RCVEN,-(R3)        ;;  CLEAR RECEIVE ACTIVE FOR RESYNC
        CLR     RPCNT-RFLAG(R5) ;; RESET PARTIAL COUNT
        BIS     #CS.RSN,RSTAT-RFLAG(R5) ;;  INDICATE A RESYNC
        BIS     #RINIT,(R3)         ;;  ENABLE RECEIVER
        BR      REXT1               ;;  FINISH IN COMMON CODE

        .DSABL  LSB
```

```
;+
; **-$DPVTI-DPV11 TRANSMIT INTERRUPT SERVICE
;
;
; THIS ROUTINE IS ENTERED ON A TRANSMITTER INTERRRUPT VIA
; A 'JSR R5,DPVTI' WITH R5 CONTAINING THE ADDRESS OF THE
; DEVICE LINE TABLE OFFSET BY 'TCSR'.
;
; INPUTS:
;
;       R5 = ADDRESS OF DEVICE LINE TABLE + 'TCSR'
;       STACK CONTAINS:
;       0(SP) = INTERRUPTED R5
;       2(SP) = INTERRUPTED BIAS
;       4(SP) = INTERRUPTED PC
;       6(SP) = INTERRUPTED PS
;
; OUTPUTS:
;
; ETC.
;-
        .ENABL  LSB

$DPVTI::
        MOV     R4,-(SP)            ;;; SAVE R4
        MOV     (R5)+,R4            ;;; GET TRANSMITTER CSR ADDRESS
        TST     (R4)+               ;;; TEST FOR UNDERRUN
        BMI     10$                 ;;; IF MI, UNDERRUN - WAIT FOR TIMEOUT
        DEC     TCNT-TCSR-2(R5)     ;;; DECREMENT COUNT
        BEQ     20$                 ;;; IF EQ, BYTE COUNT RUNOUT

        .IF DF  M$$MGE

        MOV     KISAR6,-(SP)        ;;; SAVE CURRENT MAPPING
        MOV     (R5)+,KISAR6        ;;; MAP TO DATA BUFFER

        .IFTF

        MOVB    @(R5)+,(R4)         ;;; OUTPUT A CHARACTER

        .IFT

        MOV     (SP)+,KISAR6        ;;; RESTORE PREVIOUS MAPPING

        .IFTF

        INC     -(R5)               ;;; UPDATE BUFFER ADDRESS
        MOV     (SP)+,R4            ;;; RESTORE R4
        $INTXT

;
; TRANSMITTER UNDERRUN
;
; DISABLE TRANSMITTER INTERRUPTS AND WAIT FOR A TIMEOUT
```

```
;
10$:    BISB    #TSOM/400,1(R4) ;;; CLEAR UNDERRUN BIT
        MOV     #TUNST,TSTAT-TCSR-2(R5) ;;; SET STATE TO DISABLE TRANSMITTER

;
; TRANSMIT BYTE COUNT RUNOUT
;
; OUTPUT TO STATE PROCESSING ROUTINES:
;
;       R3 = ADDRESS OF TRANSMITTER CSR
;       R5 = ADDRESS OF THREAD WORD CELL
;
20$:    ADD     #TPRI-TCSR-2,R5 ;;; POINT TO PRIORITY DATA
        BIC     #TXINT,-(R4)    ;;; CLEAR INTERRUPT ENABLE
        MOV     (SP)+,R4        ;;; RESTORE R4 SO '$INTSV' IS HAPPY
        $INTSX                  ;SAVE WITH R5 ON STACK BUT NOT R4

        .IFT

        MOV     KISAR6,-(SP)    ;;  SAVE CURRENT MAPPING

        .IFTF

        MOV     R3,-(SP)        ;;  SAVE AN ADDITIONAL REGISTER
        MOV     TCSR-TSTAT(R5),R3 ;;  GET TRANSMITTER CSR ADDRESS
        CALLR   @(R5)+          ;;  DISPATCH TO PROCESSING ROUTINE

        .DSABL  LSB

;+
; **-DPASX-ASSIGN A TRANSMIT BUFFER
;
;
; THIS ROUTINE IS ENTERED VIA THE MATRIX SWITCH TO
; QUEUE A CCB FOR TRANSMISSION.
;
; INPUTS:
;
;       R4 = ADDRESS OF CCB TO TRANSMIT
;       R5 = ADDRESS OF DEVICE LINE TABLE
;
; OUTPUTS:
;
;       IF THE TRANSMITTER IS IDLE, TRANSMISSION IS
;       INITIATED; OTHERWISE, THE CCB (OR CHAIN) IS QUEUED TO
;       THE END OF THE SECONDARY CHAIN.
;
; REGISTERS MODIFIED:
;
;       R3, R4, AND R5
;-
```

```
DPASX:
        MOV     TCSR(R5),R3     ; GET TRANSMITTER CSR ADDRESS
        BIC     #TXINT,(R3)     ; DISABLE TRANSMITTER INTERRUPTS
        ADD     #TPRIM,R5       ; POINT TO PRIMARY CELL

        .IFT

        MOV     KISAR6,-(SP)    ; SAVE CURRENT MAPPING

        .IFTF

        MOV     R3,-(SP)        ; SAVE R3
        TST     (R5)+           ; PRIMARY ASSIGNED ?
        BNE     10$             ; IF NE, YES - QUEUE TO SECONDARY CHAIN
        CALL    TBSET           ; SET UP PRIMARY
        BIT     #TXACT,(R3)     ; TRANSMITTER ACTIVE ?
        BEQ     STSTR           ; IF EQ, NO - START IMMEDIATELY
        MOV     #STSTR,-(R5)    ; SET STATE FOR STARTUP
        BR      WAITI           ; WAIT FOR INTERRUPT

10$:    MOV     R4,-(SP)        ; SAVE POINTER TO FIRST CCB
20$:    MOV     R5,R4           ; COPY POINTER TO CCB
        MOV     (R4),R5         ; GET NEXT CCB
        BNE     20$             ; IF NE, KEEP GOING
        MOV     (SP)+,(R4)      ; LINK NEW CCB CHAIN TO LAST CCB
        BR      TEXT2           ; FINISH IN COMMON CODE

;+
; **-STSTR-STARTUP STATE PROCESSING
;
;-

STSTR:  BIS     #RTS,-4(R3)     ; ASSERT REQUEST TO SEND
        BIS     #TXENA,(R3)     ; ENABLE TRANSMITTER
        MOVB    TIMS-TTHRD(R5),TIME-TTHRD(R5) ; START TIMER

;+
; **-STCTS-WAIT FOR CLEAR TO SEND STATE PROCESSING
;
;-

STCTS:  BIT     #CTS,-4(R3)     ; IS CLEAR TO SEND UP ?
        BNE     STSYN           ; IF NE, YES - START SYNC TRAIN
        MOV     #STCTS,-(R5)    ; SET STATE FOR CTS
        MOV     #$PADB,R4       ; SET ADDRESS OF PAD BUFFER
        MOV     #TSOM,-(SP)     ; SET TSOM, CLEAR TEOM
        BR      TEXT1           ; FINISH IN COMMON CODE

;+
; **-STSYN-SYNC TRAIN REQUIRED STATE PROCESSING
;
;-

STSYN:  MOV     #STDAT,-(R5)    ; SET STATE FOR DATA
```

```
        MOV     #$SYNB,R4       ; SET ADDRESS OF SYNC BUFFER
        MOV     #TSOM,-(SP)     ; SET TSOM, CLEAR TEOM
        BR      TEXT0           ; FINISH IN COMMON CODE

;+
; **-STCRC-SEND CRC STATE PROCESSING
;
;-
        .ENABL  LSB

STCRC:  BIS     #TEOM,2(R3)     ; SEND CRC
        CALL    TPOST           ; POST COMPLETION AND SET UP NEXT CCB
        BNE     10$             ; IF NE, NOTHING MORE TO SEND
        MOV     #STDAT,-(R5)    ; ASSUME NEXT STATE IS SEND SYNC'S
        BIT     #CF.SYN,C.FLG-C.BUF(R4) ; ARE SYNC'S REQUIRED ?
        BEQ     20$             ; IF EQ, NO - LEAVE ASSUMED STATE
        MOV     #STSYN,(R5)     ; ELSE CHANGE STATE TO SEND SYNC'S
        BR      20$             ; WAIT FOR CRC TO BE SENT

10$:    MOV     #STIDL,-(R5)    ; SET STATE TO IDLE
        BIC     #TXENA,(R3)     ; SHUT DOWN TRANSMITTER
20$:                            ;

;+
; **-WAITI-WAIT FOR INTERRUPT
;
;-

WAITI:  MOV     #1,TCNT-TSTAT(R5) ; WAIT FOR ONE INTERRUPT
        MOVB    TIMS-TSTAT(R5),TIME-TSTAT(R5) ; START TIMER
        BR      TEXT2           ; FINISH IN COMMON CODE

;+
; **-STIDL-IDLE STATE PROCESSING
;
;-

STIDL:  BIC     #RTS,-4(R3)     ; DROP REQUEST TO SEND
        TST     -(R5)           ;
30$:
        CLRB    TIME-TSTAT(R5)  ; CLEAR TIMER
        BR      TEXT3           ; FINISH IN COMMON CODE

        .DSABL  LSB
;+
; **-TUNST-TRANSMIT DATA UNDER RUN STATE
;
;       RETURN ALL TRANSMIT BUFFERS TO HIGHER LEVEL
;-
TUNST:  ADD     #-TTHRD,R5      ;;TIMEOUT EXPECTS DDM LINE TABLE POINTER
        CLRB    (R5)                    ;;RESET TIMER
        CALL    DPTIM                   ;;FAKE A TIMEOUT TO RETURN BUFFERS
        MOV     #STIDL,TSEC-TSTAT(R5)   ;;SET STATE TO IDLE
        BR      TEXT3                   ;;TAKE COMMON EXIT
```

```
;+
; **-STDAT-DATA STATE PROCESSING
;
;-

STDAT:  MOV     (R5),R4             ; GET ADDRESS OF FLAGS WORD FROM THREAD
        ADD     #C.FLG-C.STS,(R5)   ; UPDATE THREAD POINTER
        TST     (R4)+               ; LAST BUFFER THIS CCB ? (BIT 15 SET)
        BPL     10$                 ; IF PL, NO
        CALL    TPOST               ; POST COMPLETION AND SET UP NEXT CCB
10$:    MOV     #STDAT,-(R5)        ; ASSUME DATA CONTINUES
        BIT     #CF.EOM,C.FLG-C.BUF(R4) ; SEND CRC FOLLOWING THIS BUFFER ?
        BEQ     20$                 ; IF EQ, NO - LEAVE ASSUMED STATE
        MOV     #STCRC,(R5)         ; ELSE CHANGE STATE FOR CRC TO BE SENT
20$:    CLR     -(SP)               ; CLEAR TSOM, CLEAR TEOM

;+
; **-TEXT0-COMMON EXIT ROUTINES
; **-TEXT1-
; **-TEXT2-
; **-TEXT3-
;
;-

TEXT0:  MOVB    TIMS-TSTAT(R5),TIME-TSTAT(R5) ; START TIMER
TEXT1:  ADD     #TCSR-TSTAT+2,R5 ; POINT TO CURRENT BUFFER CELL

        .IFT

        MOV     (R4)+,(R5)+         ; COPY RELOCATION BIAS

        .IFF

        TST     (R4)+               ; SKIP OVER RELOCATION BIAS IN CCB

        .IFTF

        MOV     (R4)+,(R5)+         ; COPY VIRTUAL ADDRESS
        MOV     (R4),(R5)           ;   AND THE BYTE  COUNT

        .IFT

        MOV     -4(R5),KISAR5       ; MAP TO DATA BUFFER

        .IFTF

        BISB    @-2(R5),(SP)        ; BUILD CHARACTER TO OUTPUT
        INC     -2(R5)              ; UPDATE VIRTUAL ADDRESS
        MOV     (SP)+,2(R3)         ; OUTPUT CHARACTER AND FLAGS
TEXT2:  BIS     #TXINT,(R3)         ; ENABLE TRANSMITTER INTERRUPTS
TEXT3:  MOV     (SP)+,R3            ; RESTORE R3
        .IFT
```

```
        MOV     (SP)+,KISAR6    ; RESTORE PREVIOUS MAPPING

        .ENDC

        SEC                     ; SET C-BIT ASYNCHRONOUS COMPLETION
        RETURN                  ; RETURN TO CALLER

;+
; **-DPSTR-DEVICE START-UP
;
; THIS ROUTINE IS CALLED TO ACTIVATE THE DEVICE.
;
;-

DPSTR:  MOV     R4,-(SP)        ; SAVE THE CALLING CCB
        MOV     RDBF(R5),R3     ; GET RECEIVER DATA BUFFER ADDRESS
        MOV     #$SYNC+INPRM,(R3) ; SET INITIAL PARAMETERS
        TST     -(R3)           ; POINT TO RECEIVER CSR
        ADD     #RSTAT,R5       ; POINT TO STATUS WORD
        CALL    BUFSET          ; ASSIGN A PRIMARY CCB (AND BUFFER)
        BCS     20$             ; IF CS GO TO TRANSMITTER
        CLR     -2(R5)          ; CLEAR THE FLAGS WORD
        MOV     #RINIT,(R3)     ; INITIALIZE RECEIVER
20$:    MOV     #TINIT,4(R3)    ; TURN ON TRANSMITTER
        MOVB    DPVCH+3-RPRIM(R5),TIMS-RPRIM(R5) ;SET DDM TIME INTERVAL
        BIT     #1,DPVCH-RPRIM(R5); HALF DUPLEX
        BNE     30$             ; IF NE YES, DONT FORCE FD MODE
        BIC     #TINIT,4(R3)    ; INDICATE FULL DUPLEX
        BIT     #CH.MDT,DPVCH+2-RPRIM(R5)       ;IS THIS A MULTIPOINT SLAVE?
        BNE     30$             ;YES - DO NOT SET REQUEST TO SEND
        BIS     #RTS,(R3)       ; ASSERT REQUEST TO SEND FOR FULL DUPLEX
30$:    MOV     (SP)+,R4        ; RESTORE THE CALLING CCB
        CLC                     ; CLEAR C-BIT SYNCHRONOUS COMPLETION
        RETURN                  ; RETURN

;+
; **-DPSTP-STOP DEVICE
;
; RETURN OUTSTANDING BUFFERS AND CLEAR TIMERS
;-

DPSTP:  MOV     R4,-(SP)        ; SAVE THE CALLING CCB
        MOV     RDBF(R5),R3     ; GET RECEIVE DATA BUFFER ADDRESS
        MOV     #DTR,-(R3)      ; DISABLE RECEIVER - LEAVE DTR UP
        CLR     4(R3)           ; DISABLE TRANSMITTER
        MOV     RPRIM(R5),R4    ; GET PRIMARY RECEIVER CCB
        BEQ     10$             ; IF EQ, NONE ASSIGNED
        CALL    $RDBRT          ; RETURN BUFFER TO THE POOL
10$:    CLR     RPRIM(R5)       ; CLEAR PRIMARY POINTER
        MOV     LINE(R5),R4     ; SET SYSTEM LINE NUMBER
        CALL    $RDBQP          ; REMOVE ANY WAIT REQUESTS
        MOV     (SP)+,R4        ; RESTORE THE SAVED CCB
        TST     TPRIM(R5)       ; IS ANYTHING ACTIVE
        BNE     20$             ; YES, SO SAVE FOR TIMEOUT
```

```
        CALL    $DDCCP          ; NO, SO GIVE THE COMPLETION NOW
        BR      30$             ; AND EXIT

20$:    MOV     R4,KICCB(R5)    ; SAVE THE CCB FOR LATER
30$:    SEC                     ; INDICATE ASYNC
        RETURN                  ; AND EXIT

        .END
```

GLOSSARY

Asynchronous Transmission
Transmission in which time intervals between transmitted characters may be of unequal length. Transmission is controlled by start and stop elements at the beginning and end of each character. Also called start-stop transmission.

BDIN
Data Input on the LSI-II bus.

BDOUT
Data Output on the LSI-II bus.

BIAKI
Interrupt Acknowledge.

Bit-Stuff Protocol
Zero insertion by the transmitter after any succession of five continuous ones designed for bit-oriented protocols such as IBM's Synchronous Data Link Control (SDLC).

Bits per Second (b/s)
Bit transfer rate per unit of time.

BIRQ
Interrupt Request priority level for LSI-11 bus.

BRPLY
LSI-11 Bus Reply. BRPLY is asserted in response to BDIN or BDOUT.

BSYNC
Synchronize – asserted by the bus master device to indicate that it has placed an address on the bus.

Buffer
Storage device used to compensate for a difference in the rate of data flow when transmitting data from one device to another.

BWTBT
Write Byte.

CCITT
Comite Consultatif Internationale de Telegraphie et Telephonie – An international consultative committee that sets international communications usage standards.

Control and Status Registers (CSRs)
Communication of control and status information is accomplished through these registers.

Cyclic Redundancy Check (CRC)
An error detection scheme in which the check character is generated by taking the remainder after dividing all the serialized bits in a block of data by a predetermined binary number.

Data Link Escape (DLE)
A control character used exclusively to provide supplementary line control signals (control character sequences or DLE sequences). These are 2-character sequences where the first character is DLE. The second character varies according to the function desired and the code used.

Data-Phone DIGITAL Service (DDS)
A communicaitons service of the Bell System in which data is transmitted in digital rather than analog form, thus eliminating the need for modems.

DIGITAL Data Communications Protocol (DDCMP)
DIGITAL's standard communications protocol for character-oriented protocol.

Direct Memory Access (DMA)
Permits I/O transfer directly into or out of memory without passing through the processor's general registers.

Electronic Industries Association (EIA)
A standards organization specializing in the electrical and functional characteristics of interface equipment.

Full-Duplex (FDX)
Simultaneous 2-way independent transmission in both directions.

Field-Replaceable Unit (FRU)
Refers to a faulty unit not to be repaired in the field. Unit is replaced with a good unit and faulty unit is returned to predetermined location for repair.

Half-Duplex (HDX)
An alternate, one-way-at-a-time independent transmission.

LARS
Field Service Labor Activity Reporting System.

Non-Processor Request (NPR)
Direct memory access-type transfers, (see DMA).

Protocol
A formal set of conventions governing the format and relative timing of message exchange between two communicating processes.

RS-232-C
EIA standard single-ended interface levels to modem.

RS-422-A
EIA standard differential interface levels to modem.

RS-423-A
EIA standard single-ended interface levels to modem.

RS-449

EIA standard connections for RS-422-A and RS-423-A to modem interface.

Synchronous Transmission

Transmission in which the data characters and bits are transmitted at a fixed rate with the transmitter and receiver synchronized.

V.35

(CCITT Standard) – Differential current mode-type signal interface for high-speed modems.

DPV11 Serial Synchronous
Interface User Guide
EK-DPV11-UG-001

Reader's Comments

Your comments and suggestions will help us in our continuous effort to improve the quality and usefulness of our publications.

What is your general reaction to this manual? In your judgement is it complete, accurate, well organized, well written, etc? Is it easy to use? _____

_____

_____

_____

What features are most useful? _____

_____

_____

_____

What faults or errors have you found in the manual? _____

_____

_____

_____

Does this manual satisfy the need you think it was intended to satisfy? _____

Does it satisfy *your* needs? _____ Why? _____

_____

_____

_____

☐ Please send me the current copy of the *Technical Documentation Catalog*, which contains information on the remainder of DIGITAL's technical documentation.

Name _____ Street _____
Title _____ City _____
Company _____ State/Country _____
Department _____ Zip _____

Additional copies of this document are available from:

Digital Equipment Corporation
Accessories and Supplies Group
Cotton Road
Nashua, NH 03060

Attention *Documentation Products*
Telephone 1-800-258-1710

Order No. EK-DPV11-UG

Fold Here

Do Not Tear – Fold Here and Staple

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MERRIMACK, NH

POSTAGE WILL BE PAID BY ADDRESSEE

Digital Equipment Corporation
Educational Services Development & Publishing
Continental Blvd. (MK1/2M26)
Merrimack, N.H. 03054

digital

digital equipment corporation

Printed in U.S.A.

L:11

Storage System Diagnostics and Utility Protocol
AA-L620A-TK

A Part of UDA50 Programmer's Doc. Kit
QP905-GZ

digital software

CHAPTER 1

INTRODUCTION

1.1 Overview of MSCP Subsystem

Mass Storage Control Protocol (MSCP) is the protocol used by a family of mass storage controllers and devices designed and built by Digital Equipment Corporation. In a system that uses an MSCP mass storage subsystem, the controller contains the intelligence to perform the detailed I/O handling tasks. This arrangement allows the host to simply send command messages (for example, requests for reads or writes) to the controller and receive response messages back from the controller. The host does not concern itself with details such as device type, media geometry, media format, error recovery, etc.

The host uses two levels of software to communicate with the mass storage subsystem. They are the class driver and the port driver. The class driver is the higher level and is concerned with tasks being performed. The class driver's concern with details is limited to the general type of device (such as disk) and the capacity. The class driver is not concerned with the communication link (I/O bus), type of controller, or the exact model of device(s) being used.

The port driver is the lower level and is concerned only with communication services such as passing messages on and off of the communications link. The port driver is not concerned with the meaning of the messages, nor is it concerned with the exact type of controller or the exact model of storage unit(s). Thus each driver has its own level of responsibilities and shields the other from unnecessary details.

In the controller architecture, there are also two levels of software. The lower of these two is also a port driver and, like the port driver in the host, is concerned only with passing messages on and off of the bus. The higher level of controller software is the MSCP Server. It constitutes the intelligence of the controller and therefore defines the functionality of the controller.

1.2 Overview of DUP

While these mass storage subsystems use MSCP to perform normal data transfer operations, they use Diagnostic/Utilities Protocol (DUP) to load, start, and monitor diagnostics and utilities in the mass storage controller. This protocol provides the method of communication between the DUP class driver in the host and the DUP server in the controller for such tasks. For example, the host program uses this protocol to request that the controller load a diagnostic or utility program (either supplied by the host or from media local to the controller) and execute it. The host program may communicate with the remote program while it is running, may make inquiries to the controller about the progress of the program, and may terminate execution of the program.

The architecture is illustrated in Figure 1-1.

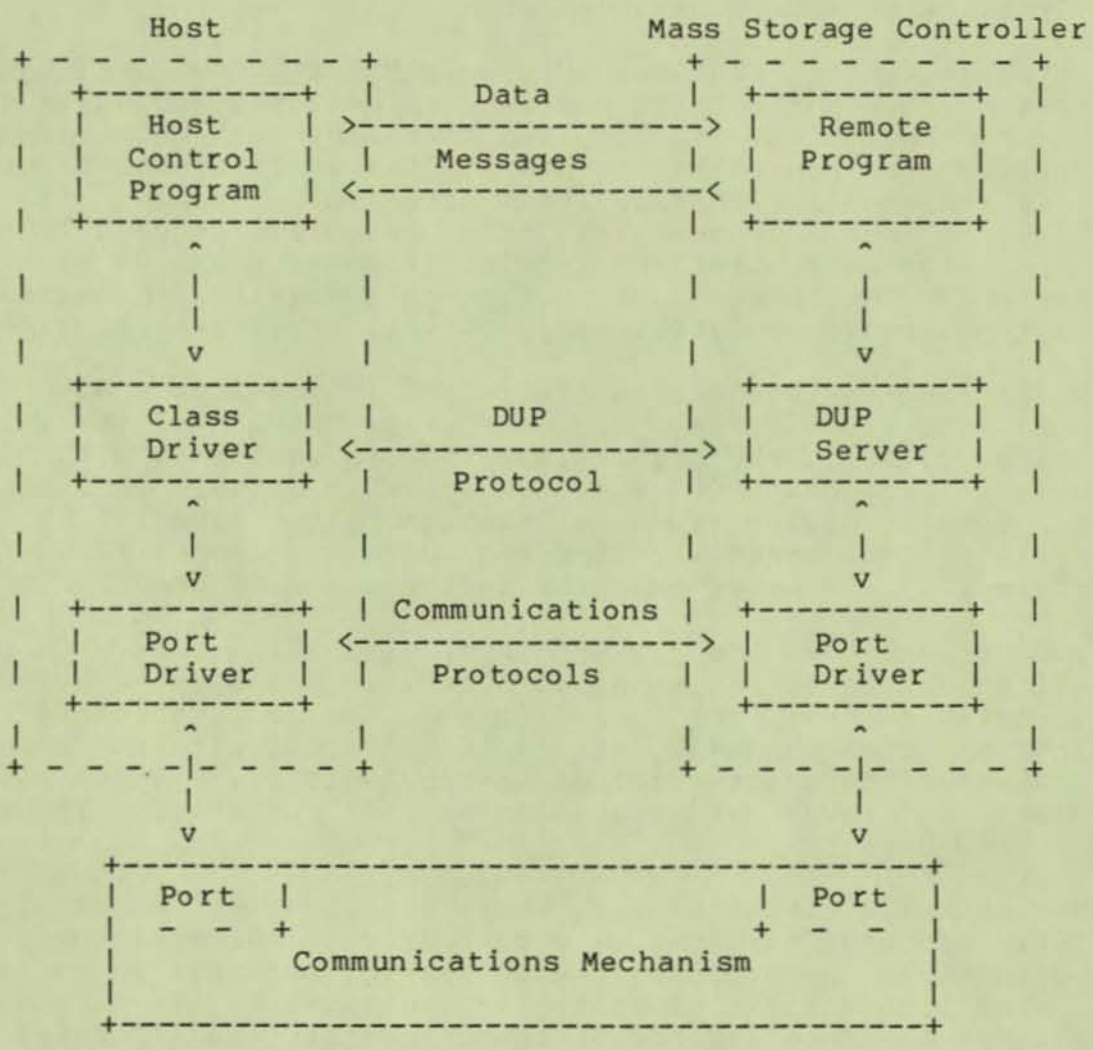


Figure 1-1 Example System

1.3 Purpose

The host class driver and the controller server communicate via a set of commands and responses. The commands and responses used by DUP are explained in this manual to the detail needed by someone writing a host DUP class driver.

1.4 Scope

The scope of this manual is limited to the details of those DUP commands and responses. This manual does not include any information on specific types of host processors or any specific operating systems. It does not assume any particular bus, controller, device type, host, or port driver implementation.

CHAPTER 2

TERMINOLOGY

2.1 Terminology

Controller Timeout Interval

The controller timeout interval is a time interval, measured in seconds, implied by the DUP command being performed or supplied by the DUP server in the GET DUST STATUS or the EXECUTE LOCAL PROGRAM command's end message.

Nugatory

Of little or no consequence: Trifling, Inconsequential.

Remote Program

The remote program is the diagnostic/utility loaded or invoked by the DUP server on behalf of the host.

Host Control Program

The host control program is the host process which loaded or invoked the remote program and provides virtual terminal services for the remote program.

Immediate Commands

Commands that DUP servers should execute immediately, without waiting for any other commands to complete.

Sequential Commands

Commands that DUP servers execute in the exact order that they were received from class drivers. Sequential commands typically allow the transfer of data between the host and the controller.

CHAPTER 3

DUP HOST / CONTROLLER COMMUNICATIONS

3.1 DUP Host / Controller Communications

3.1.1 Communications Scheme

DUP is comprised of two protocols: Class Driver/DUP Server and Host Control Program/Remote Program protocols. The Class Driver/DUP Server protocol forms the "primary" protocol; the Host Control Program/Remote Program protocol is a subprotocol which is imbeded within the SEND/RECEIVE DATA command. Both protocols use the communication mechanism described below.

3.1.2 Class Driver / DUP Server Communications

DUP runs on a dedicated connection, in the Systems Communications Architecture sense, between a host program (DUP Class Driver) and the DUP server. Flow control and connection management are provided by the underlying communications services. DUP is an asymmetric master/slave protocol, with the DUP class driver as master; i.e. DUP messages can be divided into commands and responses, where responses are only issued in response to commands and only the class driver may issue commands.

Host class drivers use the host port driver to communicate with DUP servers in controllers. DUP servers similarly use the controller port driver to communicate with class drivers in hosts. This communication takes place across a link called a connection.

The state of the connection is directly equivalent to the state of the controller or DUP server with respect to the class driver. The controller is "Controller-Online" if and only if the connection is established and functioning. The controller is "Controller-Available" if the connection is not established, but it is believed that it could be established. The controller is "Controller-Offline" if the connection is not established and it is believed that it cannot be established.

Two types of communications services are used across the connection between the DUP class driver and the DUP server.

- A sequential message communications service, used for DUP control messages, is used across the connection between a class driver and a DUP server. This service guarantees sequential, duplicate free delivery for all messages sent across the same connection. This service must support messages of at least 40 bytes in length.

- A block data communication service, used to move data between the host and the controller. This service provides a method of transferring the contents of a named buffer between the host and the controller or transfering data from the controller to a named buffer in the host. Buffers are identified by buffer descriptors which are provided by the host.

The communications mechanism or port drivers discard all messages that, at the time a connection is terminated, have been sent or queued to be sent via the message services but have not yet been delivered.

Besides using the sequential message communications service directly, DUP uses the establishment of the connection itself to synchronize the class driver and DUP servers. The class driver will terminate the connection if it determines that it must re-synchronize with the DUP server. The DUP server may terminate the connection or it may enter the "Idle" state if re-synchronization with the class driver is necessary. Events that require re-synchronization include certain errors or loss of context by either process. The connection is also terminated, by a port driver, if an unrecoverable communications error occurs. Termination of the connection signals the processes that re-synchronization is necessary. The re-synchronization is accomplished by each process discarding all context regarding outstanding commands or transactions, after which a new connection is established.

Following re-synchronization, commands which were outstanding before the re-synchronization was performed may have completed to an indeterminate extent. Such commands may have never been started, may have been partially completed, or may have been fully completed. The only guarantee is that they are no longer outstanding, implying that the controller is no longer performing work for them and that the class driver will not receive an end message for them. The fact that the controller is no longer performing work for them implies that no state changes or modification of data will take place as a result of such commands.

Especially critical to DUP is the concept of flow control and the flow control based requirements that are imposed on DUP class drivers and DUP servers. Flow control and the flow control requirements are discussed in detail in the MSCP Specification, "Class Driver / MSCP Server Communications".

In general, flow control arises from the need to avoid the congestion and/or deadlock which can occur if one process sends messages too quickly to another process. The receiving process must have buffers in which to place the incoming messages. When all such buffers are full, additional messages cannot be handled.

The sequential message communications service does use flow control. When a potential receiving process queues a buffer for receiving messages on a connection, the presence of this buffer is communicated, via the underlying communications service, to the potential sending process at the other end of the connection. This message notifying the potential sending process of the queued buffer grants the sending process a credit, which is the priviledge to send a message. Therefore messages will only be sent when the sending process knows that the receiving process has queued a buffer into which the message can be received, ensuring that the receiving process will be able to handle the message.

Note that the DUP class driver may communicate with only one DUP server per connection. However, the DUP class driver may be connected to as many DUP servers, each on a different connection, as there are DUP servers in the controller.

3.1.3 Host Program / Remote Program Communcations

The Host Program/Remote Program communications uses the block data communication services for transfer of messages and data. The details of the Host Program/Remote Program subprotocol are contained in the description of the SEND/RECEIVE DATA command.

CHAPTER 4

ALGORITHIMS AND USAGE RULES

4.1 Algorithms and Usage Rules

4.1.1 DUP Server States

The DUP server may be in any of three states relative to the DUP class driver. Each DUP server may be in a different major state relative to the class driver. The states are listed below.

Offline

The DUP server is "Offline" to the class driver whenever it is not available to that class driver and cannot perform any operations on its behalf. Possible causes include inoperative hardware or an operator disabling the controller. A server is "Offline" when it is not possible to establish a connection between the class driver and the server.

Available

A DUP server is "Available" to the class driver whenever it could perform operations for that class driver but the driver has not yet synchronized with the server. A server is "Available" exactly when it would be possible to establish a connection between the class driver and the server, but no connection has yet been established.

Online

A DUP server is "Online" to the class driver whenever it can both perform operations for that class driver and the driver has synchronized with the server. A server is "Online" exactly when a connection exists between the class driver and the server; this is the state used for normal operation.

Additionally, a DUP server has two substates relative to its class driver when it is "Online".

IDLE

A DUP server is "Idle" when it is not monitoring the operation of a remote diagnostic/utility.

ACTIVE

A DUP server is "Active" when a remote diagnostic/uility is operating on behalf of the host.

The states described above actually exist between an individual DUP class driver and an individual DUP server. A host will have only one DUP driver and a subsystem may have several DUP servers. Note also that the DUP server is distinct from the state of any units connected to the controller.

A DUP server enters the "Controller-Offline" state relative to a host whenever the DUP server ceases to function or otherwise becomes unable to perform operations for the host. Possible causes are listed below.

1. Controller hardware, software, or power failure.
2. Controller initialization, either requested or spontaneous.
3. An operator (typically Field Service) disables all or part of the controller.
4. Communications mechanism failures.

A DUP server enters the "Controller-Available" state relative to a host class driver under the following conditions.

1. The controller or DUP server is "Controller-Offline", and all causes of it being "Controller-Offline" are removed.
2. The DUP server is "Controller-Online", and the DUP server cannot successfully send a control message (i.e., a DUP end packet) to the host class driver.
3. The DUP server is "Controller-Online", and the host access timeout expires (see Section "Host Access Timeouts").
4. The host class driver terminates the connection between the class driver and the DUP server.
5. A port driver or the communications mechanism terminates the connection between the class driver and the DUP server, generally due to a communications error.

The port driver should inform the class driver whenever the DUP server enters the "Controller-Available" state. How the port driver obtains this information is communications mechanism dependent. Note that the notification that the controller has become "Controller-Available" is not necessarily prompt. In particular with some communications

mechanisms, the notification may not occur until the next time the class driver issues a command to the controller. Furthermore, the port driver need not notify the class driver at all if a compound (multiple) error is associated with the DUP server becoming "Controller-Available". In such a case, the class driver will ultimately become aware of the state change when its command timeout expires.

Since no connection exists to a DUP server that is "Controller-Offline" or "Controller-Available", the communications mechanism will either reject or discard any messages (commands) that a class driver attempts to send to it. An DUP server that becomes "Controller-Offline" or "Controller-Available" may either abort commands in progress or else continue processing the commands that it has already received.

Typically, the DUP server will continue processing outstanding commands until it "notices" that the connection to the class driver has been terminated, at which point it will abort any commands still outstanding and enter the "Idle" state.

The DUP server enters the "Controller-Online" state relative to a host class driver upon successful synchronization with the class driver. The class driver synchronizes with the DUP server by establishing a connection with the DUP server. Note that the DUP server must guarantee that there are no outstanding commands "leftover" from a previous incarnation of the connection before it allows the new incarnation of the connection to be established and enters the "Controller-Online" state.

4.1.2 Command Categories and Execution Order

Most DUP commands are only legal in a particular state. The GET DUST STATUS command is the only command legal in both IDLE and ACTIVE states. Commands received while the DUP server is in an inappropriate state will be returned with the generic response status of INVALID COMMAND. The mechanism used to return the DUP server to IDLE state from the host side is the ABORT command. The mechanism used from the remote program is implementation-dependent.

A DUP server enters the ACTIVE from the IDLE state as a result of successfully performing either an EXECUTE SUPPLIED PROGRAM or EXECUTE LOCAL PROGRAM command. Returning to the IDLE state from the ACTIVE state is a result executing an ABORT PROGRAM command or as a result of remote program termination. Because the termination notification is sent only to the host control program as part of the SEND/RECEIVE DATA subprotocol, the class driver must poll the DUP server to determine if the server has returned to the IDLE state. The class driver must also determine the state of the DUP server when it receives a request from the host control program to execute a remote program as the server may have made an undetected transition to the IDLE state. Additionally, if the class driver notices that the remote program's progress indicator is not being updated, the class driver must look at the DUP

server's state indicator before deciding that the server is no longer sane.

4.1.3 Class Driver / DUP Server Synchronization

Synchronization of a class driver with a DUP server is accomplished by establishing or re-establishing the connection between the class driver and the DUP server. When the connection is established or re-established, the DUP server aborts or otherwise terminates all commands that are outstanding from that class driver. This forces the dialogue between the class driver and DUP server to a known synchronized state, namely that of having no outstanding commands. After establishing the connection, the class driver can issue commands without worrying about duplicating command reference numbers or other unfortunate side effects. Note that synchronizing with the DUP server, if successful, causes the DUP server to become "Controller-Online".

As stated above, the main purpose of synchronization is to guarantee that there are no outstanding commands, thus forcing the dialogue between the class driver and DUP server to a known state. DUP servers must ensure that this guarantee is met before they allow synchronization to complete (i.e., before they become "Controller-Online"). In particular, DUP servers must guarantee that no end messages will be sent and their state or context changed for any commands that were issued on an earlier incarnation of the connection between the class driver and DUP server.

Class drivers must synchronize with the DUP server whenever the host boots, recovers from a power failure, loses context, or is recovering from certain errors. After synchronizing with a DUP server, the class driver should issue a GET DUST STATUS command to establish the characteristics of the DUP server.

4.1.4 Class Driver Error Recovery

The principle method of error recovery used by class drivers is to re-synchronize with the DUP server, as described in the preceeding section. All communications mechanism failures and many controller failures are reported by terminating the connection between the class driver and DUP server, in response to which the class driver should attempt to re-synchronize with the DUP server. If the class driver decides that the controller is insane, either because the class driver received an invalid message or because a command timed out, it should recover by re-synchronizing with the DUP server. Similarly, if the DUP server decides that the class driver is insane, it enters the "Idle" state and may terminate the connection to the class driver. If the class driver is in fact actually sane, it will re-synchronize with the DUP server after the port driver notifies it that the circuit has been terminated.

4.1.5 Command Timeouts

Whenever a host issues a DUP command packet, it should time out the receipt of the corresponding response. The appropriate timeout interval for each DUP command is given in the description of the command. The DUP server may timeout receipt of host commands with a 30 second timeout and the remote program may time out the reception of SEND DATA and RECEIVE DATA commands if they desire.

Host class drivers use command timeouts to guarantee that all controller or communications mechanism failures will be detected. The failures detected by command timeouts include partially sane or deadlocked controllers, which may continue to process new commands even though one or more old commands have been lost and will never complete.

The DUP server is sane if and only if the utility or diagnostic which was initiated will ultimately complete. For practical purposes, the term "ultimately" must be replaced with the phrase "within reasonable time". What constitutes a "reasonable time" varies with the complexity of the requested diagnostic/utility. The difficulty of the host DUP class driver having to derive this "reasonable time" can be eliminated by re-stating the definition of a sane DUP server as follows: The server is sane if and only if the progress indicator is being incremented within some reasonable time. This definition allows setting "reasonable time" to some fixed value and varies the units in which we measure "useful work" according to the complexity of the diagnostic/utility. Command timeouts are based on this second definition.

A class driver implements the command timeout mechanism as follows. For each DUP server to which it is "Controller-Online", the class driver monitors the progress indicator to insure that it is changing.

The class driver must never use a time interval that is shorter than the controller specified controller timeout interval for its command timeout determination, although the class driver may use a time interval that is longer than the one specified by the controller. The controller timeout interval specified by the DUP server must not be larger than 4 minutes and 15 seconds (i.e., 255 seconds).

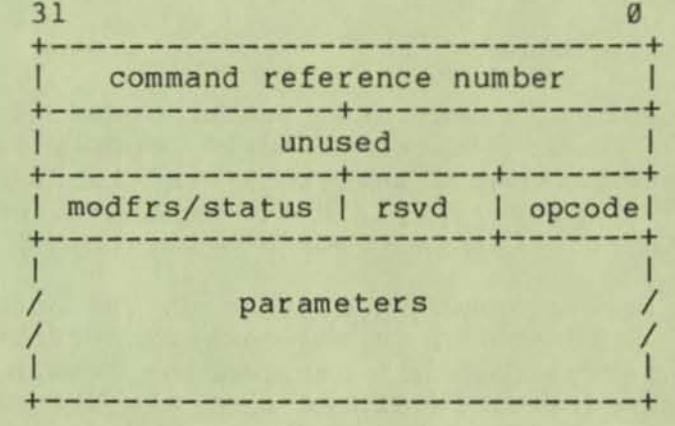
CHAPTER 5

GENERIC CONTROL MESSAGE FORMAT

5.1 Generic Control Message Format

5.1.1 Generic Control Message Format

All DUP control messages consist of a 12 byte header and a 28 byte or shorter parameter area. Multi-byte numbers are stored least significant byte first (i.e., using the standard VAX number formats). Messages are laid out as follows:



The length of the parameter area varies depending upon the opcode.

The communications mechanism conveys both the text of a message and its length. The receiver of a message uses its length to verify that all required parameters are in fact present. The communications mechanism may restrict the allowable message lengths. For example, it might require that all messages have a fixed length 48 bytes or that the length be an even multiple of 4 bytes. For this reason the message lengths defined by DUP are minimum lengths. Senders may pad messages as necessary to meet communications mechanism length restrictions. The contents of the padding is reserved and must follow the rules for reserved fields defined in the following paragraphs.

Class drivers must supply the value zero in the reserved fields of all messages (commands) that they send to a controller and must also ignore the contents of reserved fields in all the messages (end messages and attention messages) that they receive from an DUP server.

DUP servers must supply the value zero in the reserved fields of all end messages that they send to class drivers. DUP servers must either ignore the contents of reserved fields in the messages (commands) that they receive from class drivers or verify that the contents are zero. The command is treated as invalid if the contents are non-zero.

Whether or not a DUP server verifies that reserved fields are zero is controller dependent and need not be consistent for all reserved fields.

Note that the above two cases are the only allowable controller behavior for reserved fields with non-zero values. That is, if a controller may possibly respond differently depending on whether or not a reserved field is zero, then it must treat the command as invalid if the reserved field is non-zero. The only exception is reserved command message fields that correspond to end message fields in which the controller should return zeros. For such fields, the controller may merely echo the corresponding reserved fields from the command message, trusting the host to have zeroed them rather than checking and/or explicitly zeroing the fields.

A field, as used in the above discussion, may have any length. In particular, it may be an individual bit of a flag word or byte as well as an entire byte, word, or whatever. The fields in the message header are interpreted as follows shown below.

command reference number

A 32 bit, unique, non-zero number used to identify host commands. Class drivers should supply a unique reference number in each command that they send to a DUP server. A class driver may supply a zero reference number if it does not need to associate a command with its end message.

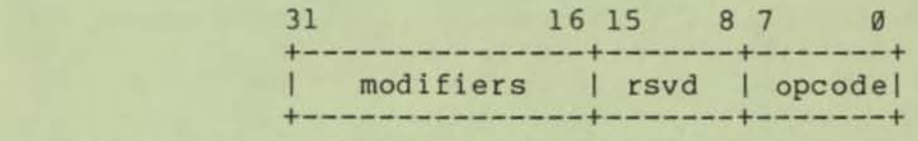
Command reference numbers must be unique across all commands that are outstanding on the same connection. That is, they must be unique across all outstanding commands issued by a single class driver (host) to a single DUP server. The class driver may re-use a command's reference number when the command is no longer outstanding -- i.e., after receiving the command's end message or after re-synchronizing with the DUP server. Command reference numbers need not be unique for commands issued by different class drivers -- i.e., commands issued by different hosts or commands for different DUP servers from the same host. Therefore, controllers must internally use the combination of a command reference number and the connection on which the command was received as the unique identifier of an outstanding command.

opcode

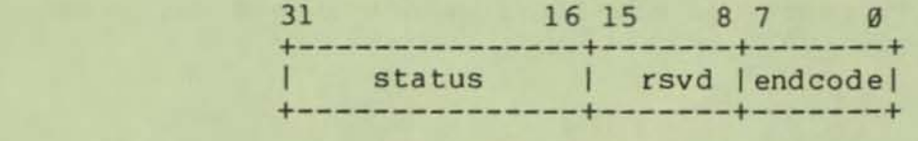
Identifies the meaning or purpose of the message. In messages sent from a class driver to a DUP server, this field specifies the operation or command to be performed. In messages sent from the controller to the class driver, this field specifies whether this is an end message or an attention message. The opcode of an end message also identifies the type (opcode) of the command to which the end message corresponds. A message's opcode implicitly specifies the length and format of the message, including the interpretation of any parameters that are present.

modifiers or status

This field has different formats in command messages and end messages. In command messages this field has the following format:



The "modifiers" field contains bit flags that modify the operation identified by "opcode" or zero if no modifiers are specified. In end messages, this field has the following format:



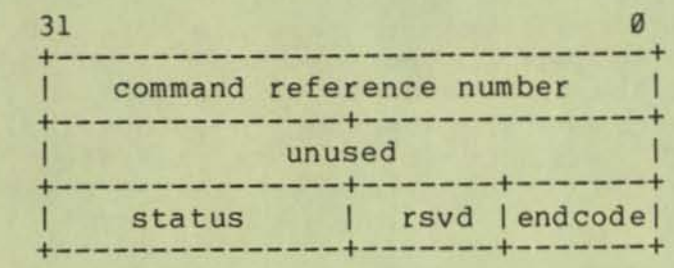
The "status" field identifies the completion status of the command. The "flags" field contains bit flags, called end flags, that report certain conditions that are disjoint from normal completion status of a command. These fields are further described in Section 5.3.

5.1.2 Command Modifiers

The allowable modifiers on a command are command (opcode) dependent. The individual command descriptions list the allowable modifiers for each command. All modifiers that are not explicitly allowed for a command are reserved and must be treated in accordance with the requirements for reserved fields described in Section 5.1. Modifiers that are only allowed on one command are described in that command's description.

5.1.3 End Message Format

A DUP server sends an end message to a class driver to report completion of a command. The generic end message format is as follows:



The command reference number is copied from the command message. The remaining fields are as follows shown below.

endcode

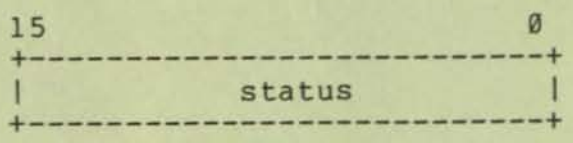
The endcode identifies this message as an end message and the type of command (opcode) that this is an end message for. This field implicitly specifies the format and interpretation of the parameters.

status

The modifiers field is used for a completion status code. The status code indicates whether the operation was successfully completed or, if it wasn't successful, what type of error occurred.

5.1.4 Status Codes

The "status code" field is a 16-bit field as follows:



The status codes that may be returned in end message "status code" fields are listed below along with the general use made of these codes. The actual codes used are listed in Appendix B. The codes are also listed in the descriptions of the commands that may return them.

Success

The command was successfully completed.

The status code value associated with "Success" is, by definition, zero.

Invalid Command

Used to report conditions such as the state of the DUP server being incorrect for the command issued (e.g., Abort Program command issued to a server in the IDLE state) or that the command is inappropriate for the particular DUP server (eg, Execute Local Program command issued to a DUP server which does not support this feature).

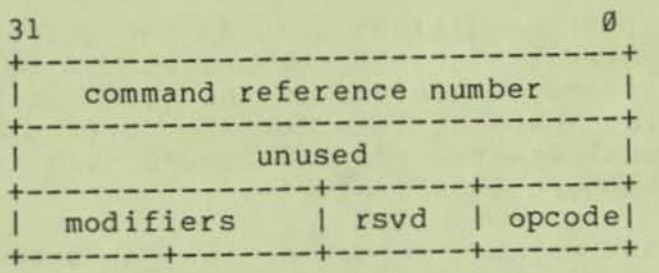
The status codes that may be returned for a specific command are command (opcode) dependent. The status codes that may be returned for each command and any special meaning that they have specific to the command are listed in the command descriptions. Note that the format of a command's end message is solely determined by its opcode. The status code returned in the end message does not affect the end message's format.

5.1.5 ABORT PROGRAM Command / Response

Command Category:

Immediate

Command message format:

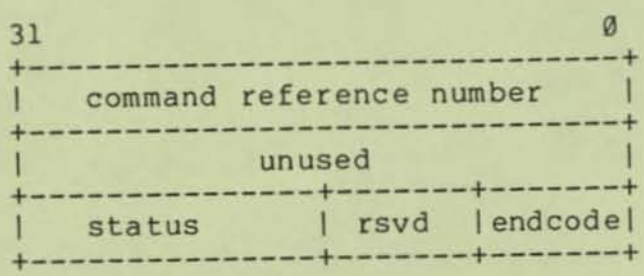


Allowable modifiers:

none

Command Opcode: 6

End message format:



Status codes:

Success
Invalid Command (server is not in ACTIVE state)

Description:

The ABORT PROGRAM command is used to terminate the execution of a remote program in an orderly fashion. When a SUCCESSFUL response is received to this command the remote program has stopped executing and the server is in IDLE state. Note that the sending of this command does not preclude further SEND DATA or RECEIVE DATA exchanges. On the contrary, the remote program may be designed to send out termination status and possibly even ask questions during its forced-exit sequence. The timeout for this command is a fixed 10 seconds, and if a response is not received

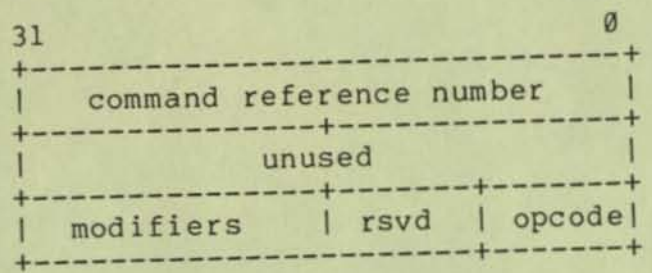
by then, the connection to the DUST should be terminated. This command is only legal if the DUST is in ACTIVE state.

5.1.6 GET DUST STATUS Command / Response

Command Category:

Immediate

Command message format:

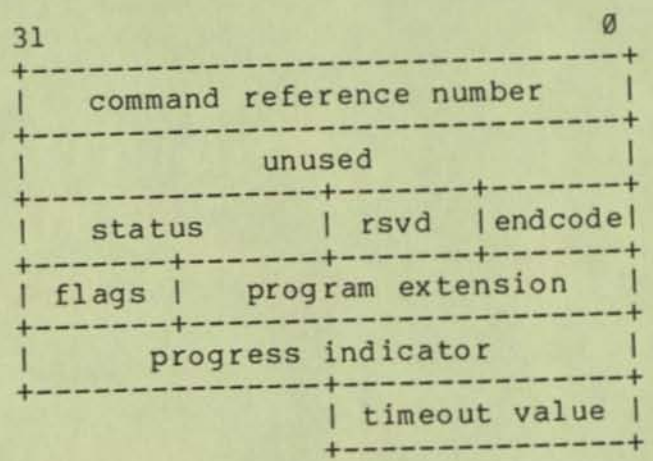


Allowable modifiers:

none

Command opcode: 1

End message format:



program extention

Extension (in ASCII) for down-line loadable programs (see appendix B).

flags

Bit Ø Set if any program execution under this server disables the operation of all other servers in the same controller.

Bit 1 Set if this controller has a local load media for loading diagnostics and utilities.

Bit 2 Set if this server will not accept the EXECUTE SUPPLIED PROGRAM Command.

Bit 3 Set if this server is currently in ACTIVE state.

progress indicator

Progress indicator for the currently running remote program; see description of the SEND DATA and RECEIVE DATA commands.

timeout

Timeout to use for the EXECUTE LOCAL PROGRAM command, in seconds; only pertinent if bit 1 of the 'flag' byte is set.

Status Codes:

Success

Description:

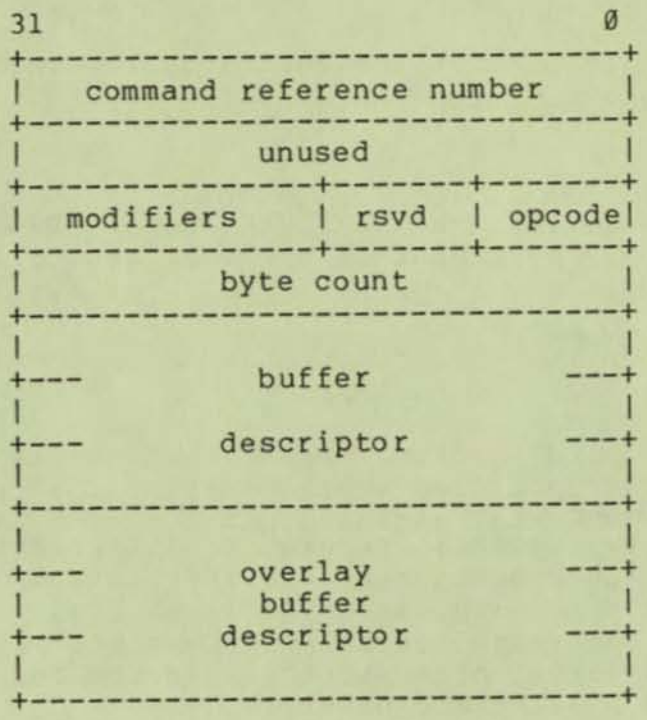
This command alows the host program to interrogate the DUP server to determine its characteristics, its state and the state of the program currently running, if any. It is legal in either IDLE or ACTIVE state and does not affect the state of server. It has a fixed timeout interval of 3 seconds. If the response times out, the host should break the connection.

5.1.7 EXECUTE SUPPLIED PROGRAM Command / Response

Command Category:

Immediate

Command message format:



byte count

Byte count for initial transfer (from bytes 0-3 of the program header)

buffer descriptor

Buffer Descriptor for initial load. This field contains the address of byte 0 of the program header.

overlay buffer descriptor

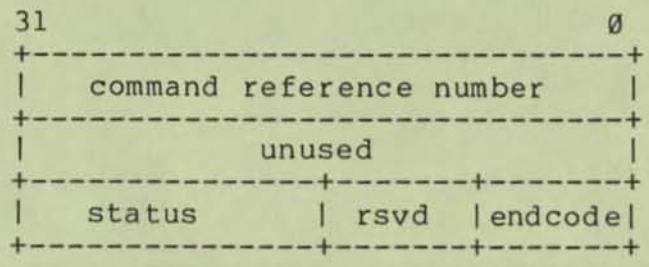
Buffer Descriptor for overlays, if required. This field contains the address of byte 0 of the overlay.

Allowable modifiers:

none

Command Opcode: 2

End message format:



Status codes:

Success
Invalid command (Server is not in IDLE state)
No region available
No region sutable
Host buffer access error (Unibus error or invalid byte count)

The format of the Buffer Descriptors shown here is dependent upon the underlying communications mechanism.

Receipt of a SUCCESSFUL response to this command means that the host may retire the buffer specified by the initial load Buffer Descriptor. The overlay buffer MUST stay assigned until execution of the remote program has terminated. Receipt of any response other than SUCCESSFUL means that both buffers may be retired.

DUP servers which do not support EXECUTE SUPPLIED PROGRAM may always return a response of INVALID COMMAND to this command.

Description:

This command causes the server to transfer the program from host memory to an area in the controller and start its execution. The host supplies the address and length (in bytes) of a buffer containing the program header (see Appendix B) and initial load. The starting address of the program, its memory requirements, and any relocation information needed to run under the server are in the program header in a format which is none of the host's business. This command is only legal when the server is in the IDLE state and return of a SUCCESSFUL end packet puts the server into the ACTIVE state.

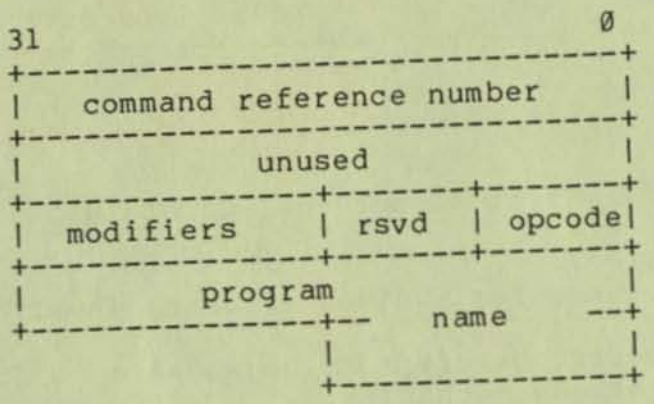
The timeout for this command is 30 seconds.

5.1.8 EXECUTE LOCAL PROGRAM Command / Response

Command Category:

Immediate

Command message format:



Program name

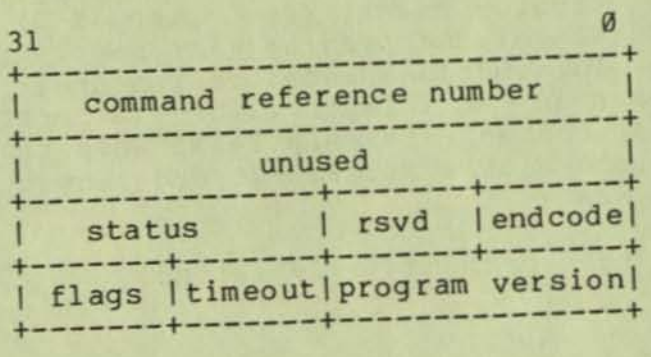
Name of local program, in ASCII, space-filled on right

Allowable modifiers are shown below.

Allow standalone - If not set, programs which have the STANDALONE characteristic will not be executed.

Command Opcode: 3

End message format:



program version

Program Version number (16 bits, binary)

timeout

Timeout value for SEND DATA and RECEIVE DATA commands in seconds. If Ø, these commands are not timed out.

flags

Flags byte of program characteristics (see appendix B)

Status codes:

Success
Invalid command (server is not in IDLE state)
No region available
No region suitable
Program not known (no such program on media)
Load failure (input error while loading program)
Standalone (STANDALONE modifier not specified for a standalone program)

** Note: servers which indicate that they do not support local programs always return a response of INVALID COMMAND to EXECUTE LOCAL PROGRAM commands.

Description:

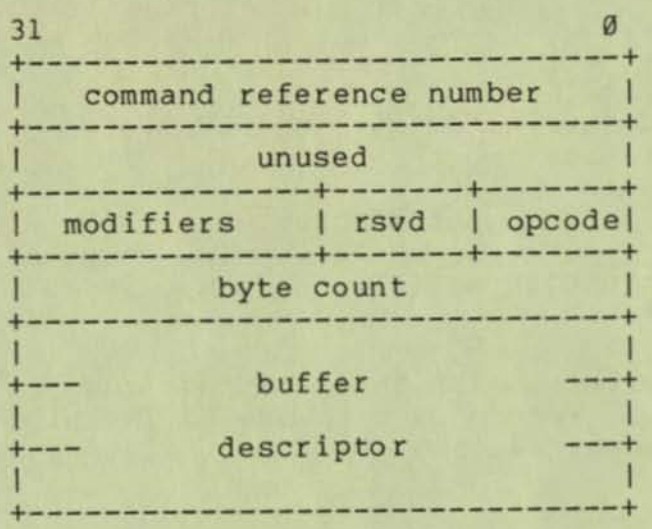
Receipt of this command causes the controller to search its local media for the named program, load it and execute it. Receipt of a SUCCESSFUL response by the host means that the program is executing and the server is in the ACTIVE state. This command is only legal when the server is in the IDLE state. The timeout value for this command is specified in the GET DUST STATUS response.

5.1.9 SEND DATA and RECEIVE DATA Commands / Responses

Command Category:

Sequential

Command message format:



byte count

byte count for transfer

buffer descriptor

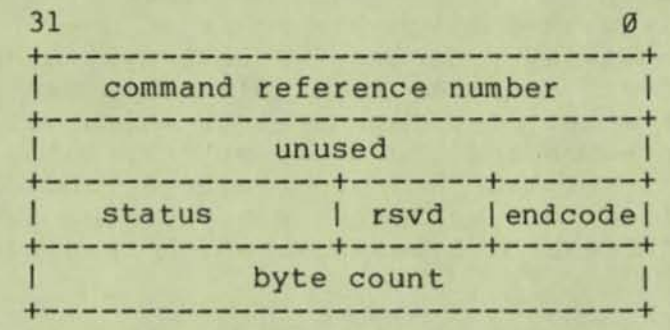
Buffer Descriptor for initial load. This field contains the address of byte 0 of the program header.

Allowable modifiers:

none

Command Opcode: 4 SEND DATA
Command Opcode: 5 RECEIVE DATA

End message format:



byte count

Number of bytes actually transfered

Status codes:

Success
Invalid Command (server is not in ACTIVE state)

Description:

These commands are used to communicate between the initiating host program and the remote program. Both commands specify a host buffer descriptor and a byte count. In the case of SEND DATA, the information in the buffer is read by the remote program and a SEND DATA response sent back to the host to acknowledge receipt. In the case of RECEIVE DATA, the remote program writes data into the buffer up to the amount specified by the byte count and then sends a RECEIVE DATA response to the host to notify it of the transmission.

The SEND DATA and RECEIVE DATA commands are only legal when the server is in the ACTIVE state. If the remote program terminates abnormally, putting the server back in the IDLE state, outstanding SEND DATA and RECEIVE DATA commands may be lost. In the event that the specified timeout interval is exceeded, the host program should issue a GET DUST STATUS command to see if the remote program is still running (ie, the DUP server is active). If it is, the Progress Indicator should be remembered and the timeout interval should be re-instated. If the second timeout expires without a response and a second GET DUST STATUS shows the remote program having made no progress in the interim (i.e. the Progress Indicator has not increased), the program should be considered broken and should be aborted.

The SEND DATA and RECEIVE DATA commands provide a general, full-duplex mechanism for communication between the host control program and a remote program under the DUP server, with the host program controlling the data flow (under the constraints of the underlying flow control mechanism). In many cases, however, it is

the remote program which wants to be "in control" of the conversation since it is where the work is being done. In addition, in any system which has at least one local terminal, the diagnostic and utility programs for that system have an existing command interface to that terminal. This generally consists of the printing of a set of prompt strings which elicit the accepting of a set of corresponding input parameters, followed by the output of the program's results and/or progress indicators. The DUP protocol provides an optional sub-protocol embedded within the SEND DATA and RECEIVE DATA messages which provides the features shown below.

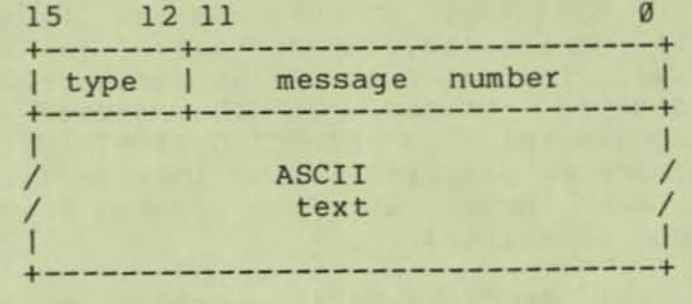
1. Remote programs written to interface to a terminal can be run transparently over a DUP connection.

2. The existence of a single Dialog Driver on the host side gives terminal users on that host access to all remote utilities and diagnostics utilizing the Standard DUP Dialog.

3. Host programs running without supervision by a user on a terminal need not analyze strings of ASCII text in order to hold up their end of the dialog if they are familiar with the remote program's input requirements.

The DUP Dialog is record-oriented and driven by the remote program. In order to embed this in the DUP protocol, the sequence of commands which the host program may issue must be restricted. The host program must issue a RECEIVE DATA with a byte count >= 80 upon receipt of the response to the command which initiated execution of the remote program. When the response to that RECEIVE DATA is received, the host program examines a type field in the received data and, depending on the value of the field, issues either another RECEIVE DATA or a SEND DATA followed by a RECEIVE DATA.

The format of messages from the remote program to the host program, which appear in the buffer specified in RECEIVE DATA commands, is as follows:



The ASCII text from the remote program to the host may include printable characters, including space, and the carriage return-line feed combination ONLY. No non-printable characters, escape sequences, or control characters are allowed.

Where Type is one of the following codes:

TYPE

1 QUESTION - The ASCII text is a prompt for information. The host program must issue a SEND DATA with the answer to the question in a form that the remote program understands. The message number uniquely identifies the question and the content and format of the answer to host programs which know the characteristics of the remote program. The host response must be in ASCII.

2 DEFAULT QUESTION - The DEFAULT QUESTION message is identical to the QUESTION message except that a null (zero-length) SEND DATA is taken to be a default answer to the question. If the host program is answering questions via message number and does not recognize the message number of a DEFAULT QUESTION message, it should respond with a null SEND DATA as its answer. This message type allows remote programs to add new capabilities requiring new user input parameters and still communicate with host programs which know only about an old version. The host response must be in ASCII.

3 INFORMATION - The ASCII text is an informative message. The message number uniquely identifies the type of information being transmitted. The INFORMATION message has its own message number space. The host program should issue another RECEIVE DATA command.

4 TERMINATION - The ASCII text is a normal termination message. The message number uniquely identifies the type of information being transmitted. The TERMINATION message has its own message number space. No further SEND DATA or RECEIVE DATA commands should be issued. Message number 1 is reserved to mean "simple termination" and does not have any ASCII text. Minimal-memory remote programs may omit the ASCII text on all TERMINATION messages.

5 FATAL ERROR - The ASCII text is a fatal error message. The FATAL ERROR message has its own message number space. No further SEND DATA or RECEIVE DATA commands should be issued. Minimal-memory remote programs may omit the ASCII text on all ERROR messages.

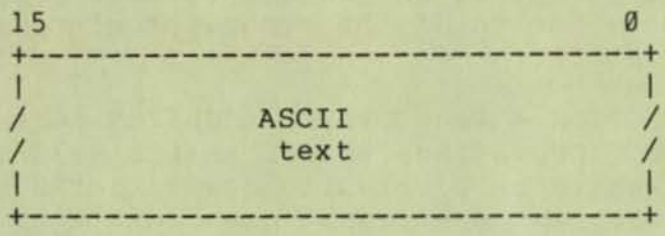
6 SPECIAL - This type is used when only a host program could respond. The message number indicates the type of special message. The data field is type dependent and not necessarily ASCII text.

Note that use of this message type is program dependent. Reference the remote program's functional specification. This message type should ONLY be used in cases where ONLY a program could respond.

Types Ø and 7-15 are reserved.

Message numbers are unique within type for a given program. Other programs may have the same message numbers.

The format of messages from the host program to the remote program, which are placed in the buffer specified in SEND DATA commands, is as follows:



APPENDIX A

MODIFIER CODES/RESPONSE STATUS CODES/OPCODES

The following are the opcodes for DUP commands:

GET DUST STATUS	-	1
EXECUTE SUPPLIED PROGRAM	-	2
EXECUTE LOCAL PROGRAM	-	3
SEND DATA	-	4
RECEIVE DATA	-	5
ABORT PROGRAM	-	6

The following are the modifier codes for DUP commands:

ALLOW STANDALONE	-	1 (bit Ø)

The following are the status codes for DUP responses:

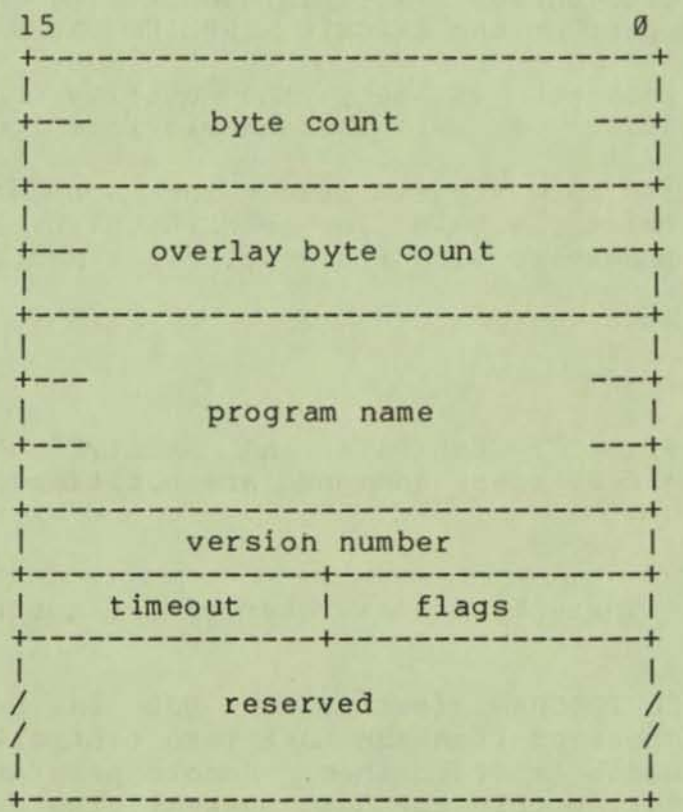
SUCCESSFUL	-	Ø
INVALID COMMAND	-	1
NO REGION AVAILABLE	-	2
NO REGION SUITABLE	-	3
PROGRAM NOT KNOWN	-	4
LOAD FAILURE	-	5
STANDALONE	-	6
HOST BUFFER ACCESS ERROR	-	9

The response codes returned by the controller have a value of 2ØØ octal plus the value of the opcode in the originating command. The only exception is that the response to an invalid command may have a value of 2ØØ octal instead of 2ØØ plus the originating command opcode. This is controller dependent.

APPENDIX B

REMOTE PROGRAM HEADER

Remote program images may reside either in the host or on media local to the controller. The host program may obtain certain of the characteristics of these programs in order to aid it in starting their execution and communicating with them when they are running. In the case of controller-local programs some of these characteristics are passed to the host in the response to the EXECUTE LOCAL PROGRAM command. In the case of host-resident program images, the program image starts with a fixed set of bytes, the program header, which gives the program's characteristics as follows. This header is added to the program via a host-dependent mechanism. The information in this header is used by the host to issue the EXECUTE SUPPLIED PROGRAM command.



byte count

Number of bytes in initial load plus the program header. The initial load image immediately follows the program header in the program image. The initial load image may contain an extended header with DUP-specific information.

overlay byte count

Number of bytes in overlay area. The overlay area immediately follows the initial load image in the program image.

program name

Program name (ASCII), space-filled on right

version number

Program version number, binary

flags

Bit Ø Set if this program is standalone.

Bit 1 If this bit is set, the program needs overlays from host memory during execution. The host must pass a second Buffer Descriptor describing the overlay buffer as part of the EXECUTE SUPPLIED PROGRAM packet.

Bit 2 If this bit is set, the overlay buffer should be writeable as well as readable from the controller.

Bit 3 Set if this program uses the standard DUP dialogue embedded within the SEND/RECEIVE DATA messages to communicate with the host. (see Section 5.Ø)

Bits 4-7 Reserved

timeout

Timeout value for SEND DATA and RECEIVE DATA commands in seconds. If Ø, these commands are not timed out.

reserved

Reserved. These bytes are reserved for future use.

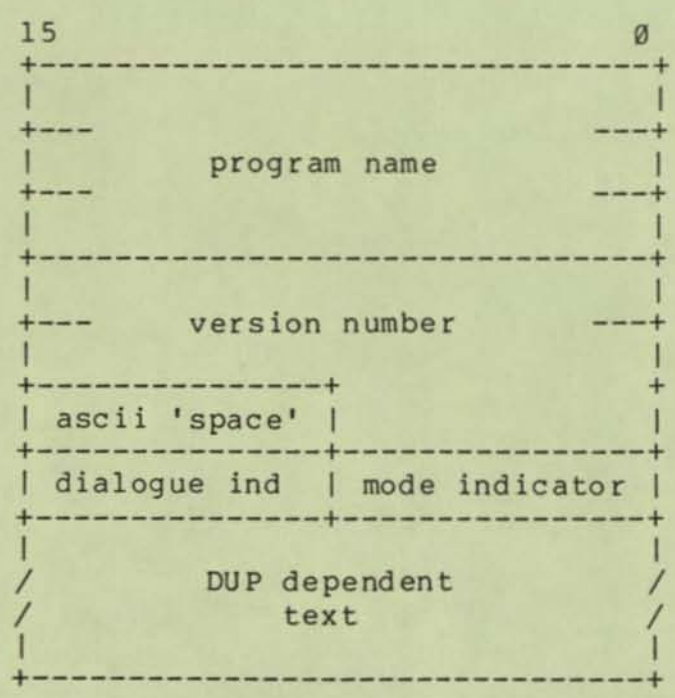
The EXECUTE SUPPLIED PROGRAM feature of DUP is provided to load utilities and diagnostics from the host into controllers which do not have enough local media to store them. Remote programs should be kept in a fixed directory within the host system. The extension of those down-line load files which will run on a given controller is specified

in the GET DUST STATUS Response.

APPENDIX C

THE DIRECT PROGRAM

DUP servers which desire to provide host programs with a directory of local programs should do so using a local program named DIRECT which uses the standard DUP Dialogue to report the directory. The directory is reported as a series of INFORMATION messages, all with a message number of 1 of the following form:



program name

program name (in ascii), right-filled with spaces

version number

Program version number (in ascii), left-filled with spaces

Space

ASCII blank

mode indicator

ASCII 'S' if program is STANDALONE, otherwise and ASCII space

dialogue indicator

ASCII 'D' if program uses standard DUP Dialogue, else ' '

DUP dependent text

Extra DUP-dependent informative text.

These INFORMATION messages may have other INFORMATION messages interspersed with them as long as those other messages have a message number other than 1. The stream of messages should end with a TERMINATION message with a message number of 1.

Storage System Unibus
Port Description
AA-L621A-TK

A Part of UDA50 Programmer's
Doc. Kit
QP905-GZ

digital
software

CHAPTER 1

INTRODUCTION

1.1 Overview of MSCP Subsystem

Mass Storage Control Protocol (MSCP) is the protocol used by a family of mass storage controllers and devices designed and built by Digital Equipment Corporation. In a system that uses an MSCP storage subsystem, the controller contains intelligence to perform the detailed I/O handling tasks. This arrangement allows the host to simply send command messages (requests for reads or writes) to the controller and receive response messages back from the controller. The host does not concern itself with details such as device type, media geometry, media format, error recovery, etc.

The host uses two levels of software to communicate with the mass storage subsystem. They are the class driver and the port driver. The class driver is the higher level and is concerned with tasks being performed. The class driver's concern with details is limited to the general type of device (such as disk) and the capacity. The class driver is not concerned with the communication link (I/O bus), type of controller, or the exact model of device(s) being used.

The port driver is the lower level and is concerned only with communications services such as passing messages on and off of the communications link. The port driver is not concerned with the meaning of the messages, nor is it concerned with the exact type of controller or the exact model of storage unit(s). Thus each driver has its own level of responsibilities and shields the other from unnecessary details.

In the controller architecture, there are also two levels of software. The lower of these two is also a "port driver" and, like the port driver in the host, is concerned only with passing messages on and off of the bus. The higher level of controller software is the "MSCP Server". It constitutes the intelligence of the controller and therefore defines the functionality of the controller.

The MSCP server concerns itself with determining the number of devices, their type, geometry, unit number, availability, status, etc. The MSCP server receives requests from the host and sends responses to the host. It optimizes the requests, performs the operations, transfers the data to/from the host, transfers the data to/from the

device, and buffers the data as necessary. The MSCP server performs error detection and recovery, and reports any significant errors to the host.

Because the MSCP server handles the error detection and recovery by itself, the host sees a "perfect media", an important characteristic of an MSCP subsystem. That is, the host need only report errors to higher level (user) software, as the MSCP server performs all error recovery and media defect (bad block) handling.

The host's class driver and the controller's MSCP server route their messages through the path of the two port drivers and a hardware interconnect. This is their physical connection. However, their logical communication is a direct connection because the port driver details are below their level of concern. Therefore, there are two paths to consider, a physical message path and a logical MSCP connection. This is illustrated in Figure 1.

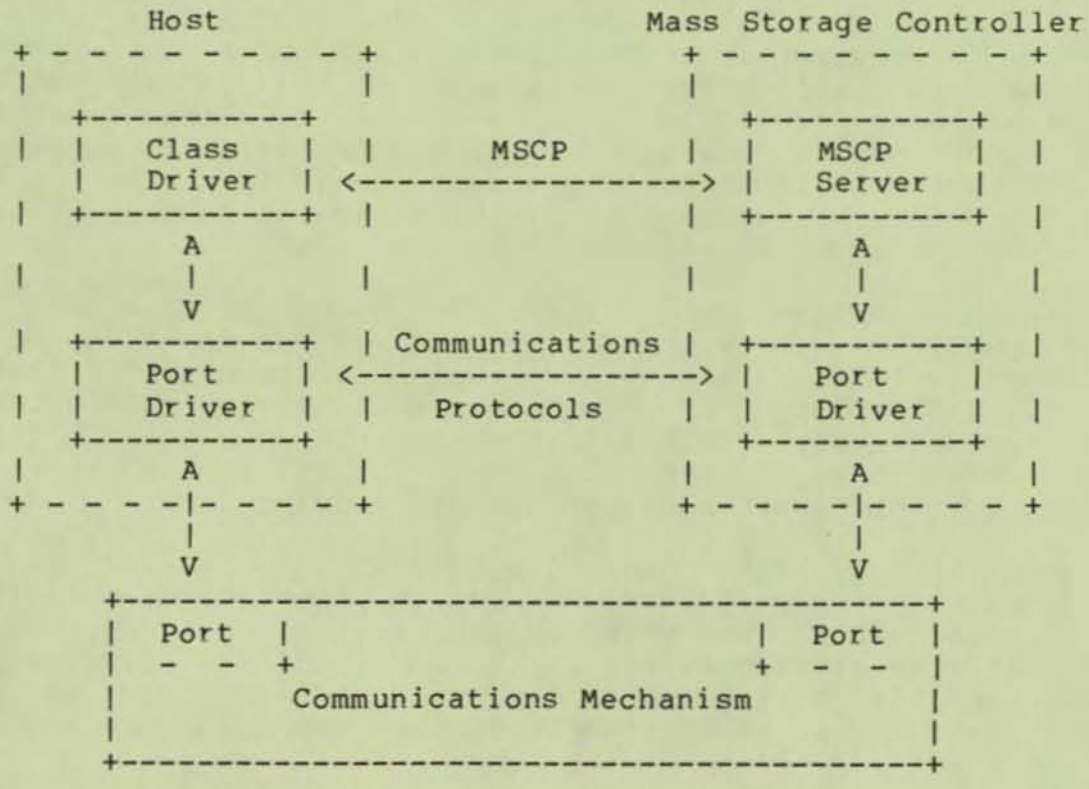


Figure 1 Example System

In summary, an MSCP subsystem is characterized by an intelligent controller that provides the host with the view of perfect media of a fixed size for a given device. It is further characterized by host independence from a specific bus, controller, or device type.

1.2 Purpose

The purpose of this manual is to provide information on the protocol of the port to the detail necessary for writing a port driver.

1.3 Scope

The scope of this manual is limited to the details of the port itself. It does not assume any specific type of host processor or operating system. It does not assume any particular controller or device type.

CHAPTER 2

OVERVIEW OF THE PORT ARCHITECTURE

2.1 OVERVIEW OF THE PORT ARCHITECTURE

The port is a member of a family of related ports and communications mechanisms. This family shares a single System Communications Architecture (SCA) which is described briefly below:

In SCA, communication takes place between pairs of processes resident in separate subsystems. Subsystems include host computers and device controllers. The corresponding processes are host-resident class drivers and controller-resident protocol servers.

The actual transfer of messages uses a set of services called Systems Communications Services (SCS). The Unibus port, in conjunction with a host-resident port driver, implements an instance of SCS.

Communication between a pair of processes takes place over a "connection" which is a soft communication path through the port. A single port typically will implement several connections concurrently. SCS provides primitives for establishing and terminating connections and for determining their states. Once a connection has been established, the following three SCS services are available across that connection.

1. Sequential Message
2. Datagram
3. Block Data Transfer

When a connection is terminated, all outstanding communications on that connection must be discarded; that is, the receiver must "throw away" all unacknowledged messages and the sender must "forget" that such messages have been sent.

The Unibus implementation of SCS/SCA has the following characteristics.

1. Communication is always point-to-point in that exactly two subsystems are connected; one of these is always a host.

2. The port need not be aware of mapping or memory management, since buffers are identified by a Unibus address and are contiguous within the bus address space.

3. The host need never directly initiate a block data transfer.

Because of the point-to-point property, and because the device controller knows the class of device (such as disk) which it controls, the port effectively is integral with the controller and all necessary connections can be established by the port/controller when it is initialized. The host must provide for the necessary connections at system generation or system configuration time. The connections are terminated either by the port entering the fatal error state, or by reinitializing the port. Note that with the Unibus port, all connections are established or terminated as a group.

The Sequential Message service guarantees sequential and duplicate-free delivery for all messages sent over a given connection. Messages are received by the receiving process in the exact order in which the sending process queued them for sending, and no duplicate of a given message will be received. If these guarantees cannot be met, or if a message cannot be delivered for any reason, the port enters the fatal error state and all port connections are terminated.

The Datagram service does not guarantee reception, sequential reception or duplicate-free reception of datagrams, though the probability of failure is required to be "very low". The Unibus port itself can never be the cause of such failure so that, if the using processes do make such guarantees for datagrams, then the Datagram Service over the Unibus port would be equivalent to the Sequential Message service.

The Block Data Transfer service is used to move data between named buffers in host memory and the device controller. In order to allow the port to be unaware of mapping or memory management, the "name" of a buffer is merely the bus address of the first byte of the buffer. Since the host never directly initiates a Block Data Transfer, there is no need for the host to be aware of controller buffering.

A problem common to all communicating asynchronous processes is the need for flow control. This need arises from the possibility of a sending process producing congestion or deadlock in a receiving process by sending messages more quickly than the receiver can cope with them. Flow control simply guarantees that the receiving process has buffers in which to place incoming messages. The sending process is inhibited when all such buffers are full.

The Datagram service does not use flow control. This means that if the receiving process does not have an available buffer in which to store the datagram, then the datagram must be either processed immediately or discarded. Discarding is explicitly permitted by the

rules of Datagram service.

The Sequential Message service does use flow control. Each potential receiving process must reserve or pre-allocate some number of buffers into which messages may be received over its connection. This number is therefore the maximum number of messages which the sender may have outstanding with the receiver, and it is communicated to the sender by the receiver in the form of a "credit" for the connection. The message credits machinery for the Unibus port is described in detail in Section "Message Credits".

2.2 THE UNIBUS PORT ARCHITECTURE

The Unibus Port is the software/hardware interface between port drivers and controllers. It provides the following general capabilities.

1. Provides, at initialization time, information for verifying correct operation of the subsystem controller.

2. Allows for parallel operation of multiple devices attached to the controller, with full duplexing of operation initiation and completion signals, and of data transfers.

3. Mimimizes host interrupts during peak I/O loads.

The overall port consists of two layers.

1. The Transport Layer: This is the machinery for the bi-directional transmission of words and control signals; in the case of the Unibus Port, it is the I/O bus along with any needed bus adapter logic in both the host and the controller.

2. The Logical Layer: This is a set of rules and procedures implemented partly in the host and partly in the controller. The tasks of the logical layer are the exchange of control messages and the verification of correct operation of the transport mechanism and of the overall port.

   The host is assumed to provide a port driver which is the interface between a class driver (e.g. for disk or tape) and the transport mechanism. Thus the port driver implements the host's side of the port proper.

The port's logical layer is implemented as a set of data structures in host memory which are operated on by both the host and the subsystem controller by a set of rules to be discussed later.

The port architecture does use interrupts and Unibus I/O page registers for certain aspects of port operation. These are concerned with the operation of the port itself and not directly with that of the I/O devices attached to the controller.

The port design assumes a command/response relartionship. Command/response transmission uses an asynchronous, packet-oriented protocol. The actual transmission of commands and responses is effected by the port via DMA transfers from/to a communication region in host memory. The port polls this region for commands; the host polls it for responses. From the viewpoint of the host, an I/O operation begins when the host deposits a command descriptor in the command ring. The operation is seen as complete when the corresponding response packet is removed by the host from the response ring.

Interrupts are generated by the port when the command ring makes the transition full to not-full or when the response ring makes the transition empty to not-empty.

CHAPTER 3

TRANSPORT LAYER

3.1 TRANSPORT LAYER

The transport layer is the Unibus and any associated host based or controller based logic for adapting to the bus. This machinery must have the following characteristics.

1. Conform to the Unibus specification (PDP11 BUS HANDBOOK)

2. Allow repeated access (whether reads, writes or any mixture) to the same host memory location.

3.2 Transport Layer Requirements

The host-resident port driver and the controller must provide transport layer control facilities for dealing with the following situations.

1. Transmission of commands and responses.

2. Sequential delivery of commands: Commands must be fetched by the controller in the order in which they were queued to the transport mechanism.

3. Asynchronous communication: It is possible that responses will be sent in an order different from that of the triggering commands.

4. Unsolicited responses: The controller may send unsolicited messages at any time, assuming they have been enabled. The enabling mechanism and the message formats/semantics depend on the higher-level command/response protocol.

5. Full duplex communication: A command, response or unsolicited response may occur at any time.

6. Port failure recovery: The port driver or class driver must be able to place a timer on the controller, and must be able to reinitialize the port in the event of controller timeout.

The port/controller will enter the fatal error state if a command or response is lost and the port/controller cannot notify a higher level of host software.

3.3 UNIBUS I/O PAGE REGISTERS

Two 16-bit registers in the Unibus I/O page are used for control of the port. These registers are always read as words. The register pair begins on a longword boundary. The register names, addresses and functions are:

IP	7xxxx0/4	initialization and polling
SA	7xxxx2/6	status, address and purge

The IP register has two functions as detailed below.

1. When written with any value, it causes a hard initialization of the port and the device controller.

2. When read while the port is operating, it causes the controller to initiate polling as discussed in Section "Polling".

The SA register has four functions as listed below.

1. When read by the host during initialization, it communicates data and error information relating to the initialization process.

2. When written by the host during initialization, it communicates certain host-specific parameters to the port.

3. When read by the host during normal operation, it communicates status information including port- and controller-detected fatal errors.

4. When zeroed by the host during both initialization and normal operation, it signals the port that the host has successfully completed a bus adapter purge in response to a port-initiated purge request.

The detailed operation of these registers is discussed in Section "DETAILS OF INITIALIZATION". Note that only word transfers to/from IP and SA are permissible; the behavior of byte transfers is undefined.

CHAPTER 4

LOGICAL LAYER

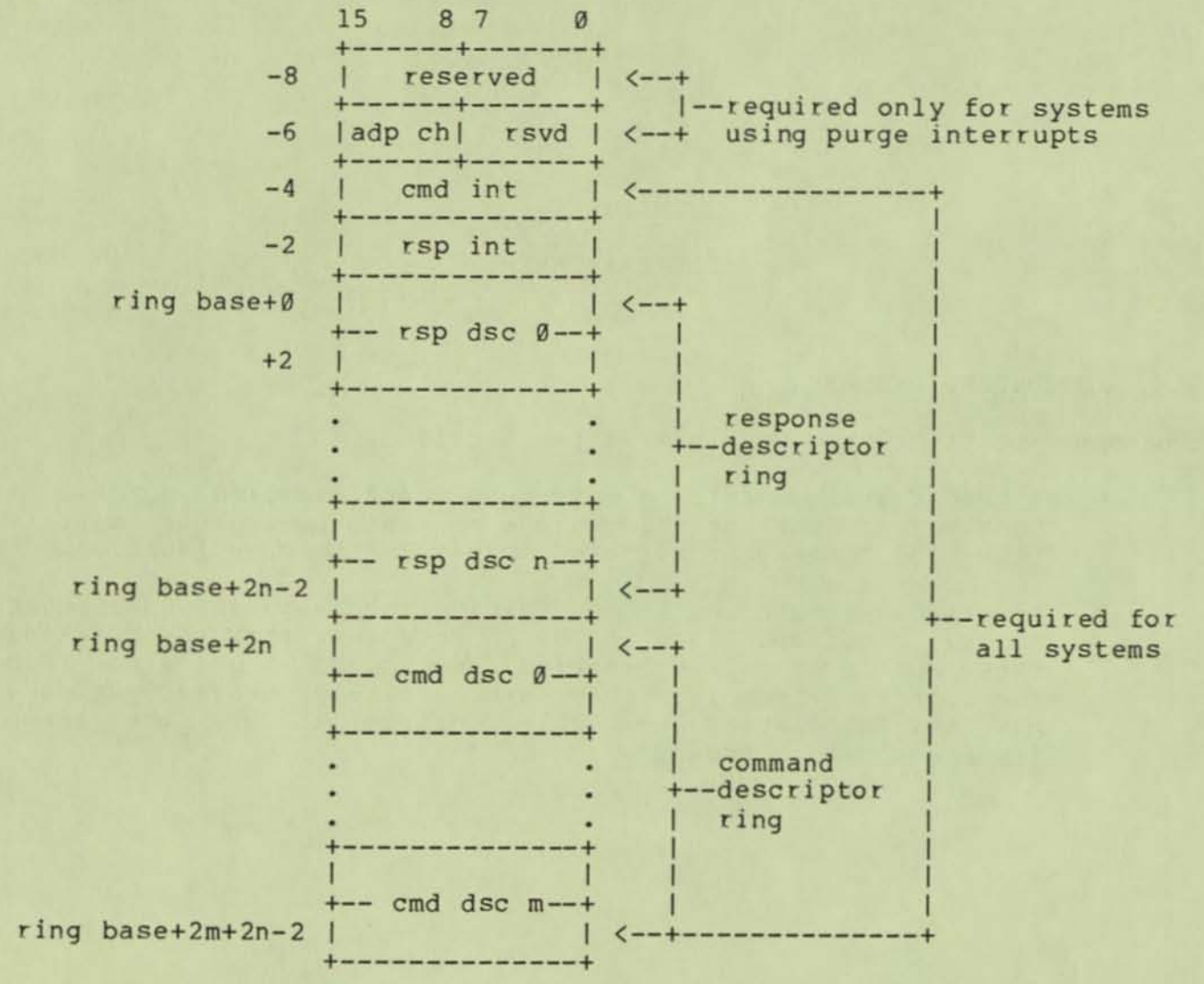
4.1 COMMUNICATIONS AREA

The communications area consists of two sections.

1. A header area containing interrupt identification words. A portion of this area is unique to hosts having bus adapters requiring purges for repeated access to the same location.

2. A variable-length section containing the response and command rings, organized into rings. Note that strictly speaking these should be called "receive" and "send" rings, since from the port's viewpoint they are entirely general-purpose. However, for clarity we will continue to use the terms "response" and "command".

The communications area is aligned on a 16-bit word boundary. Its layout is:



note: n = response ring size
m = command ring size

4.1.1 Interrupt Indicators and SA Register

Words [ringbase -6,-4,-2] are used as indicators which are zeroed by the host but which, when the port interrupts the host, will have been set non-zero by the port to indicate the reason for the interrupt. The word meanings are:

ring base-6: Port is requesting a bus adapter purge. The non-zero value is the adapter channel number and is contained in the high-order byte. It is derived from the triggering command.

The host responds by performing the purge. It then signals purge completion by writing zeroes to the SA register.

-4: Command ring transitioned full to not-full.
SET NON-zero by port. Cleared by host.

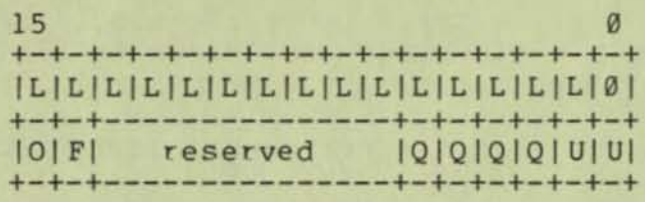
-2: Response ring transitioned empty to not-empty.
Set non-zero by port. Cleared by host

However entered, the port driver should examine the SA register regularly to verify normal port/controller operation. A port/controller self-detected fatal error is reported in the SA register as discussed in Section "SELF DETECTED FATAL PORT/CONTROLLER ERRORS".

4.1.2 Command and Response Rings

The command and response rings each are organized into a ring of 32-bit descriptors. The length of each ring is determined by the relative speeds with which the host and the port/controller generate and process messages and is unrelated to the controller command limit discussed in Section "Message Credits". The host sets the ring lengths at initialization time as discussed in Section "DETAILS OF INITIALIZATION".

A formatted descriptor has this layout:



where:

Bit Ø is zero. This is because the envelope address [text+Ø] is word-aligned. Note that the port/controller will always assume that bit Ø is set to zero.

L is a bit in the low-order envelope address, for all systems.

U is a bit in the high-order portion of an 18-bit Unibus address.

Q is reserved

O is an "ownership" bit indicating whether the descriptor is owned by the host (O=Ø) or by the port (O=1) and which interlocks the descriptor against premature access by either party.

F is a "flag" bit whose meaning varies depending on the state of the descriptor. These are described below.

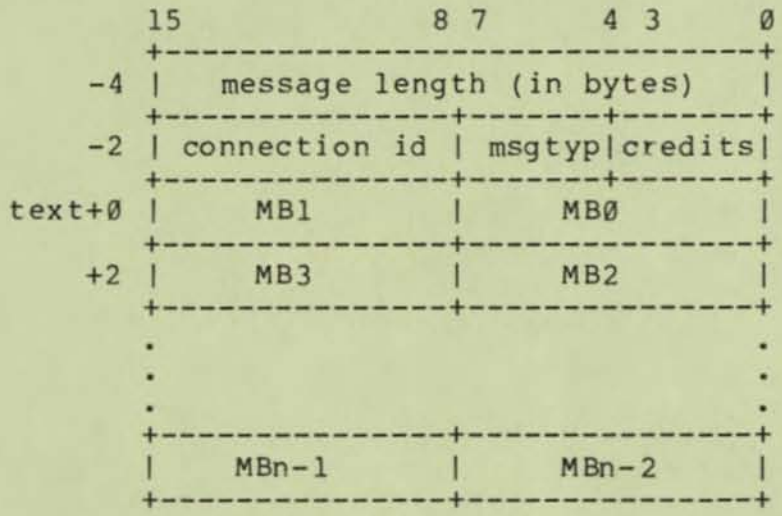
When the port returns ownership of a descriptor to the host, it sets F=1 to indicate that the port has completed action on the descriptor.

When the port acquires ownership of a descriptor from the host, F=1 indicates that the host is requesting a ring transition interrupt. If F=Ø, the host is not requesting a ring transition interrupt. The ring transition interrupt will occur only if this descriptor causes a ring transition and transition interrupts were enabled during initialization.

Since the port always will set F=1 when returning a descriptor to the host, a host desiring to override ring transition interrupts must always clear F when passing ownership of a descriptor to the port.

4.1.3 Message Envelopes

The command or response descriptor points to word [text+Ø] of a 16-bit word-aligned message envelope formatted as follows:



The fields have the meaning shown below.

msg length gives the length of the message text, in bytes.

For commands, this length is equal to the size of the command, in bytes, beginning with [text+Ø].

For responses, the host sets the length equal to the size of the response buffer, in bytes, beginning with <text+Ø>. The minimum acceptable size is 6Ø bytes of message text (64 bytes overall). Before actual transmission of a response, the controller reads the length field in the message envelope. If the port's response is longer than the response buffer, the port will fragment its response into as many response buffers as necessary.

The port sets the resulting value into the message length field. The host therefore must keep re-initializing the value of this field for each proposed response.

Note that if a controller's responses are less than or equal to 6Ø bytes, then the controller need not check the size of the response slot.

MBj is a byte of message text.

connection id identifies the connection serving as source of, or destination for, the message in question.

credits gives the credit value associated with the

message, as discussed in Section "Message Credits".

msgtyp indicates the message type, as follows:

Ø - Sequential Message: The "credits" and "message length" fields are valid.

1 - Datagram: "Credits" must be zero; "message length" is valid.

2 - Credit notification: "Credits" is valid; "message length" must be zero.

3-14 - Reserved (unassigned).

15 - Maintenance.

Note that the port provides no information protection for the text of the message beyond bus parity, which is not implemented by all systems.

4.1.4 Message Credits

The higher-level protocol is required to use a credit-based command limit mechanism. The credits field of the envelope supports the algorithm explained below.

1. In its first response the controller will return, in the credits field of that response, a number N. This number is one more than the controller's limit for non-immediate commands, the "extra" being to allow the host always to be able to issue an immediate-class command. The class driver is to remember the delivered number in its "credit account".

2. Each time the class driver queues a command, it decrements the credit account by one.

3. Each time the class driver receives a response, it increments the credit account by the value contained in the credits field of that response. For unsolicited responses this value will be zero; for solicited responses it normally will be one (an exception is discussed below).

4. If the credit account has a value of one, the class driver may issue only an immediate-type command. If the account balance is zero then the class driver may not issue any commands at all.

Note that for a controller having N > 15(10), responses beyond the first will have [credits] > 1, allowing the controller to "walk" the class driver's credit balance up to the correct value.

Note also that for a well-behaved class driver, enlarging the command ring beyond the value N+1 provides no performance benefits, and that in this situation command ring transition interrupts will not occur since the class driver will never fill the command ring.

4.2 MESSAGE TRANSMISSION

4.2.1 Command Transmission

For a command descriptor, the ownership bit transition O Ø-->1 means that the host has filled the descriptor and is releasing it to the port. The transition O 1-->Ø means that the port has emptied the command descriptor and is returning the empty descriptor to the host. Thus to send a command, the host sets O=1; the port clears O when the command has been received, returning the empty slot to the host.

To guarentee that the port/controller sees each command, the host must read the IP register whenever it inserts a command in the command ring. This forces the port to poll the command if it was not already accessing the command ring.

4.2.2 Response Transmission

For a response descriptor, the transition O 1-->Ø means that the port has filled the descriptor and is releasing it to the host. The transition O Ø-->1 means that the host has emptied the response descriptor and is returning the empty descriptor to the port. Thus to send a response the port clears O; the host sets O=1 when the response has been received, returning the empty slot to the port.

Just as the port must poll for commands, so must the host poll for responses. This is especially true because of the possibility of unsolicited responses. See Section "Host Polling".

4.2.3 Interrupts

The transmission of a message will result in a host interrupt if and only if interrupts were armed suitably during initialization (see Section "OVERVIEW OF INITIALIZATION") and one of the following conditions has been met.

1. The message was a command with F=1 and the port's fetching it caused the command ring to transition from full to not-full. This interrupt means that the host may place another command in the command ring.

2. The message was a response with F=1 and the port's depositing it caused the response ring to transition from empty to not-empty. This interrupt means that there is a response for the host to process.

3. The port is interfaced to the host via a bus adapter and a command required the port/controller to re-access a given location during data transfer. This interrupt means that the port/controller is requesting the host to purge the indicated channel of the bus adapter.

4.2.4 Polling

4.2.4.1 Port Polling -

An initially idle port/controller will not poll for commands or response slots until stimulated by the host's reading of the IP register. When that happens, the port/controller begins reading commands out of host memory (the controller may have an internal command buffering capability). The port will continue to poll for full command slots until the command ring has been found to be empty. Thus it is extremely important that the class driver adhere to the message credits policy discussed in Section "Message Credits".

Having ceased polling, the port will resume polling either when it delivers a response to the host, or if no responses will be delivered, when the host reads the IP register.

Response polling (for empty slots) continues until any commands buffered within the controller have been completed and the responses sent to the host. At that time the controller will cease accessing the communications area.

Such I/O rundown is a necessary (but insufficient) condition for the host to be able to reallocate the memory occupied by a transient driver. The additional conditions to be met are listed below.

1. Unsolicted responses must be disabled (higher-level protocol-dependent).

2. At least 100 milliseconds must have elapsed since the host last read the IP register.

The port/controller will cease accessing the communications area when it is initialized.

4.2.4.2 Host Polling -

Because of the possibility of unsolicited responses, it is not generally sufficient for the host to cease polling for responses when all outstanding commands have been acknowledged. An accumulation of such unsolicited messages would first saturate the response ring and then any controller internal message buffers, blocking the controller and preventing it from processing additional commands.

Thus the host must at least occasionally scan the response ring, even though it may not be expecting a response. The best way to mechanize this requirement is through the use of the ring transition interrupt facility described earlier, supplemented with the advice that the host should pull from the response ring as many responses as it finds, ceasing polling only when the response ring is found to be empty. This policy not only guarantees that the host will see every response in a timely way, it also eliminates unnecessary host polling.

CHAPTER 5

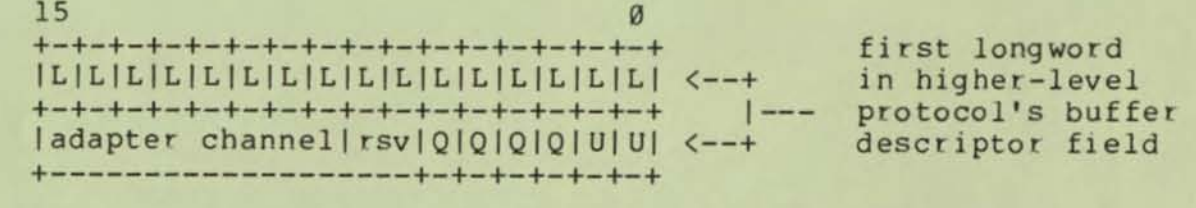
DATA TRANSMISSION

5.1 DATA TRANSMISSION

The details of data transmission are port/controller dependent, however the general philosophy will be outlined here.

In the command ring, the command descriptor points to a command packet. Within the command packet there is a buffer descriptor which contains a pointer and a byte or word count. The buffer descriptor points to the data buffer. The data buffer serves as a source or sink for data transfers. The data is moved by the port in to/out of the buffer as DMA transfers to/from Unibus addresses (host memory being the most likely source/destination).

A buffer descriptor begins at the first word allocated for this purpose in the formats of higher-level commands. The Unibus port uses a two-word buffer descriptor formatted as follows:



in which:

L is a bit in the low-order buffer address, for all systems. Note that the higher-level protocol may require buffers to begin on word boundaries; the port makes no such restriction.

U is a bit in the high-order portion of an 18-bit Unibus address.

Q is reserved

To support such higher-level protocol functions as transfer restarts, compares, etc., the host memory interface must allow repeated access to a given host memory location for both reads and writes. On purely

Unibus systems such as 11/44, this requirement is trivially met with no participation by the host CPU.

On systems with bus adapters such as the 11/780, the repeated access requirement means that the relevant adapter channel may have to be purged, requiring the active cooperation of the host CPU. The port signals its desire for an adapter channel purge by interrupting the host. The host writes zeroes to the SA register to indicate purge completion.

The port architecture allows the host to require the port/controller to break a lengthy DMA transfer into a series of bursts, each limited to a host-specified number of word. See Section "Step 4".

CHAPTER 6

TRANSMISSION ERRORS

6.1 TRANSMISSION ERRORS

Four classes of errors are considered in the Unibus Port Architecture. They are listed below.

1. Failure to become Bus Master: This can arise whenever the port attempts to access host memory for any reason.

   To deal cleanly with this condition before requesting the bus, the port always sets up a corresponding "last fail" response packet (see Section "Purge and Poll Test") before actually requesting the bus. The port then requests the bus and waits "forever" for bus grant. Should the grant fail to come, the host eventually will experience a timeout and will reinitialize the port/controller. At that time the port will report the Bus Master error via the previously set up "last fail" response packet, assuming such packets were enabled during the reinitialization.

2. Failure to become Interrupt Master: This can arise whenever the port attempts to interrupt the host for any reason.

   The treatment and reporting of this error are analogous to those of failure to become Bus Master.

3. Bus Data Timeout error: Whether retries are performed is controller-dependent. If they are, a persistent error results in differing action depending on whether the offending operation was a control or data transfer.

   1. If a control transfer, the port is to set a failure code into the SA register and then to terminate the connection with the port. The port/controller will have to be reinitialized.

   2. If a data transfer, the port/controller remains online to the host. The failure is to be reported in the response to the offending operation.

4. Bus Parity error: The action here is the same as for bus data timeout errors.

CHAPTER 7

SELF-DETECTED FATAL PORT/CONTROLLER ERRORS

7.1 SELF-DETECTED FATAL PORT/CONTROLLER ERRORS

Various fatal errors may be self-detected by the port or controller. Some of these may also arise while the controller is operating its attached device(s).

In the event of a fatal error, the port posts the followin bits in SA.

1. Bit 15 = 1 Fatal Error Indicator.

2. Bits 10-0: Fatal Error Code.

Port-generic fatal error codes are binary numbers occupying the value range 1-199 (10). Currently-assigned codes are:

001 - Envelope/Packet Read (parity or timeout).
002 - Envelope/Packet Write (parity or timeout).
003 - Controller ROM and RAM parity.
004 - Controller RAM parity.
005 - Controller ROM parity.
006 - Ring Read (parity or timeout).
007 - Ring Write (parity or timeout).
008 - Interrupt Master.
009 - Host Access Timeout (higher-level protocol-dependent).
010 - Credit Limit Exceeded.
011 - Unibus Master Error
012 - Diagnostic Controller Fatal Error.
013 - Instruction Loop Timeout.
014 - Invalid Connection Identifier.
015 - Interrupt Write.
016 - MAINTENANCE READ/WRITE Invalid Region Identifier.
017 - MAINTENANCE WRITE Load to non-loadable controller.
018 - Controller RAM error (non-parity)
019 - INIT sequence error
020 - High-level protocol incompatibility error
021 - Purge/poll hardware failure
022 - Unassigned
 * - "
099 - Unassigned

Codes 100(10) and higher are assigned to specific controllers in groups of 100 so that a given controller uses the code range N00-N99 (10). Appendix B contains a table of currently-assigned code ranges.

CHAPTER 8

PORT PERFORMANCE CONSIDERATIONS

8.1 PORT PERFORMANCE CONSIDERATIONS

As earlier mentioned, ring interrupts are generated by the port when the command ring makes the transition full to not-full or when the response ring makes the transition empty to not-empty.

Since a full condition in either ring implies that the intelligence responsible for filling that ring is blocked in some way, ring sizes should be specified large enough to keep the incidence of full rings small. For the command ring, the optimal size depends on the latency in the controller's polling the command ring. For the response ring, the optimal size depends on the latency in invoking the host software process which will empty the ring. The message credit scheme generally ensures that resource allocation will not increase these latencies.

CHAPTER 9

INITIALIZATION

9.1 OVERVIEW OF INITIALIZATION

The initialization procedure serves the following purposes.

1. Identify the parameters of the host-resident communications region to the port.

2. Provide a confidence check of port/controller integrity.

3. Bring the port/controller online to the host. Note that this action does not bring the devices online to the controller.

The initialization procedure consists of a hard initialization during which the port/controller runs preliminary diagnostics and which either fails, requiring reinitialization, or succeeds and is followed by the following four-step procedure.

1. The host writes into the SA register the lengths of the rings, whether interrupts are to be armed and, if so, the address of the interrupt vector. The port/controller then runs a complete internal integrity check and signals either success or failure.

2. The host reads from SA an echo of the ring lengths, and then writes into SA the low-order portion of the ring base address and whether the host is one which requires purge interrupts.

3. The interrupt vector address and the master interrupt arming signal are echoed in the SA register. The host then writes the high order portion of the ring base address to the SA register along with a signal that conditionally triggers an immediate test of the polling and adapter purge functions of the port.

4. The port tests the ability of the I/O bus to perform DMA transfers. If successful, the port then zeroes the entire communications area. The port then signals the host that initialization is complete. At this point, the port driver sets the response slot ownership bits so that the port owns

all the slots. The controller then waits for a signal from the host to begin normal operation.

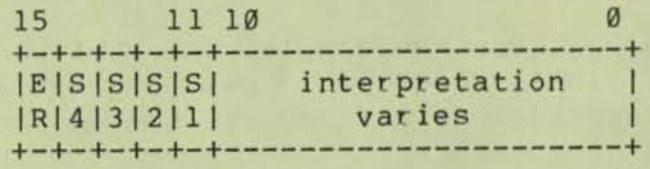
At each step the port informs the host of either failure (requiring a restart of the initialization sequence) or success (and therefore willingness to progress to the next initialization step). The host maintains timeouts on the various port/controller responses.

At various points information is echoed by the port to the host. The intent here is to echo all bit positions but not necessarily to echo all fields.

9.2 DETAILS OF INITIALIZATION

During initialization, the detailed format and meaning of the SA register depends on the initialization step and whether SA is being read or written.

When being read, certain aspects of the SA format are constant and apply to all steps. This constant SA read format is:



The bits S1-S4 are set separately by the port to indicate which step it is ready to perform. If the host detects more than one S-bit set at any time, it should reinitialize the port/controller. If this happens a second time, the host should consider the port/controller to be "dead".

If ER=1, then either a port/controller-based diagnostic test has failed or there has been a fatal error. Bits 10-0 display an error code which may be either port-generic or controller-dependent. See Section "SELF DETECTED FATAL PORT/CONTROLLER ERROR.

Thus in the event of an error, the host can determine not only the nature of the error but also the step during which the error occurred. If ER=1 and a step bit is set, then a fatal error was detected during hard initialization.

In the event of an initialization error, the port driver must retry the sequence at least once. It is suggested, however, that a second failure be considered as meaning that the port/controller is "down".

Note also that during both initialization and normal operation an error is always posted in SA as described above. In addition it may be recorded within the controller and reported at a higher level when initialization completes (see the LF bit discussion in Section "Step 4").

The host begins the initialization sequence either by issuing a bus INIT or by writing any value to the IP register. The port must guarantee that the host will read zeroes in SA on the next bus cycle. Initialization then sequences through Steps 1-4 as described on the following pages.

At the beginning of Step n, the port is to clear bit Sn-1 before setting bit Sn so that the host will never see bits Sn-1,Sn set simultaneously.

From the host's viewpoint, Step n is deemed to have begun when reading SA shows the transition Sn 0-->1. Of course, Step n ends when Step n+1 begins as just defined. This transition from Step n to Step n+1 may be accompanied by an interrupt, depending on whether interrupts are enabled.

Steps 1-3 each are required to complete within 10 seconds. If any of these steps fails to complete within that period, this is to be treated as a host-detected fatal error.

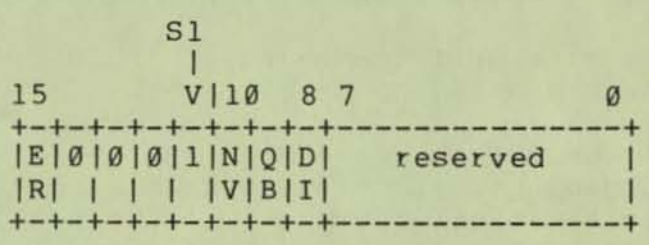
There is no explicit signal for the completion of Step 4. Rather, the host observes either that controller operation has begun or that a higher-level protocol-dependent timer has expired.

During initialization, the host must wait 100 microseconds after any interrupt before reading the SA register to see if there was an error. This is because the port may use the SA register to deliver the vector address to the processor interrupt sequence. If it does, then time will be required by the port to set SA to the value to be read by the host initialization code.

Assuming that ER continues to be zero, the remainder of initialization is as described in the following sections.

9.2.1 Step 1

The host knows that Step 1 has begun when S1 makes the transition Ø-->1. At that time the following pattern may be read from SA:



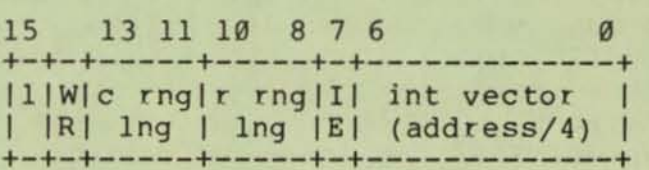
This pattern should appear within 1ØØ microseconds after the hard-initialize. Bits 1Ø-8 are controller-dependent. Their interpretations are shown below.

NV=1 means that the port does not support a host-settable interrupt vector address.

QB reserved.

DI=1 means that the port implements enhanced diagnostics, i.e. wraparound, purge and poll tests.

The host responds by writing the following pattern into SA:



Note that bit 15=1. This is to guarantee that the port does not interpret the pattern as a host "adapter purge complete" response (after a spontaneous reinitialize).

Note also that the vector address pertains to all interrupts generated by the port.

The remainder of the host-generated Step 1 SA pattern has this interpretation:

WR=1 means that the port should enter diagnostic wrap mod. See Section "PORT DIAGNOSTIC FACILITIES".

The port will ignore WR if it delivered DI=Ø at the beginning of Step 1.

c rng lng is the number of slots (32 bits each) in the command ring, expressed as a power of two. Thus for a maximal command ring the host would set this field equal to seven, requesting 2**7=128 slots.

r rng lng is the number of response ring slots, again expressed as a power of two.

IE=1 means that the host is requesting an interrupt at the completion of each of Steps 1-3.

Note that no interrupt will be generated at the completion of Step 4 since this step requires only a small number of microseconds.

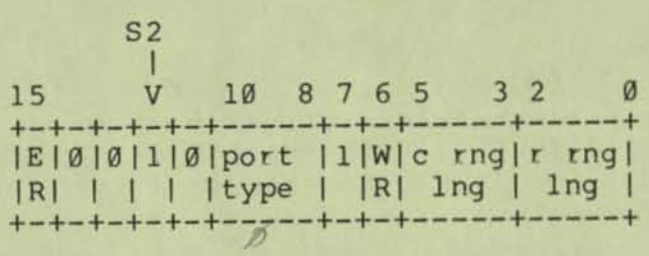
int vector determines if interrupts will be generated by the port. If this field is non-zero, interrupts will be generated during normal operation and, if IE=1, during initialization. If this field is zero, then interrupts will not be generated during normal operation and initialization (regardless of the "IE" bit state). When this field is non-zero, those ports which accept an interrupt vector from the host use the value in the field as the address/4 of the interrupt vector.

Upon receipt of the above data the port/controller begins running its integrity check diagnostics. When finished, the port conditionally interrupts the host as described above. If enabled, the interrupt will take place whether the diagnositics succeeded or failed.

Step 1 must complete within 1Ø seconds after the host writes to the SA register. The completion will result in an interrupt if IE was set to one in Step 1.

9.2.2 Step 2

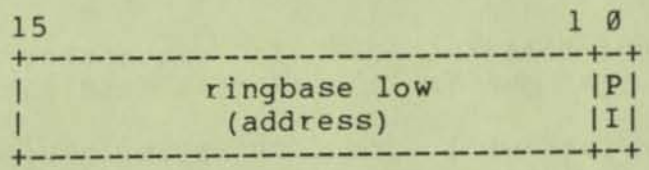
The host knows that Step 2 has begun when the transition S2 Ø-->1 takes place. At that time the following pattern may be read from SA:



Bits 10-8 are a port type number. All controllers conforming to this port specification will use the value assigned to "Unibus Storage Systems Port". (See Appendix C for a table of assigned port type numbers. The controller type is reported via the higher-level protocol.)

Bits 7-Ø are echoes of the data written by the host to SA bits 15-8 during Step 1.

The host responds by writing into SA the following pattern:



The above data has the following interpretation:

ringbase low is the low-order portion of the address of word [ringbase+Ø] of the communications area. This is a 16-bit byte address whose low-order bit is zero implicitly.

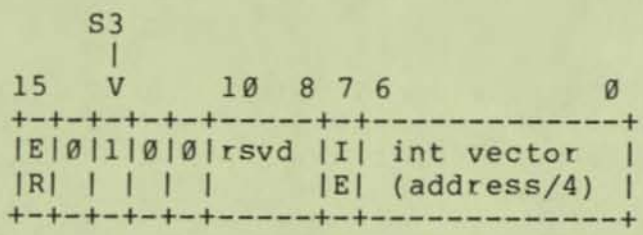
Note that the high-order portion of this address is written to SA by the host during Step 3.

PI=1 means that the host is requesting adapter purge interrupts.

Step 2 must complete within 10 seconds from the time the host writes the Step 2 data to SA. The completion will result in a host interrupt if IE was set to one in Step 1.

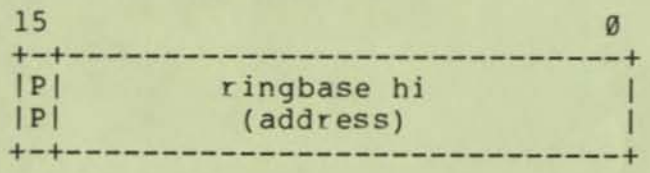
9.2.3 Step 3

The host knows that Step 3 has begun when the transition S3 Ø-->1 takes place. At that time, the following may be read from SA:



In this pattern, bits 7-Ø are the echo of bits 7-Ø of the data which was written to SA by the host during Step 1.

The host responds by writing to SA the following pattern:



This pattern has the following interpretation:

PP=1 means that the host is requesting execution of "purge" and "poll" tests as described below.

The port will ignore PP if it responded with DI=Ø at the beginning of Step 1.

ringbase hi is the high-order portion of the address [ring base+Ø].

Note that the low-order portion of this address will have been written to SA by the host during Step 2.

If PP has been set, then immediately upon writing to SA the host must wait for SA to transition to a zero value. The host must then do two things.

1. Write zeroes to the SA register. (This simulates a "purge completed" host action).

2. Read (and then disregard) the IP register. (This simulates a "start polling" command from the host to the port.)

   The host must complete this sequence within 1ØØ milliseconds from the time SA was first written during Step 3.

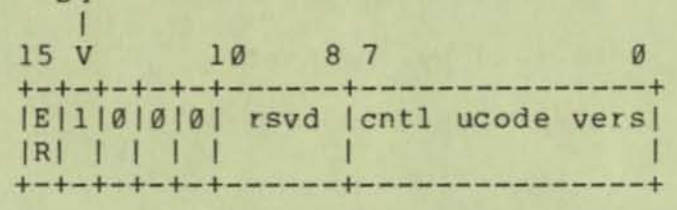
While the host is performing the above steps, the controller, having seen PP=1, does the following functions.

1. Load zeroes into SA.
2. Wait for SA to be written by the host.
3. Wait for IP to be read by the host.
4. Announce in SA the transition to Step 4.

Step 3 must complete within 10 seconds. When it does complete, an interrupt will be generated if IE was set to one in Step 1.

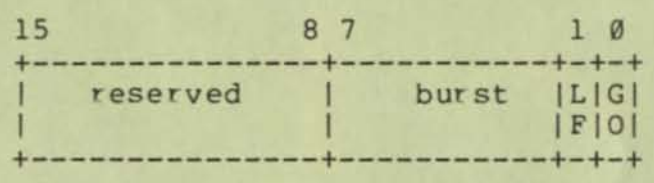
9.2.4 Step 4

The host knows that Step 4 has begun when the transition S4 0-->1 takes place. At that time, the following pattern may be read from SA:



Bits 7-0 give the version number of the port/controller microcode.

The host responds by writing the following pattern into the SA register:



This data is interpreted as follows:

burst is one less than the maximum number of longwords the host is willing to allow per DMA transfer. If this field is zero, then the port is to use its default burst count.

The values of both the default and the maximum the port will accept are controller-dependent.

LF=1 means that the host wants a "last fail" response packet when initialization is complete. The state of LF does not have any effect on the enabling/disabling of unsolicted responses in the higher-level protocol.

GO=1 means that the controller should enter its functional microcode as soon as initialization completes.

Note that if GO=0 when initialization completes, the port will continue to read SA until the host forces SA bit 0 to make the transition 0-->1.

Note that there will be no explicit interrupt at the end of Step 4. Instead, if interrupts were enabled, the next interrupt will be due either to a ring transition or to an adapter purge request.

CHAPTER 10

PORT DIAGNOSTIC FACILITIES

10.1 PORT DIAGNOSTIC FACILITIES

10.1.1 Diagnostic Wrap Mode

Diagnostic Wrap Mode (DWM) provides host-based diagnostics with a means of verifying the lowest levels of host/controller communication via the port.

In DWM, the port attempts to echo in the SA register any data written to SA by the host. DWM is a special path through initialization Step 1; Steps 2-4 are suppressed and the port/controller is left disconnected from the host.

DWM is initiated by the setting of the 'WR' bit in the host's Step 1 response (see Section "Step 1") and takes effect immediately, causing the host's Step 1 response to be echoed in the SA register.

DWM is terminated by the host's re-issuing a hard initialize (bus INIT or host write to IP). Assuming the host found the results of DWM to be satisfactory, the host would then bypass DWM during the second initialization sequence.

It is recommended that a diagnostic program which uses DWN not enable interrupts at Step 1. A port/controller failure could result in interrupts to unintended host addresses.

For each datum to be echoed, the host does the following functions.

1. Write any bit pattern to SA.
2. Wait for SA to match the data written. (This must happpen within 10 seconds or the port/controller has failed.)
3. Loop back to 1 above for another pass.

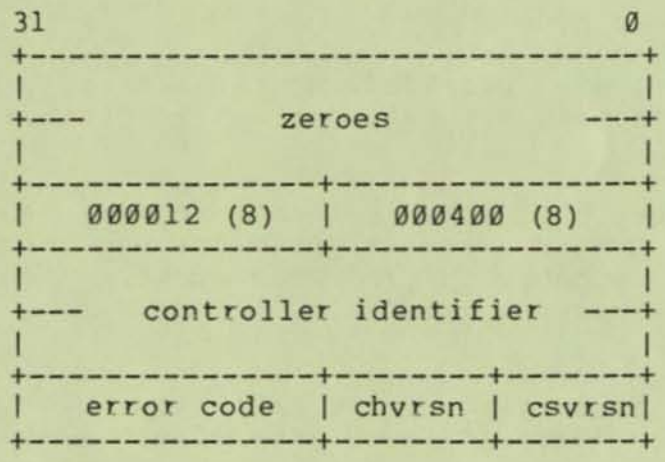
The sequence is terminated by re-initializing the port/controller.

10.1.2 Purge and Poll Tests

These tests are described under Initialization Step 3, Section "Step 3".

10.1.3 Last Fail

The port will send a Last Fail Error Packet to the host if an error was detected during a previous "run" and the LF bit was set in Step 4 of the current initialization. The Last Fail Error Packet has the format shown below.



The error code is discussed in Section "SELF DETECTED FATAL PORT/CONTROLLER ERRORS" and "controller identifier", "chvrsn" and "csvrsn" are discussed in the MSCP Basic Disk Functions Manual under "Error Log Message" formats.

10.1.4 MAINTENANCE READ and MAINTENANCE WRITE

The controller is required to support some variant of the MAINTENANCE READ and MAINTENANCE WRITE commands. Their intended purpose is to allow host-based diagnostics to read and write controller-internal storage.

These commands are not formally part of any higher-level protocol; nevertheless, their formats and many of their field meanings coincide with those of MSCP and the MSCP specification should be consulted for details.

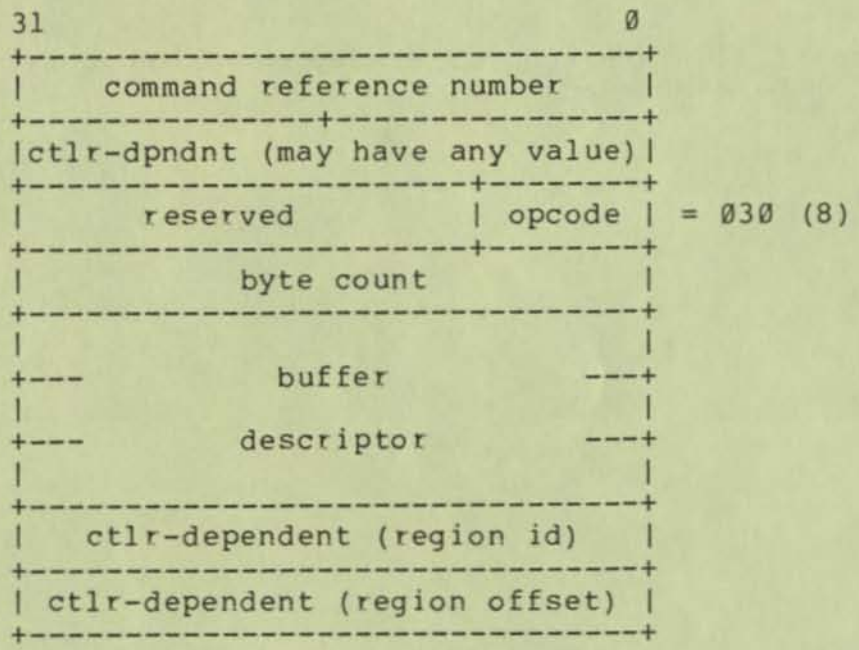
The remaining fields are controller-dependent and only their general outlines will be discussed here.

10.1.4.1 MAINTENANCE READ -

The format of the MAINTENANCE READ command packet is given below. The displayed packet comprises the text of the message contained in the earlier-specified command/response envelope (begining at word [text+0]).

Command Category: Sequential

Command Packet Format:

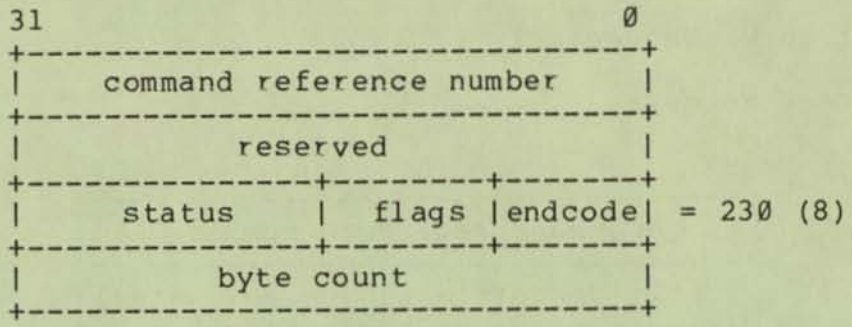


The "region id" serves to select among various classes of storage within the controller. The "region offset" creates sub-classes within a class. At present only one region id is defined.

1. ID = 1: The contents of controller buffer storage, without regard to how the contents got generated.

The controller will respond to the MAINTENANCE READ by delivering the requested data to the requested host buffer (to the limit of the specified byte count) and then delivering to the specified response envelope the following text:

End Packet Format:



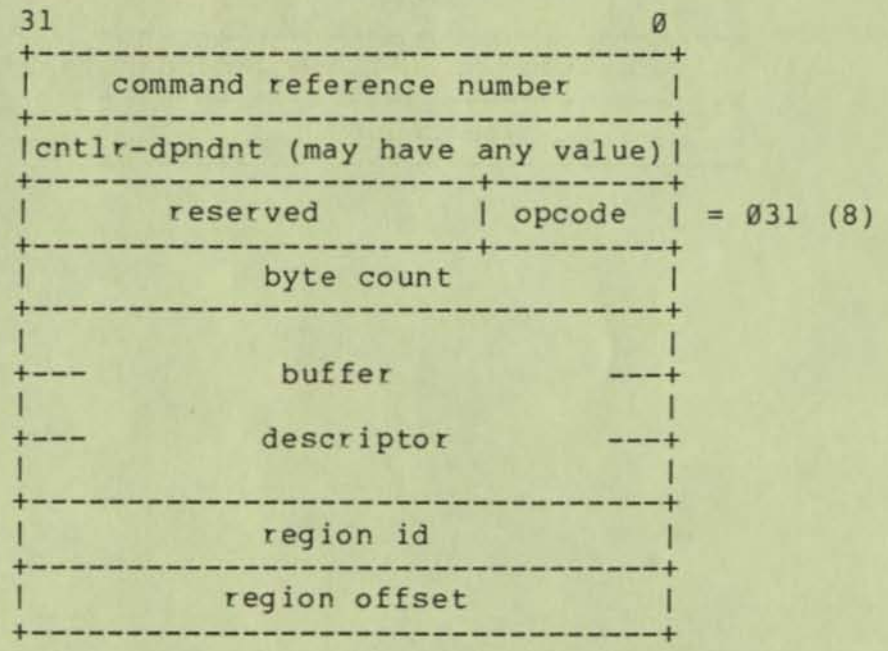
10.1.4.2 MAINTENANCE WRITE -

The MAINTENANCE WRITE command is used by the host to send to the controller information which is outside the scope of the higher-level protocol.

The general MAINTENANCE WRITE command packet format is given below:

Command Category: Sequential

Command Packet Format:

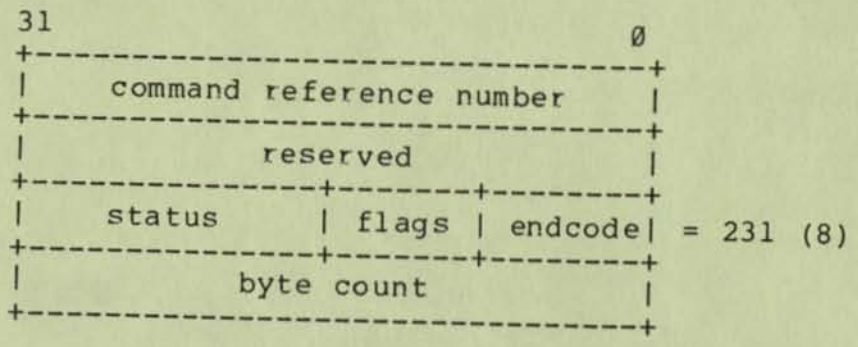


The distinction among data types is made via the region id and region offset fields as shown below.

1. ID = 1: Transfers [byte count] worth of data into the controller at the (virtual) address specified by the region offset.

The controller responds to MAINTENANCE WRITE by fetching the indicated data and then delivering to the indicated response envelope this End Packet text.

End Packet Format:



APPENDIX A

TABLE OF ASSIGNED CONNECTION ID'S AND USES

ID Number	Purpose
Ø (1Ø)	Disk MSCP
1	Reserved
2	DUP
3-254	Unassigned
255	Implementation specific maintenance protocol

APPENDIX B

TABLE OF ASSIGNED ERROR CODE RANGES

Range	Associated Controller
Ø	Reserved
ØØ1-Ø99	Generic, all controllers
1ØØ-199	UDA
2ØØ-299	Reserved
3ØØ-399	Reserved
4ØØ+	Unassigned

APPENDIX C

TABLE OF ASSIGNED PORT TYPE NUMBERS

Number	Corresponding Port Type
Ø	Unibus Storage Systems Port
1-7	Unassigned