EDUCATIONAL
SERVICES

VAX/VMS
Training

VAX/VMS
Device Driver
Appendix:
Supplementary Readings

digital

EY-2278E-ML-0001

APPENDIX: SUPPLEMENTARY READINGS

Prepared by Educational Services
of
Digital Equipment Corporation

First Edition, October 1984

Printed in U.S.A.

The manuscript for this book was created using DIGITAL Standard Runoff. Book production was done by Educational Services Development and Publishing in Nashua, NH.

CONTENTS

FIGURES

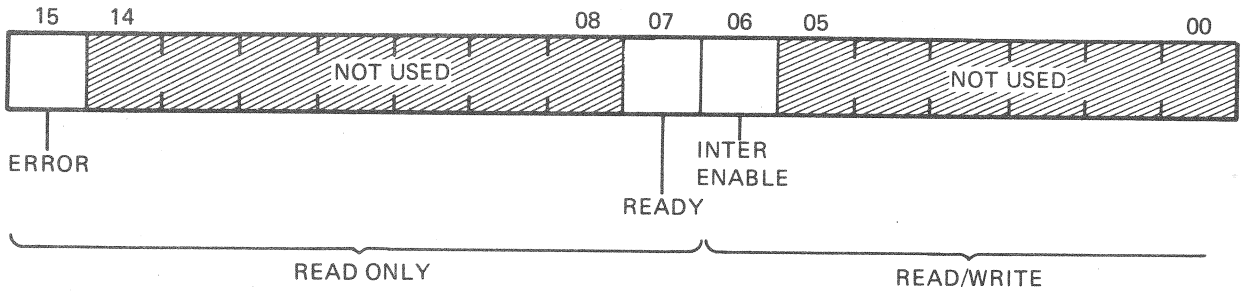
TABLES

APPENDIX
SUPPLEMENTARY READINGS

LP11 Series Registers

The LP11 lineprinter systems use the same controller. Following are the register drawings and bit definitions of the controller's two registers: the Control and Status Register and the Data Buffer Register.

Control and Status Register (LPCS) 777 514



TK-9094

Bit	Name
15	ERROR

Set when an error condition exists in the line printer. Error conditions are power off, no paper, torn paper, line printer drum gate or band gate open, over-temperature alarm, or line printer off-line. Generates an interrupt if INTERRUPT ENABLE <Ø6> is also set. Cleared by correcting the error condition.

14-Ø8 NOT USED

Ø7 READY

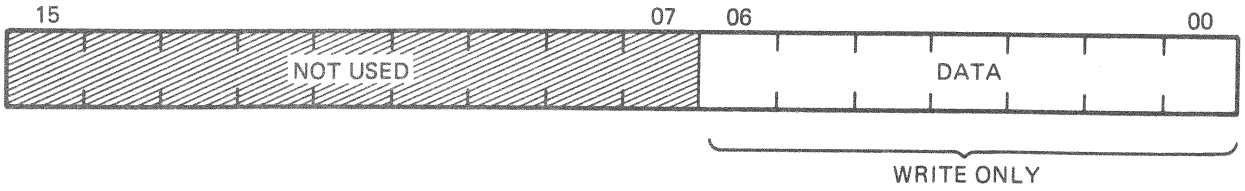
Set when the line printer is ready for the next character to be loaded into the data buffer register. Generates an interrupt if INTERRUPT ENABLE <Ø6> is also set.

Ø6 INTERRUPT ENABLE

When set allows an interrupt to occur when either the ERROR <15> or READY <Ø7> bit is also set. Cleared by loading with a Ø. Also cleared by INIT.

Ø5-ØØ NOT USED

Data Buffer Register (LPDB) 777 516



15-Ø7 NOT USED

Ø6-ØØ DATA

7-bit ASCII character buffer. Characters are transferred to the lineprinter by loading this buffer.

Interrupt vector address	2ØØ
Priority level	BR4

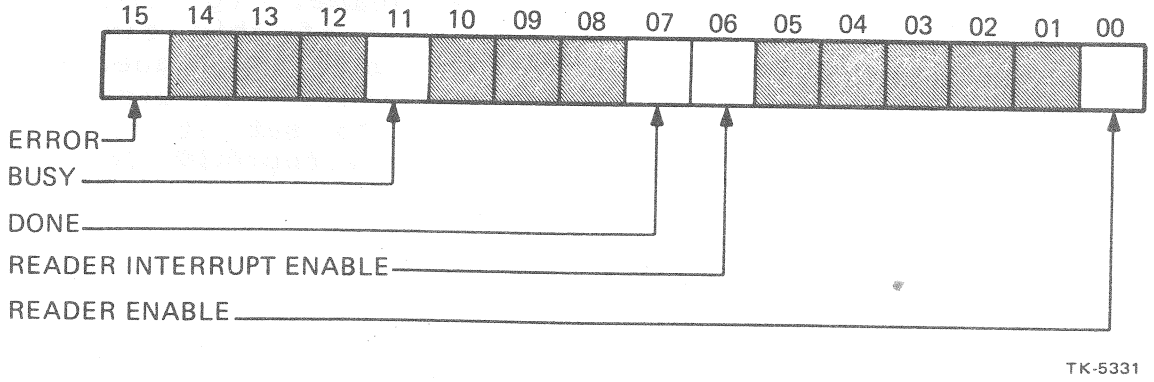
PC11 Paper Tape Reader/Punch Description

The PC11 high-speed reader and punch system consists of a paper tape reader, punch and control. The system reads eight-hole unoiled perforated tape at 50 characters per second. A unit containing a reader only (PR11) is also available.

Operation - In reading the tape, a set of photodiodes translates the presence or absence of holes in the tape to logic levels representing ones and zeros. All information read or punched is paralleltransferred through the controller. When an address is placed on the UNIBUS, the controller decodes the address and determines if the reader or the punch has been selected. If one of the four device register addresses has been selected, the controller determines whether an input or an output operation should be performed. An input operation from the reader is initiated when the processor transmits a command to the Paper Tape Reader Status Register. An output operation is initiated when the processor transfers a byte to the Paper Tape Punch Buffer Register.

The controller enables the PDP-11 system to control the reading or punching of paper tape in a flexible manner. The reader can be operated independently of the punch. Either device can operate under direct program control or can operate without direct supervision, using interrupts to maintain continuous operation.

Paper Tape Reader Status Register (PRS) 777 550



Effect of the initialize (INIT) signal: clear bits 11, 7, and 6.

Read only: bits 15, 11, and 7

Write only: bit Ø

Bit: 15 **Name:** Error

Function: Set when an error occurs: no tape in reader, reader is off-line, or reader has no power.

Bit: 11 **Name:** Busy

Function: Set when a character is being read. It is set when Reader Enable is set and cleared when the present operation is complete (when Done is set).

Bit: 7 **Name:** Done

Function: Set when a character is available in the Reader Data Buffer. It is cleared by any program reference to the Reader Data Buffer or by setting Reader Enable.

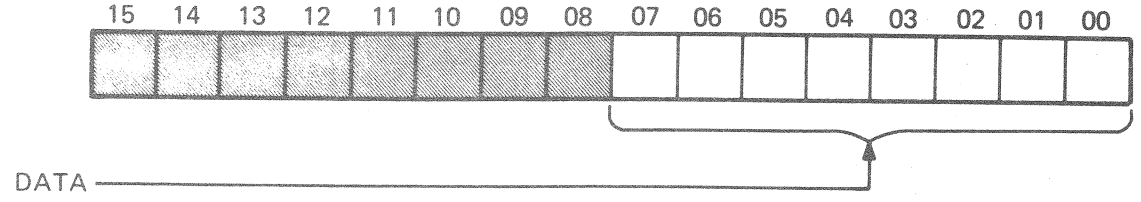
Bit: 6 **Name:** Interrupt Enable

Function: Set to allow Error or Done = 1 to cause an interrupt.

Bit: Ø **Name:** Reader Enable

Function: Set to allow the Reader to fetch one character. The setting of this bit clears Done, sets Busy, and clears the Reader Buffer (PRB). Operation of this bit is disabled if Error = 1; attempting to set it then will cause an immediate interrupt if Interrupt Enable = 1.

Paper Tape Reader Buffer (PRB) 777 552

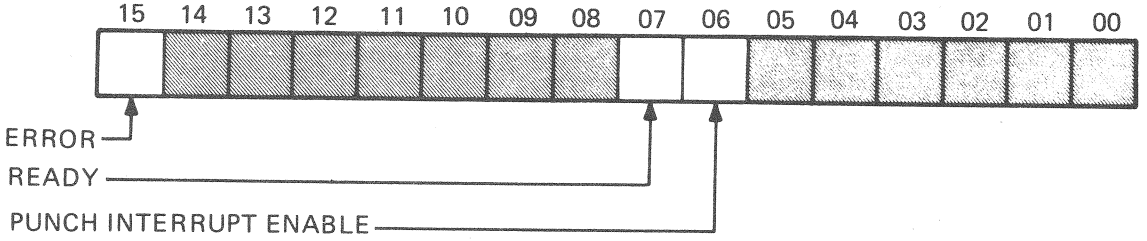


TK-5332

Bits 7 through Ø hold the coded data for the character read. The bits are cleared when Reader Enable, bit Ø of PRS, is set. To the processor, the data is read only.

Any program reference to PRB (777 552 or 777 553) as a word or byte will clear Done, bit 7 of PRS.

Paper Tape Punch Status (PPS) 777 554



TK-5333

Effect of the Initialize (INIT) signal: clear bit 6, set bit 7.

Read only: bits 15 and 7

Bit: 15 **Name:** Error

Function: Set when an error occurs: no tape in punch, or punch has no power.

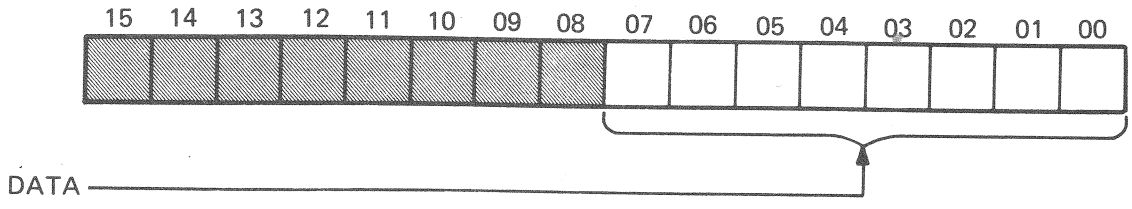
Bit: 7 **Name:** Ready

Function: Set when ready to punch a character. It is cleared when the Punch Buffer is loaded and set when punching is done.

Bit: 6 **Name:** Interrupt Enable

Function: Set to allow Error or Ready = 1 to cause an interrupt.

Paper Tape Punch Buffer Register (PPB) 777 556



TK-5334

Bits 7 through Ø hold the coded data for the character to be punched. To the processor, the data is write only.

An instruction that could modify PPB as a word or byte clears Ready (bit 7 of PPS) and initiates punching. An immediate interrupt will occur when punching is initiated if Error = 1 and Interrupt Enable = 1.

SPECIFICATIONS

Main Specifications

Storage medium	8-hole paper tape, unoiled
Reader speed	3ØØ char/sec
Punch speed	5Ø char/sec
Paper tape	Fanfold
Data format	8-bit characters

Register Addresses

Reader Status	(PRS) 777 55Ø
Reader Buffer	(PRB) 777 552
Punch Status	(PPS) 777 554
Punch Buffer	(PPB) 777 556

UNIBUS Interface

Interrupt vector address	7Ø (for reader); 74 (for punch)
Priority level	BR4 (reader has precedence over punch)
Bus loading	1 bus load

Mechanical

Mounting	1 panel mounted unit + 1 SPC slot
Size	1Ø1/2" front panel height + quad module
Weight	5Ø lb

Power

Input current	3A at 115 Vac; 1.5A at +5V
Heat dissipation	35ØW

Environment

Operating temperature	10°C to 40°C
Relative humidity	10% to 90%

Models

PC11: Reader/punch and control, 115 Vac, 60 Hz
PC11-A: Reader/punch and control, 230 Vac, 50 Hz
PR11: Reader and control
H722: Transformer-lsp (required for 230 Vac, 50 Hz operation of PC11-A or PR11)

UNIBUS Theory and Operation

The **UNIBUS** is a common set of signal wires that connect peripheral devices. Addresses, data, and control information are transmitted along the 56 lines of the UNIBUS. The form of communication is the same for every device on the UNIBUS.

Communication between two devices on the UNIBUS takes the form of a master-slave relationship. During any bus operation, one device has control of the bus **(master)**. The device addressed is called the **slave**. Master-slave relationships are dynamic, i.e., a device may sometimes be a master, sometimes a slave.

Every device on a UNIBUS capable of becoming bus master has an assigned priority. When two devices with equal priority simultaneously request the use of the bus, the device that is electrically closest to the processor receives control.

Because all bus activity is **asynchronous**, communication on the UNIBUS is **interlocked** between devices, i.e., each control signal issued by the master device must be **acknowledged** by a response from the slave to complete the transfer.

To allow the control and organization of peripheral devices, **registers** in peripheral devices are assigned addresses similar to memory. Therefore, instructions that address memory locations can become I/O instructions. Device **control functions** are assigned to addressable registers, and individual bits within a register can cause control operations to occur (e.g., the command to make the paper tape reader read a frame of tape is provided by setting the reader enable bit in the control register of the device). Device **status conditions** are also handled by the assignment of bits within a register. Device status can therefore also be checked by program instructions. There is no limit to the number of registers a device may have, providing an unlimited flexibility in the design and control of peripheral equipment. Depending on the function, registers (and register bits) may be read/write, read only, or write only.

A device generally requests use of the bus for one of two purposes:

a. To make a nonprocessor (NPR) transfer of data directly to or from memory (often referred to as direct memory access, DMA)

b. To interrupt program execution and force the processor to jump to a specific address where an interrupt service routine is located.

Requesting and granting bus mastership is performed parallel with data transfers on a completely independent set of bus lines. Thus, while one device is using the bus, the next request is being checked for priority, and the next user is being assigned. Recall that the priority of a device is a function of:

1. the priority level assigned to the device, and

2. its position on the bus with respect to other devices of the same priority level.

All devices are assigned one of five priority levels: Nonprocessor requests (NPR) and four bus requests (BR4, BR5, BR6, BR7). The NPR has highest priority; BR7 is next highest, and BR4 is the lowest. A signal line (referred to as a request line) is dedicated to each of the levels. A device that requires the use of the data section of the bus asserts a request on one of these lines. The arbitrator (the UBA) issues a grant at the level of the highest priority active request. A grant is a signal that informs a requesting device that it may become bus master after the current master releases the data section of the bus. A grant asserted by the arbitrator is received by the first device on the bus assigned to the same priority level as the grant. If this device requests the use of the data section of the bus, it accepts and acknowledges receipt of the grant (it also blocks the grant from being given to another device).

If the device does not request the use of the data section, it passes the grant to the next device on the same grant line. Grants are issued on the Nonprocessor Grant line (NPG), or on the Bus Grant line (BG7-BG4).

Direct memory access (DMA) transfers can be accomplished between any two peripherals without processor supervision. These are called NPR level data transfers. Normally, NPR transfers are made between memory and a mass storage device (disk). The direct access capability permits operations such as a disk directly refreshing a CRT display. An NPR device provides extremely fast access to the bus and can transfer data at high rates once it gains control. The processor state is not affected by this type of transfer.

Devices that gain bus control with one of the bus request lines (BR7, BR6, BR5, BR4) can take full advantage of the power and flexibility of the processor by requesting an interrupt. Interrupt requests can be made only if bus control has been gained through a BR priority level. An NPR level request must not be used for an interrupt request. When the device gains control of the bus, it sends an interrupt command and a unique address of a memory location which contains the starting address of the device interrupt service routine. This is called the interrupt **vector** address.

The UNIBUS consists of 56 signal lines which may be divided into three functional groups: bus control, data arbitration, and miscellaneous signals (see Table A-1). The **bus control** group consists of those signals necessary to gain bus control through an NPR/BR, or for a priority arbitration to select the next bus master while the current bus master is still in control of the bus. The **data transfer** group are the signals required during data transfers to or from a slave device. The **miscellaneous** group contains the initialization and powerfail signals required by the UNIBUS.

The control line signals **C0** and **C1** are sent by the master to the slave, and indicate one of four possible data transfer operations:

Data In (**DATI**)	C1=0, C0=0 A data word or byte transferred into the master from the slave.
Data In Pause (**DATIP**)	C1=0, C0=1 Similar to DATI except that it is always followed by a DATO or DATOB to the same location.
Data Out (DATO)	C1=1, C0=0 A data word is transferred out of the master to the slave.

Data Out Byte (DATOB)	C1=1, C0=1. Identical to DATO except a byte is transferred instead of a word. Data transferred on: D<15:08> for A00=1 D<07:00> for A00=0

The notations "DATI/P" and "DATO/B" are equivalent to "DATI or DATIP" and "DATO or DATOB", respectively. For detailed descriptions of the DATI/P and DATO/B operations, and the various UNIBUS signal lines, see Chapter 5 of the 1976 PDP-11 Peripherals Handbook.

The UBA, described in the module text, serves as the interface between the asynchronous UNIBUS and the SBI. In addition, the UBA provides the following functions:

- Access to UNIBUS address space (i.e., UNIBUS device registers) from the SBI.
- Mapping UNIBUS addresses to SBI addresses for UNIBUS DMA transfers to SBI memory.
- Data transfer paths for UNIBUS device access to random SBI memory addresses and high-speed transfers for UNIBUS devices that transfer to consecutive increasing memory addresses
- UNIBUS interrupt fielding.
- UNIBUS priority arbitration.
- UNIBUS powerfail sequencing.

For more information on the UNIBUS and UBA, consult the following:

PDP-11 Peripherals Handbook, Chapter 5, 1976 edition

VAX-11/780 Hardware Handbook, Chapter 9

VAX-11/780 DW780 UNIBUS Adapter Technical Description (order number EK-DW780-TD-001)

VAX/VMS Guide to Writing a Device Driver

Table A-1 UNIBUS Signals

Name	Mnemonic	Number of Lines	Function
DATA TRANSFER			
Address	A<17:00>	18	Selects slave device or memory address
Data	D<15:00>	16	Information transfer
Control	C0,C1	2	Type of data transfer
Master Sync	MSYN	1	Timing control for
Slave Sync	SSYN	1	data transfer
Parity	PA,PB	2	Device parity error
Interrupt	INTR	1	Interrupt
BUS CONTROL			
Bus Request	BR4-BR7	4	Requests use of bus (interrupt)
Bus Grant	BG4-BG7	4	Grants use of bus (for BRs)
Nonprocessor Request	NPR	1	Requests use of bus (data)
Nonprocessor Grant	NPG	1	Grants use of bus (for NPR)
Selection Acknowledge	SACK	1	Acknowledges grant
Bus Busy	BBSY	1	Data section is in use
MISCELLANEOUS			
Initialize	INIT	1	System reset
AC Low	ACLO	1	AC power monitoring
DC Low	DCLO	1	DC power monitoring

See the VAX-11/780 Hardware Handbook for more information.

The DR32

The DR32 is an architectural specification for a bus called the DR32 Device Interconnect (DDI). (The DR78Ø is an implementation of the DR32 architecture that connects the DDI to the SBI as shown in Figure A-1.) Throughout this discussion, you could substitute DR78Ø for DR32. Two DR32s can be connected to form a CPU-CPU link. The DR32 is capable of moving data to or from memory at much faster speeds than the UBA or MBA. If the data is to be stored on a disk, it must go through an intermediate buffer in main memory.

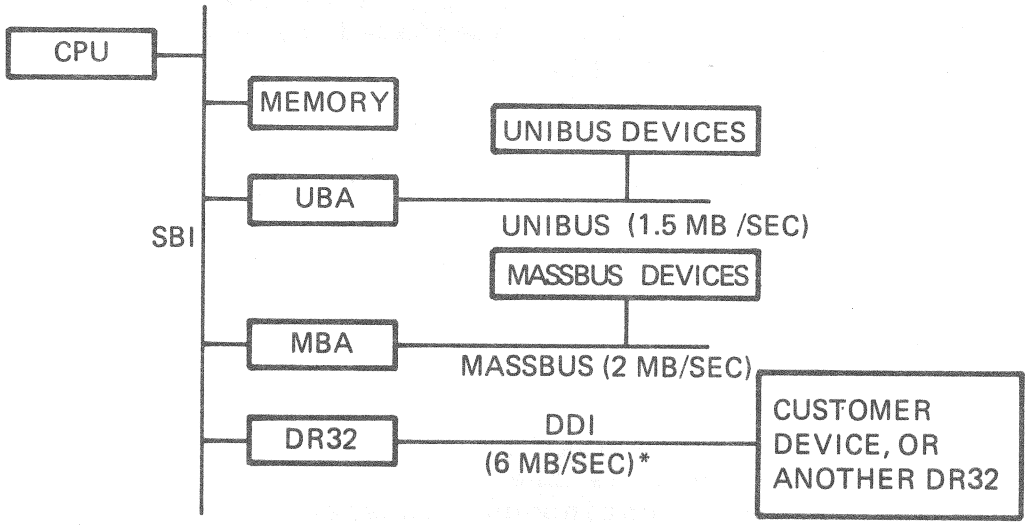


Figure A-1 Where DR32 fits into the I/O Architecture (The DR32 is an SBI nexus.)

The DDI is electrically like the MASSBUS, with extra shielding (i.e., uses similar cable). Its maximum length is 1Ø0 feet.

DR32 Features and Capabilities

- 32-bit parallel data transfers
- High bandwidth (6 MB/sec)
- Half-duplex operation

  Data may be transferred in one direction or the other, but not both at the same time.

- Point-to-point interconnect

  Only one device can be interfaced to a DR32 (i.e., does not support multidrop).

- Command and data chaining

  Command chaining - user can specify a sequence of commands for the device to perform (an unlimited number); the device needs only the command to go, after which it performs all requested commands. Commands may be added to the sequence of commands to perform as the device is performing, without reactivating the device.

  Data chaining - data transfers of unlimited length are possible; the user must be able to "feed" buffers to the hardware (as fast as the hardware can go). The result appears as one continuous transfer. Often, the device fills a second buffer while the first is being emptied by the user (e.g., copied onto disk). The buffers then reverse roles, and the entire process is repeated until the transfer terminates.

- Can provide CPU to CPU link.
- Supports large transfers (>64K bytes).
- Random access (byte alignment) of data.

  Data can be aligned on any byte boundaries, and scattered in memory (i.e., nonconsecutive). However, throughput may be reduced in such a case to less than 6 MB/sec.

- Transfers may be initiated by external device provided a user process has set up buffers to receive the data.
- Direct communication with user process (no driver involvement).

- Hardware-supported memory mapping.
- Separate paths for control and data information (can be overlapped).
- User program may be synchronized with DR32 data transfers.

Control and Data Paths

Figure A-2 illustrates how the DR32 supports separate control and data paths. Note that overlapping control and data messages can cause synchronization problems, e.g.:

1. Data is sent from one CPU into another CPU's memory.
2. A control message is sent to the other processor saying that data was just stored in its memory.
3. Due to prefetching and overlapping of control and data, the control message may arrive first.

The solution to this problem is provided by **action routines.** Action routines are called when an operation completes. The request to send data could terminate with an action routine, and the action routine could send the control message (guaranteeing that the data has already arrived before the control message is sent).

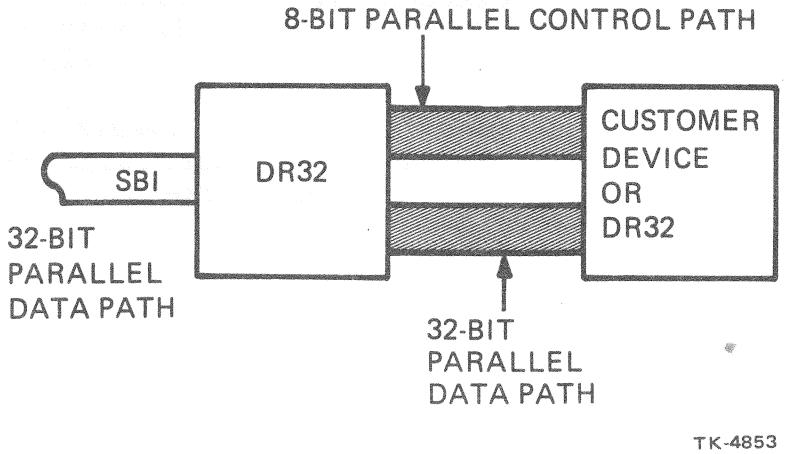


Figure A-2 How DDI supports separate control and data paths which allow overlapping of control and data messages.

Command Packets

The communication between a user process and the dr32 takes place with command packets built by the user. These command packets, and the user process interface, are described in detail in Chapter 11 of the VAX/VMS I/O User's Guide.

The command packets are placed in three queues located within the user pØ space:

- INPTQ (input queue)
- TERMQ (termination queue)
- FREEQ (free queue)

As long as there are command packets on the input queue, the DR32 will continue to process those packets (command chaining). Figure A-3 illustrates the relationships among the various queues.

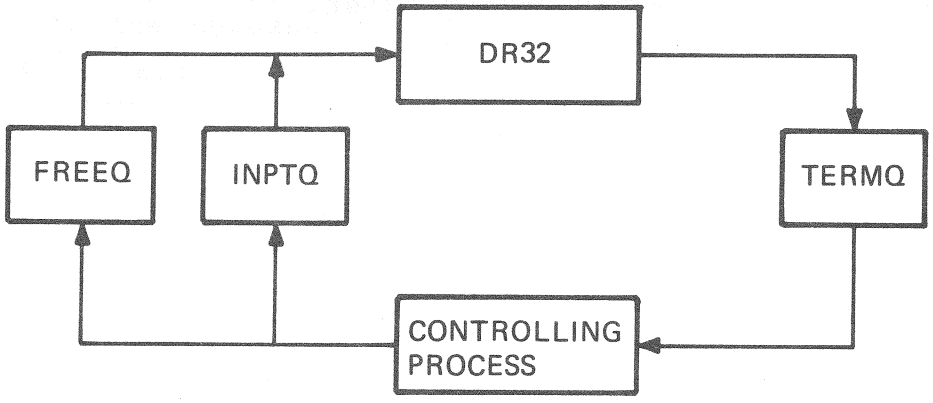


Figure A-3 The relationships among the three queues containing command packets, the DR32 and the process controlling the DR32. The arrows indicate the flow of DR32 command packets.

Command Packet Processing

The DR32 is a nonshareable device, i.e., only one user process can use it at a time. The following sections outline how the user process and dr32 interact via command packets on the free, input, and termination queues.

Software

- user builds command packet.
- user inserts packet on INPTQ.
- user tells hardware about packet by setting a GO bit, which is in a page in system space, and thus mapped to the user process. The driver changes the protection on that page to user-writeable.
- no driver involvement.

Hardware

- DR32 removes packet from queue.
- DR32 performs requested operation.
- when operation complete, command packet inserted on TERMQ.
- if user requested an interrupt on completion of the operation (in command packet), an interrupt is generated.

The user specifies in the command packet in the interrupt control bits whether the DR32 should respond by:

- always generating an interrupt.
- never generating an interrupt.
- interrupting only when the TERMQ is empty.

The FREEQ supplies a "pool" of command packets that can be filled in by the DR32 to describe an unsolicited interrupt.

Hardware

- unsolicited interrupt occurs
- DR32 removes empty packet from FREEQ
- fills in information
- puts packet on TERMQ
- generates an interrupt*

*Always receives an interrupt when an unsolicited interrupt occurs. Only where the user builds the command packet can the interrupt be suppressed.

Also, all interrupts are reported to the user by triggering an AST routine.

Software

- user removes packet from TERMQ
- user examines status bits
- user processes information
- user puts packet on INPTQ or FREEQ

Software Interface

The user can issue only two kinds of $QIOs to the DR32:

- IO$_LOADMCODE — to load DR32 microcode (usually as part of system startup)
- IO$_STARTDATA — to start device and specify command packet queues

The function code modifier io$m_setevf may be used to request that the event flag be set at the end of every transfer for IO$_STARTDATA.

The P1 parameter is used to specify the address of a buffer in which information such as the command block and buffer block sizes and addresses are specified. The P2 parameter is used to specify the size of the buffer pointed to by P1 (see Figure A-4).

The driver locks the command and buffer blocks in physical memory. It also converts interrupts to ASTs, so that the process is notified of interrupts via ASTs.

Note that for device initiated transfers, the user image may issue a $QIO to the driver without putting a command packet onto the INPTQ. For example, a graphics processor may look at a buffer in memory containing display data, and update that data every second.

Rather than issuing $QIOs, the user may choose to use a high-level language procedure library of support routines. These routines, as well as the $QIO interface, are documented in Chapter 11 of the VAX/VMS I/O User's Reference Manual

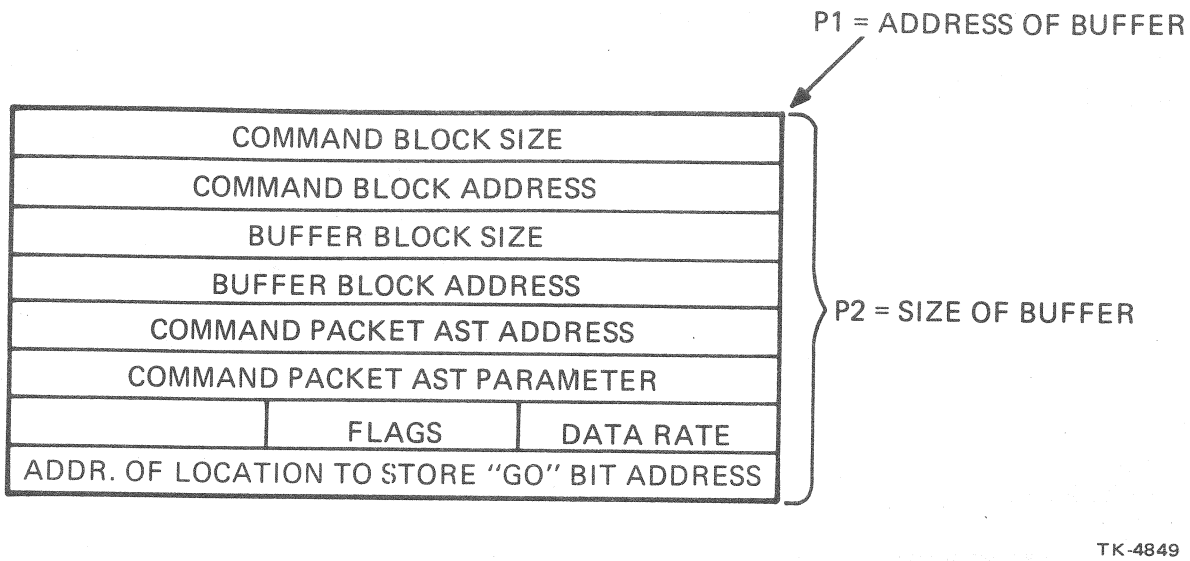


Figure A-4 Format of the Buffer Specified in the P1 and P2 Parameters of the IO$_STARTDATA $QIO

The **command block** contains the headers for the three queues (INPTQ, FREEQ, TERMQ), and space in which to build command packets.

The **buffer block** defines the area of memory that is accessible to the DR32 for the transfer of data between the DR-device and the DR32.

Use of IRPEs

To allow the user process to specify more than one buffer, and to allow transfers of more than 65535 bytes, the DR32 driver uses IRPEs (as discussed in the FDT Routines module). Note that both regions specified in IRPE$L_SVAPTE1 and IRPE$L_SVAPTE2 are unlocked by IOPOST. Note also that the BCNT fields are longwords, rather than words, to allow >64K byte transfers.

Figure A-5 illustrates the fields of greatest importance in the use of IRPEs by the DR32 driver.

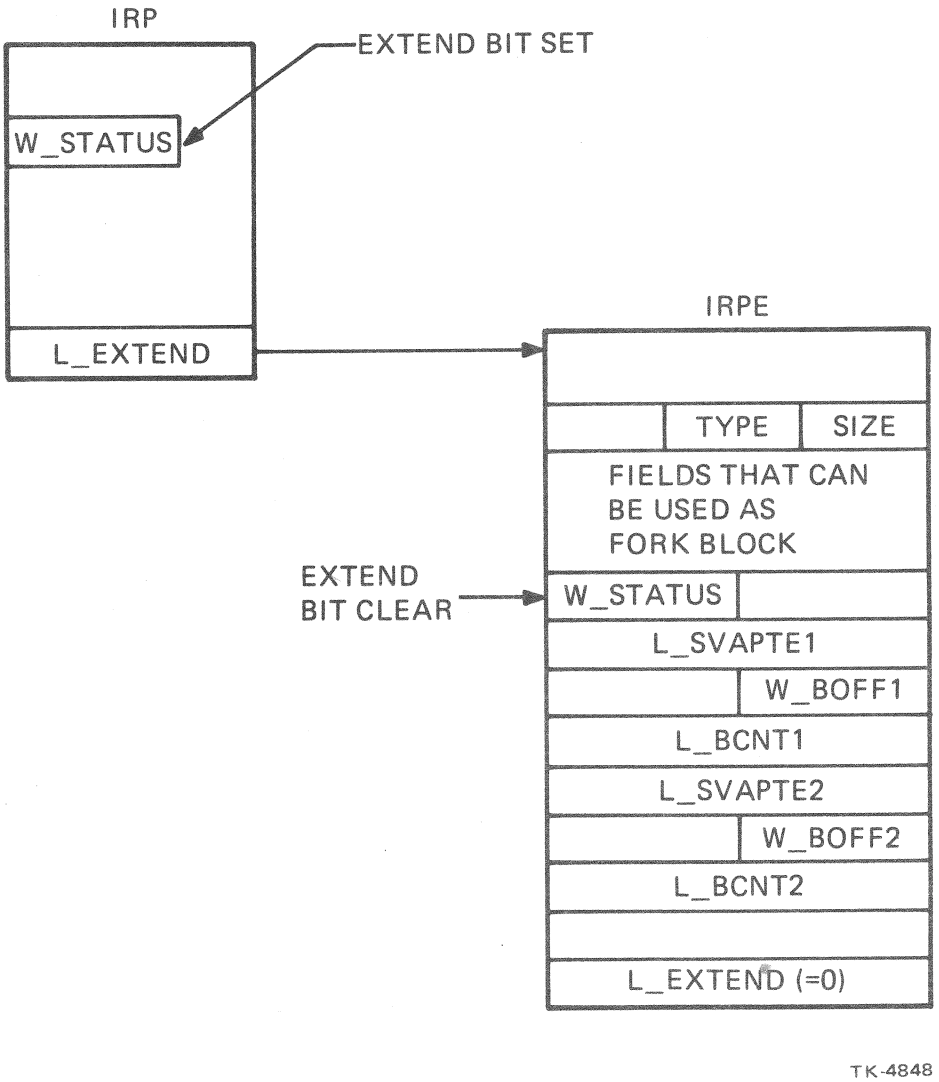


Figure A-5 Relationship Between IRP and IRPE Used by DR32 Driver

digital

EY-2278E-ML-0001
Printed in U.S.A.