EDUCATIONAL
SERVICES

VAX/VMS
Training

VAX/VMS
Device Driver
Tests and Answers

digital

EY-2278E-MN-0001

VAX/VMS Device Driver

Tests and Answers

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

10. RELATED TOPICS

11. I/O ARCHITECTURE

Overview

TEST

1. Define:

   a. device driver

   b. programmed I/O

   c. driver fork process

2. Distinguish between the following topics:

   a. wait queue-fork queue

   b. device dependent parameters-device independent parameters

   c . buffered I/O-direct I/O-programmed I/O

Overview

TEST

3. Diagram the basic organization of a device driver.

4. List four functions a device driver may perform.

Overview

ANSWER SHEET

1. a. A **device driver** is a set of tables and routines that controls I/O operations on a peripheral device.

   b. **Programmed I/O** is a data transfer method in which the CPU transfers one word or byte at a time using device registers.

   c. A **driver fork process** is a dynamically created execution agent (with minimal context) that executes driver code in order to complete an I/O request.

2. a. Driver fork processes are placed into wait queues because they need unavailable resources, while driver fork processes are placed into fork queues because they want to lower their IPL.

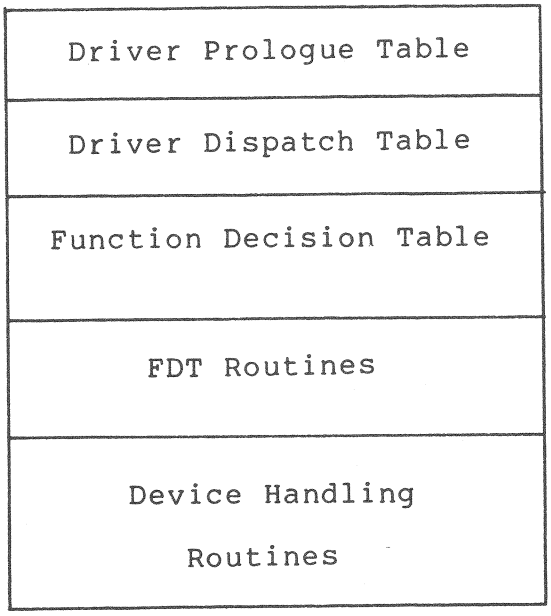
   b. Device independent parameters (in a $QIO request) are those which have the same meaning for all device units in the system (efn, iosb, astadr, astprm, func, chan); device dependent parameters (P1-P6) may have different meanings for different types of devices. The contents of P1-P6 are determined by the driver writer.

   c. Programmed I/O is a data transfer method; direct and buffered I/O refer to where data is stored (in buffers) during the transfer (a hardware characteristic). For buffered I/O, the transferred data is stored in a system buffer (before being put in a user buffer); in direct I/O, the data is transferred directly into the user buffer (which is locked into physical memory).

Overview

ANSWER SHEET

3. The basic organization of a device driver is:



4. A device driver may perform any four of the following:

- Initialize Device (unit and/or controller)
- Validate User I/O Requests
- Start I/O on Device
- Request/Handle (Un)expected Device Interrupts
- Log/Respond to Device Errors/Timeouts
- Cancel I/O Operations on Device
- Complete I/O Requests

I/O Data Structures

TEST

1. Draw a diagram showing the relationship among the UCB, DDB, and CCB data blocks.

I/O Data Structures

TEST

2. One argument to the $QIO service is the channel number. How is this used to locate the associated device?

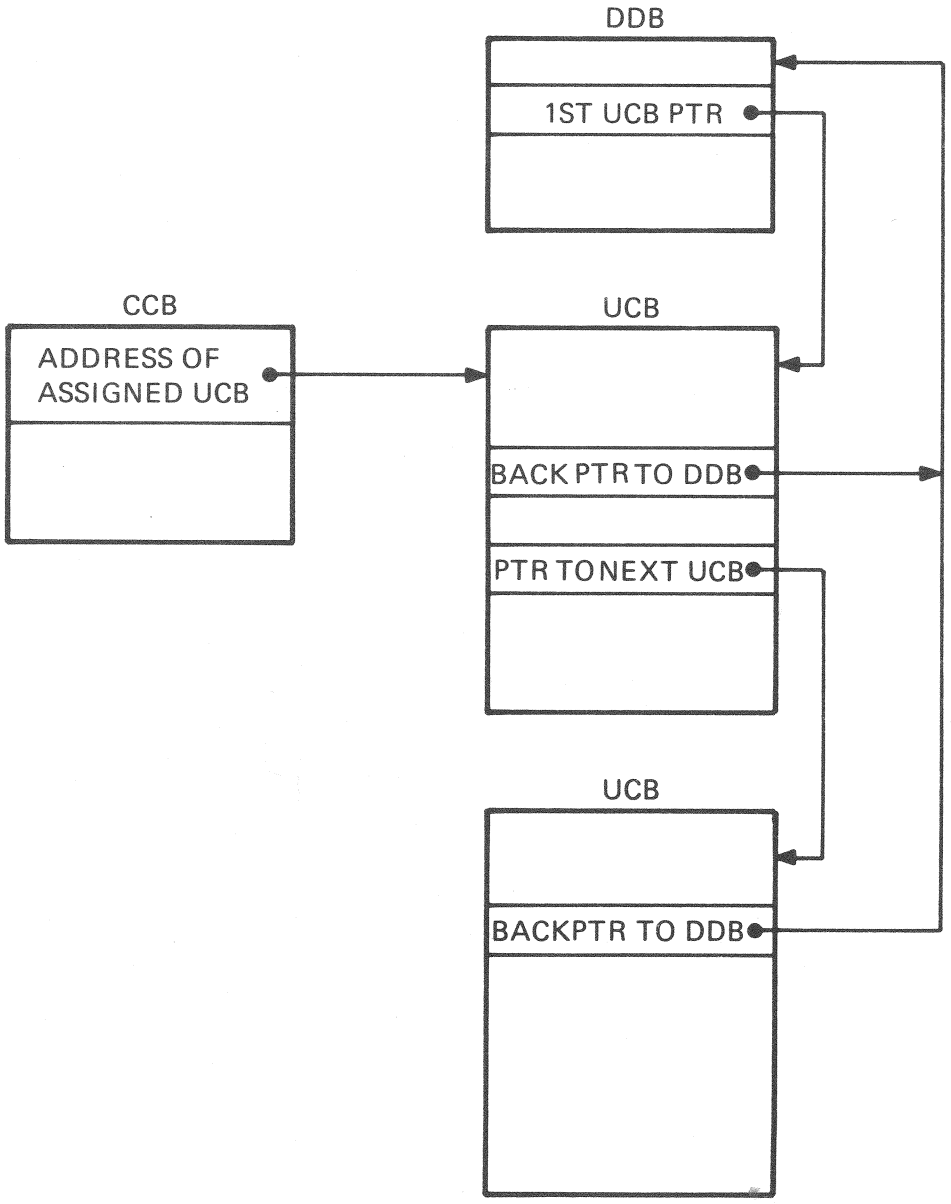
3. Describe the contents of the FDT and explain the functions of this data block.

4. Which data blocks (in sequence) must be examined to dispatch to the FDT routines?

I/O Data Structures

ANSWER SHEET

1.



TK-4871

I/O Data Structures

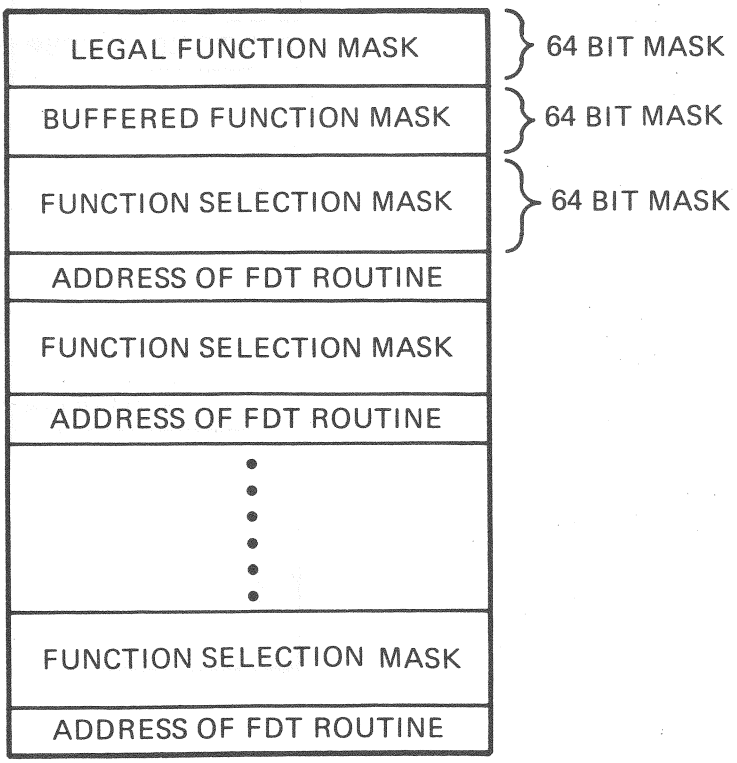
ANSWER SHEET

2. The channel number is used as an index into a channel table in the process control region to locate the CCB. The CCB contains a pointer to the UCB of the associated device.

3. FDT Functions:

   - Is used to check the legality of the function code specified for this device driver.
   - Is used to indicate which functions require an ancillary memory buffer (buffered I/O functions).
   - Contains a table of dispatch pointers to FDT routines used to perform device/function dependent preprocessing in validating the $QIO request and completing the IRP.

4. CCB, UCB, DDT, FDT.



TK-4874

I/O Sequence

TEST

1. Draw a diagram showing the major steps involved in a typical successful I/O operation from I/O request to I/O complete.

I/O Sequence

TEST

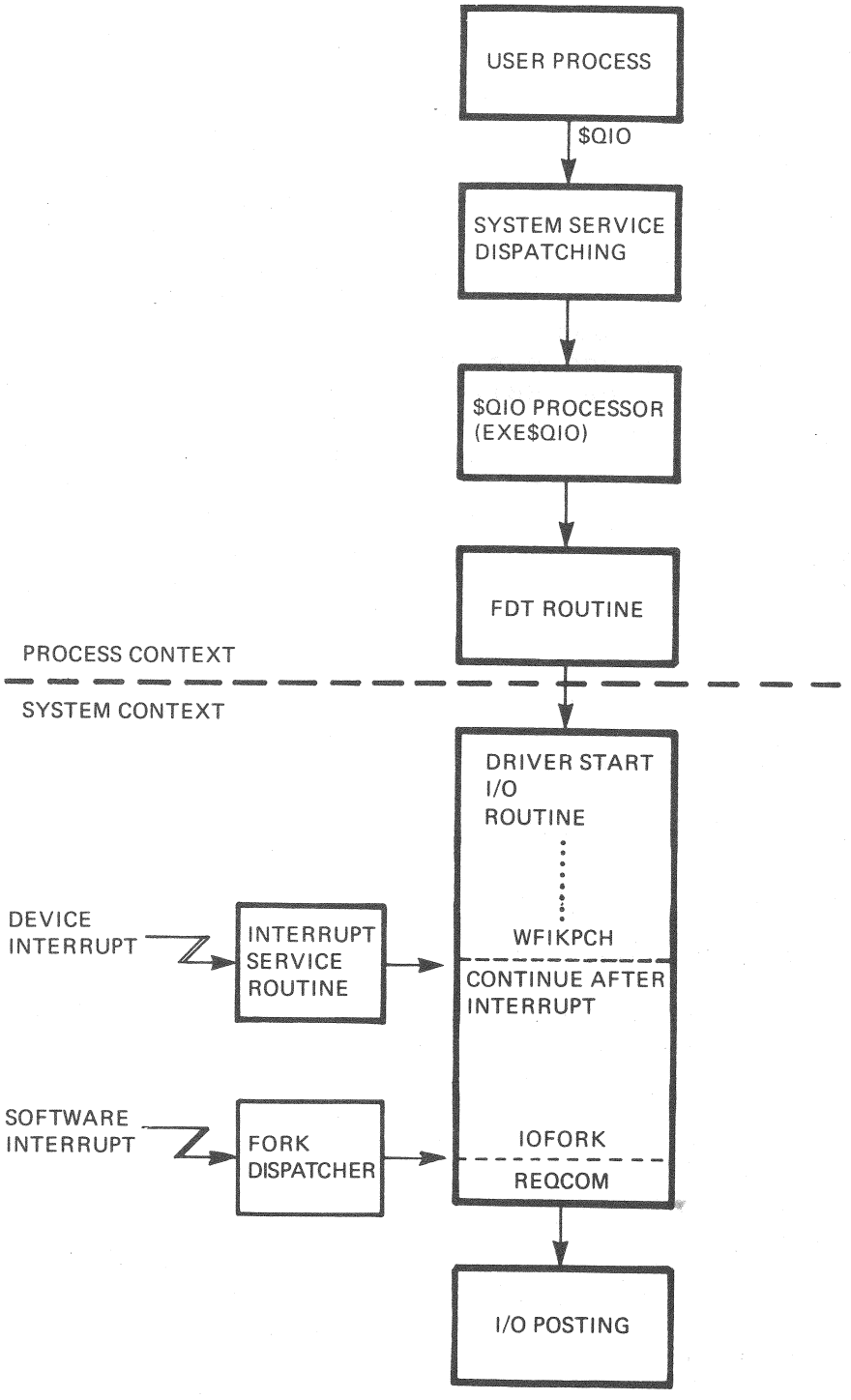
2. List the sequence of events that takes place from a device interrupt to entering the device interrupt service routine for an 11/780 UBA device.

3. List the sequence of events that takes place in I/O posting.

I/O Sequence

ANSWER SHEET

1.



I/O Sequence

ANSWER SHEET

2. For a device interrupt:

    1. Device Interrupt (at some BR level).

    2. Vector through the System Control Block to UBA interrupt service routine (corresponding to device BR level).

    3. UBA interrupt service routine saves general registers, reads BRRVR to obtain device UNIBUS vector address, uses UNIBUS vector address as index into VECTAB to obtain CRB address, and passes control to code in CRB.

    4. Code in CRB does a JSB into the device interrupt service routine (the address of the IDB is pushed on the stack as a result of the JSB).

    5. The driver interrupt service routine locates the associated UCB through the owner field of the IDB (or through polling of the device registers, e.g., reading the attention summary register in RKØ7), depending on whether or not the owner field is used. By restoring the driver context stored in the fork block of the UCB, the instruction following the WFIKPCH macro is executed.

3. For I/O posting:

    1. I/O Post Interrupt

    2. I/O Post Routine entered

    3. Queue Special Kernel Mode AST

    4. Recompute ASTLVL

    5. Reschedule if current process has lower priority than the one to receive the AST

    6. AST interrupt

    7. AST dispatching

    8. Special kernel AST Routine

    9. Queue Process AST

    1Ø. Recompute ASTLVL

Driver Incorporation

TEST

1. Write an assembler command to assemble a driver named XYDRIVER in the file ABCDEF.MAR under directory [XXX.YYY] on DB1.

2. Write a linker command to link the above driver.

3. List the necessary SYSGEN commands to incorporate the above driver into an 11/780 system, assuming the following conditions:

   Adapter TR# = 3

   Vector = 230_8

   Number of Vector = 1

   CSR Address = 777160_8

   Adapter Unit Number = 0

Driver Incorporation

ANSWER SHEET

1.
```
$MACRO/LIST=XYDRIVER/OBJECT=XYDRIVER -
$_DB1:[XXX.YYY]ABCDEF+SYS$LIBRARY:LIB/LIB
```

2.
```
$ CREATE    MYDRIVER.OPT
BASE=Ø
<control-Z>

$ LINK/NOTRACE/MAP/FULL XYDRIVER,-
$_MYDRIVER.OPT/OPTIONS,-
$_SYS$SYSTEM:SYS.STB/SELECTIVE
```

3.
```
$RUN SYS$SYSTEM:SYSGEN
SYSGEN>CONNECT XYAØ/ADAPTER = 3 -
/VECTOR = %O23Ø -
/CSR = %O777160
SYSGEN>EXIT
$
```

Driver Tables

TEST

1. List the data structures that must be initialized by a driver.

2. List the fields in the UCB which must be initialized in the driver prologue table. What is the function of each field?

3. Use the $DEF and _VIELD macros to define register offsets and bit field symbols for the paper tape reader/punch. (See Appendix C of the Student Guide for a description of the paper tape reader/punch registers.)

4. Use the DPT, DDTAB, and FUNCTAB macros to define the driver prologue table, driver dispatch table, and function decision table for the paper tape reader/punch. Assume the driver controls both the reader and punch (so you have two interrupt vectors), and that the driver processes $QIOs with function codes of IO$_READVBLK and IO$_WRITEVBLK.

5. Use the $DEF macros to add extensions to a UCB with the error-logging extensions. Add two words and a longword.

Driver Tables

ANSWER SHEET

1. DDB, UCB, CRB, DDT, FDT

2.

UCB$B_FIPL	Fork IPL at which driver fork processes will execute.
UCB$L_DEVCHAR	Characteristics of the device for which the driver is written.
UCB$B_DIPL	Hardware interrupt IPL at which device generates interrupts.

3. See the Solution Driver for an answer to this question. If you have any questions, see your instructor.

4. See the Solution Driver for an answer to this question. If you have any doubts about your answer, see your instructor.

5. See the Solution Driver for an answer to this question. If you have any questions, see your instructor.

FDT Routines

TEST

1. Define the following terms:

   a. position-independent code

   b. reentrant code

2. List seven exit methods used in FDT routines. Which method is used when the IRP is to be passed to the start I/O entry point of the driver?

3. Write FDT routines to support punch and read operations for the paper tape reader/punch. Assume IO$_WRITEVBLK is used for punching, P1= buffer address of data to be punched, P2= buffer size, and IO$_READVBLK is used for reading, with P1= buffer address into which data should be read, P2= buffer size. The FDT routines should perform buffered I/O.

FDT Routines

ANSWER SHEET

1. a. **position-independent code** - works the same regardless of its position in memory.

   b. **reentrant code** - works the same every time (same logic, same output) given the same input.

2. 1. RSB
   2. JMP G^EXE$ABORTIO
   3. JMP G^EXE$FINISHIO
   4. JMP G^EXE$FINISHIOC
   5. JMP G^EXE$QIODRVPKT
   6. JMP G^EXE$QIOXQPPKT
   7. JMP G^EXE$ALTQUEPKT

   JMP G^EXE$QIODRVPKT is used when the IRP is to be passed to the start I/O entry point of the driver.

3. See the Solution Driver for an answer to this question. If you have any doubts about your answer, see your instructor.

Debugging

TEST

1. What command(s) are used to invoke XDELTA at the system console?

2. What are the commands used to assemble, link, and run a program called PROG (in directory [DIR]) with the DELTA debugger?

3. Use the CHMK program to display the contents in the ADP vector table (VECTAB) from VECTAB+ 3Ø (hex) through VECTAB+ 6Ø(hex). Note that most vectors point to the unexpected interrupt service routine. If possible, load a paper tape reader/punch driver and verify that the addresses in the vector table at offsets 7Ø and 74 (octal) point to the paper tape reader/punch CRB, from which there are pointers to interrupt service routines in the driver.

Debugging

TEST

4. Invoke XDELTA to do the following:

   a. Store a command string starting at address 80000E58 which will display the contents of 80000A88 and use the contents as an address for displaying the next contents.

   b. Set a breakpoint at 8000B17E.

   c. Display the contents of address 80000000 when the breakpoint set in step b is reached.

   d. Execute the command string stored in step a.

   e. Proceed from breakpoint.

   The addresses given in this test are hypothetical. When actually performing the test, the addresses should be selected as follows:

80000E58	Address of an unused UCB. The program SYSGEN can show you all the driver data structure addresses. Pick a device unit control block you do not need to run the system standalone.
80000A88	Address of the first DDB.
8000B17E	Any instruction address in XDELTA following the first breakpoint. This can be selected by single stepping through XDELTA after the first breakpoint.
80000000	Any address you wish to display.

Debugging

TEST

5. What are the commands necessary to invoke SDA, to determine the cause of a system crash, and to view device-related data structures (say, for device PT:)?

6. List three requirements for a driver test program.

Debugging

ANSWER SHEET

1. <control-P>
   >>> HALT <return>
   >>> D/I 14 5 <return>
   >>> CONTINUE <return>

2. $MAC [DIR]PROG +SYS$LIBRARY:LIB/LIB
   $LINK/DEBUG [DIR]PROG, SYS$SYSTEM:SYS.STB/SELECTIVE
   $DEFINE LIB$DEBUG DELTA
   $RUN [DIR]PROG

3. a) First, invoke SYSGEN: $MCR SYSGEN

   b) Issue a SHOW /DEVICES command. If the paper tape read/punch driver is loaded, say PTDRIVER, use a SHOW /DEVICES = PT SYSGEN command.

   c) Record the values listed, and EXIT from SYSGEN.

   d) To invoke the CHMK program, type $MCR CHMK, followed by 215;B;P (to invoke DELTA).

   Use DELTA commands to:

   - Add offset IDB$L_ADP to the IDB base address. (Use this value as a pointer to the ADP.)
   - Add offset ADP$L_VECTOR to the ADP base address (found above) to find VECTAB base address.
   - Display above-address +3Ø (hex) through above-address +6Ø (hex).
   - Verify that the paper tape reader/punch vectors (VECTAB+ 7Ø (octal) and VECTAB+ 74 (octal)) point into CRB.
   - Display (as word) CRB address pointed to (will be JSB instruction). Display (as longword) address of interrupt service routine (should be in range of driver start and end addresses).
   - The following is a sample solution. Other steps/commands could be used. Specific addresses vary from system to system (even from system boot to system boot).

Debugging

ANSWER SHEET

```
$ MCR SYSGEN
SYSGEN>  SHOW /DEVICES=PT
   Driver      Start     End     Dev   DDB      CRB      IDB      Unit  UCB
PTDRIVER    8005ABC0 8005B1B0
                                 PTA 80059250 80059B80 800591C0
                                                                    0 80059EB0
                                                                    1 8005A560
SYSGEN>  EXIT
$ MCR CHMK
1 BRK AT 00000D7A 215$B$P <CR>   START CHMK PROGRAM

2 BRK AT 00000215 G591C0+C/80052000 <CR>   FIND IDB$L_ADP (TO GET ADP ADDRESS)
Q+10/80052114 <CR>                         FIND ADP$L_VECTOR (TO GET VECTAB ADDRESS)
Q+30,Q+60/8000B218
80052148/8000B218
8005214C/80059B96        } READER VECTOR
80052150/80059BBA        } PUNCH VECTOR
80052154/8000B218
80052158/8000B218
8005215C/8000B218        VECTAB
80052160/8000B218        TABLE
80052164/8000B218
80052168/8000B218        (MOST GO TO
8005216C/8000B218         UNEXPECTED
80052170/8000B218         UBA INTERRUPT
80052174/8000B218 <CR>    SERVICE ROUTINE)
CW  <CR>
G59B96/9F16 CL <CR>      } JSB INSTRUCTION IN CRB (NOTE G59B96 IN CRB)

80059B9A/91C08005 <CR>   OOPS!
G59B96+2/8005B11F <CR>   INT. SERVICE ROUTINE ADDRESS (READER) [IS IN DRIVER]
CW  <CR>
G59BBA/9F16 CL <CR>      JSB INSTRUCTION IN CRB (DIFFERENT VEC FIELD)
G59BBA+2/8005B116 <CR>   INT. SERVICE ROUTINE ADDRESS (PUNCH) [IS IN DRIVER]
$P   LEAVE DELTA

 EXIT 00000001  SS$_NORMAL – ALL IS OKAY
8000F67D/50D503BA EXIT  LEAVE PROGRAM
```

Debugging

ANSWER SHEET

4. The following commands will invoke XDELTA:

```
<control-P>
>>>H <RET>
>>> D/I 14 5 <RET>
>>>CONTINUE <RET>
```

a.

```
1 BRK AT 8000B17D [B
[B
80000E58/00 5B

80000E59/00 4C

80000E5A/00 D

80000E5B/00 38

80000E5C/00 30

80000E5D/00 30

80000E5E/00 30

80000E5F/00 30

80000E60/00 41

80000E61/00 38

80000E62/00 38

80000E63/00 2F

80000E64/00 2F

80000E65/00 2F

80000E66/00 D

80000E67/00
80000E58;E
80000E58 80000BE4 80000D78 80000E20
;P
```

Debugging

ANSWER SHEET

4. b. 8000B17E; B

   c. ;P
   2 BRK AT 8000B17E <RET>
   80000000/ 8FBC0FFC

   d. 80000E58;E

   e. ;P
   Steps b through d could also be executed with a single command:

   8000B17E, 2, 80000000, 80000E58; B <RET>

   ;P

5. $MCR SDA

   Enter dump file name> <return>

   SDA> SHOW CRASH (determine cause of crash)
   SDA> SHOW DEVICE PT (show PT data structures)

6. A driver test program must

   - be debugged.
   - test all driver-supported function codes.
   - test all possible paths through the driver.
   - test all conditions relevant to the driver.

Required Driver Routines

TEST

1. List two restrictions on the use of the instruction set when accessing device registers in UNIBUS drivers.

2. Write the following routines for a paper tape reader/punch driver. The driver is to perform buffered I/O, and is to service both the reader and punch (i.e., respond to interrupts from both reader and punch vectors).

   - start I/O routine
   - interrupt service routine(s)
   - timeout routine

Required Driver Routines

ANSWER SHEET

1. 1. Must use noninterruptible instructions.

   2. Must reference device registers only as words or bytes, as appropriate.

2. See the Solution Driver for an answer to this question. If you have any doubts about your answer, see your instructor.

Optional Driver Routines

TEST

1. Write the following routines for a paper tape reader/punch driver:

   - cancel I/O routine
   - unit initialization routine
   - controller initialization routine
   - register dump routine

2. Incorporate error logging capabilities into the paper tape reader/punch driver.

3. Cause an error (or timeout) to occur on the paper tape reader/punch, obtain an ERF report describing the error, and verify that the dumped register contents indicate the appropriate error condition generated.

Optional Driver Routines

ANSWER SHEET

1. See the Solution Driver for an answer to this question. If you have any doubts about your answer, see your instructor.

2. See the Solution Driver for an answer to this question. If you have any doubts about your answer, see your instructor.

3. See your instructor.

Related Topics

TEST

1. Write a driver for the paper tape reader/punch that uses the set attention AST scheme to allow a user to control the reader/punch from AST routines. Write FORTRAN AST routines that read/punch data to test your driver.

2. Which components of the executive may need to be modified when writing a device driver for disks?

3. What four components are necessary for an ACP-$QIO interaction to be completed?

4. What is the most important coding technique to remember for increasing execution speed?

5. Write a program that uses the connect to interrupt scheme to control the lineprinter. The program should prompt a user for data to be printed, and in the AST routine (for each interrupt), display the printed character. The lineprinter registers are identical to those of the paper tape punch with a CSR address of ^O777514, and an interrupt vector of ^O200.

Related Topics

ANSWER SHEET

1. See the Solution Set Attention AST Driver and FORTRAN AST Routines for an answer to this question. If you have any doubts about your answer, see your instructor.

2. INIT
   MOUNT

3. Mount Image
   Dismount Image
   ACP itself
   Driver for device (which interacts with the ACP)

4. Naturally align data on longword boundaries.

5. See the solution connect to interrupt program listings for an answer to this question. If you have any doubts about your answer, see your instructor.

I/O Architecture

TEST

1. Draw a diagram of the basic bus configuration of the VAX-11/780.

I/O Architecture

TEST

2. a. How are SBI addresses translated into UNIBUS addresses.

   b. Give one instance where this translation would be required.

3. a. How are UNIBUS adresses translated into SBI addresses.

   b. Give one instance where this translation would be required.

I/O Architecture

TEST

4. In each case listed below, state whether it would be most appropriate to use the direct, buffered, or no data path:

   a. to transfer to an odd SBI address

   b. to transfer to nonconsecutive addresses

   c. to mix read/write functions

   d. to use the MBA

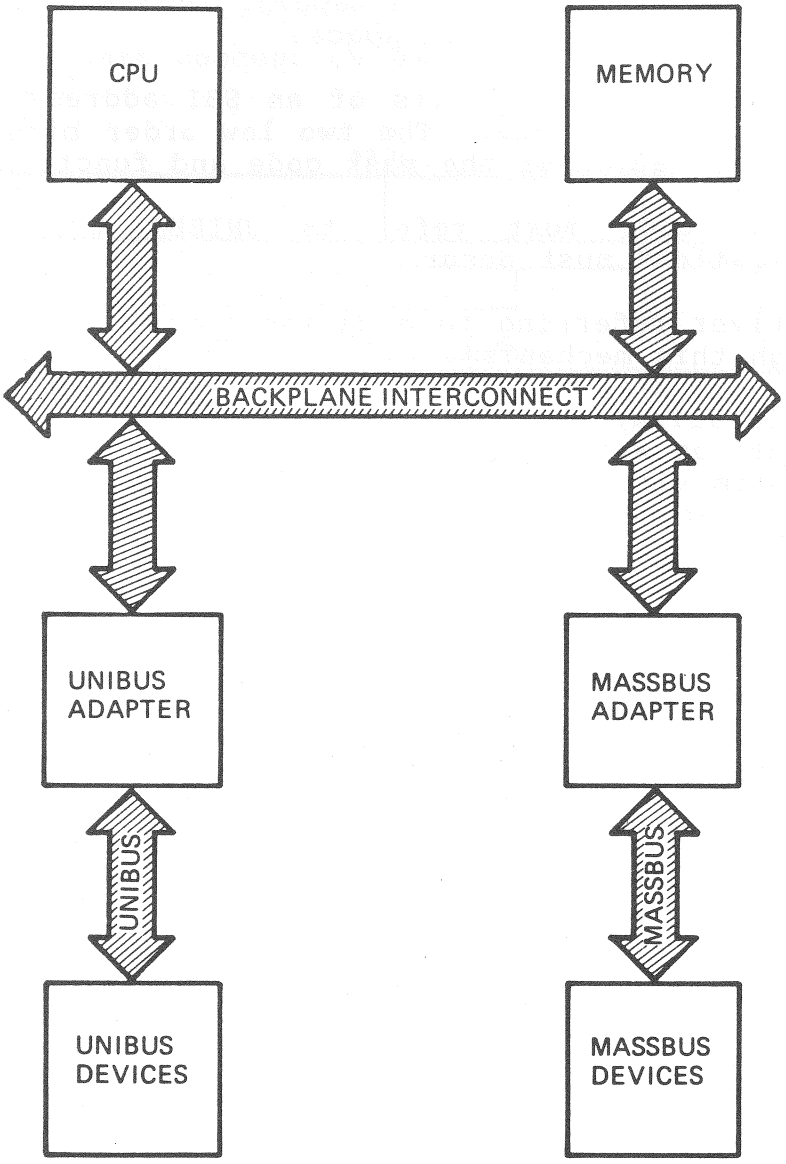
   e. to transfer data as fast as possible

5. Diagram the MASSBUS to SBI address translation scheme.

I/O Architecture

ANSWER SHEET

1.



I/O Architecture

ANSWER SHEET

2. a) Since an SBI address is 28-bits and a UNIBUS address is 18-bits long, there is no mapping involved. A subset of SBI addresses map UNIBUS space.

   In fact, the low 16 bits of an SBI address form UNIBUS address bits <17:Ø2>. The two low order bits are formed by a combination of the mask code and function code.

   b) Whenever code must refer to UNIBUS I/O space, this "translation" must occur.

   Any driver referring to a device register would be going through this mechanism.

   At initialization time, the VMS executive loads certain SPT entries with appropriate PFN fields so that a portion of system virtual address space maps the UNIBUS I/O page. (The lower 124K words (248K bytes) of UNIBUS address space are not mapped virtually by VMS.)

3. a) In going from a small number of bits (18) to a large number (28), mapping is required. The low two bits of the UNIBUS address are coded into the mask code and function code.

   UNIBUS <17:Ø9> selects a mapping register which contains 21 bits of PFN.

   UNIBUS <Ø8:Ø2> directly becomes the low seven bits of the SBI address.

   b) In a DMA transfer, the mapping registers must be allocated and loaded to map the transfer. The UNIBUS address corresponding to the first register is then loaded into the appropriate device register (byte offset into page complicates this slightly).

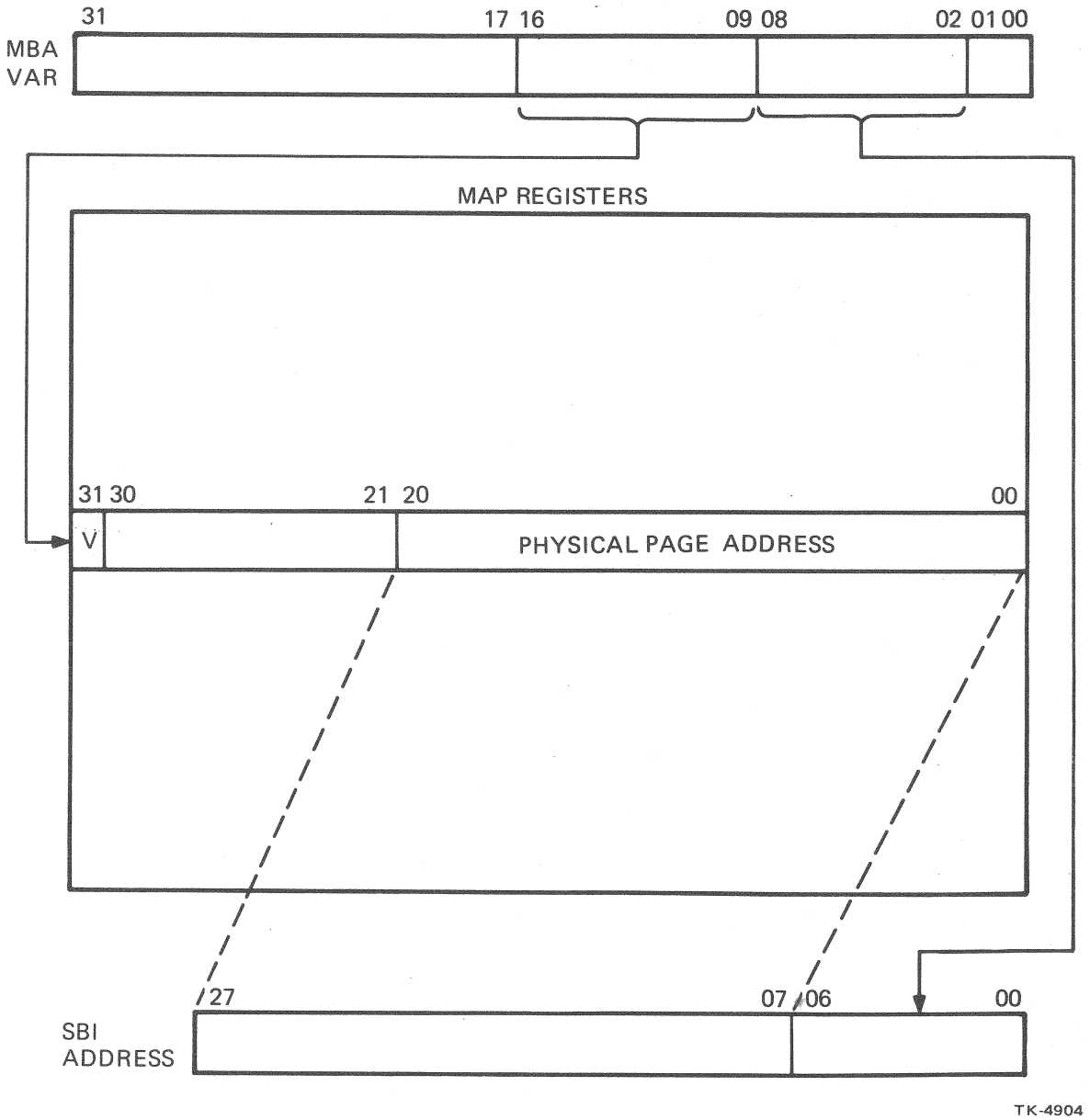
   The contents of the mapping registers are usually obtained from a series of page table entries.

I/O Architecture

ANSWER SHEET

4. a. buffered
   b. direct
   c. direct
   d. no data path concept on MBA
   e. buffered

5.



digital

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