Contents

1. Introduction
2. The IO Bus
3. Processor Initiated Activity
4. Device Initiated Activity
5. Appendix Bus Sequences
   Figure 1 - Input One Byte
   Figure 2 - Output One Byte
   Figure 3 - Input Two Bytes
   Figure 4 - Output Two Bytes
   Figure 5 - Interrupt - No Channel
   Figure 6 - Interrupt - Channel
   Figure 7 - Channel - Read 16-bit Address, Don't Use BC, BA
   Figure 8 - Channel - Count Only BC Single Byte Device (Data)
   Figure 8a - Channel - Count Only BC Double Byte Device (Data)
   Figure 9 - Channel - Output One Byte
   Figure 10 - Channel - Input One Byte
   Figure 11 - Channel - Add In One Byte
1. Introduction

The PDP-X IO Bus provides many facilities that are available to both standard and customer-designed peripheral devices. Some of these facilities are:

1. Multiplexor Channel input and output
2. Data Packing
3. Priority Interrupt synchronization
4. ability to specify arbitrary memory locations for data transfer operations
5. addition of external data to the contents of main memory

The data input/output and packing operations are described in PDP-X Technical Memorandum #29 (sections 3.1, 3.2, 3.3). This document describes additional features of the IO Bus and provides detailed information on the sequences of events on the fully interlocked bus. Details concerning the electrical and mechanical characteristics of the IO Bus are beyond the scope of this document. It is sufficient to say, however, that the IO Bus uses a twisted pair, push/pull current system. The reader is first referred to Technical Memorandum #29 in the sections mentioned above, as well as to section 3.7.
2. The IO Bus

The IO Bus is the connection between IO device control units and the central processor. All communication lines to and from the processor are common to all control units except for the priority request and grant lines which are common only to devices on a given priority level. At any one instant, however, only one control unit is logically connected to the processor (selected). This connection is maintained until the processor re-selects a different device. Since all standard devices can operate with a comparatively small amount of processor intervention, many devices may be operational simultaneously.

The IO Bus consists of 8 bi-directional data lines and 32 control lines. The control lines may be broken into four groups:

a. 8 outbound control lines (Command Out) (from the processor to the device)

b. 8 inbound control lines (Response In) (from the device to the processor)

c. 8 inbound priority request lines (from the devices to the processor, 1 line per each priority level)

d. 8 outbound priority grant lines (from the processor to the devices, 1 line per priority level)

In addition, power and ground return are available on the bus.

The priority request and grant lines are used to synchronize device service requests with processor activity. The bi-directional data lines and the outbound and inbound control lines are used to control the actual data transfers and to indicate special functions or exceptional conditions.
For purposes of further discussion, it is meaningful to distinguish between two types of operation on the bus:

1. Processor initiated activity

2. Device initiated activity

(It must be remembered that both types of activity are directly or indirectly initiated by the processor. The distinction is made here between specific tasks performed by the program controlling the processor such as reading data or changing device status and those processor functions performed without the intervention of the program because of requests generated by a device. Of course, the program had to initialize the processor and the device to generate and respond to these requests. It might further be noted that the IO Bus is, quite literally, merely a collection of wires. The functional properties ascribed to these wires arise only because there is hardware in the processor controlling the signals transmitted by these wires.)
3. Processor Initiated Activity

The processor (or to more exact, the program controlling the processor) may initiate activity on the IO Bus by the execution of an IO class instruction. These instructions are:

IOR/IOW/IORC/IOWC  Read or write data
IOS/IOC             Read or write status
RIO                  Reset IO system

One additional instruction, IOT, is provided for the convenience of the programmer. IOT appears to the IO Bus (and hence the selected device) as an IOS (read status). All of these instructions (with the exception of RIO) contain a Device Address (device selection code, device number, etc.).

When an IO class instruction is executed, this Device Address is transmitted on the Data Lines. The outbound control lines are set to indicate that an Address is on Data Lines and a SYNC level is transmitted (one of the outbound control lines). Every device on the IO Bus decodes the command lines and the Data Lines, but only one device recognizes the Device Address code. This device becomes selected (sets a Flip-Flop in itself) and signals the processor that it has received and decoded the address by setting the inbound control lines and raising RTN (one of the inbound control lines). The processor may now disconnect the Device Address from the Data Lines and drop SYNC. The selected device then drops RTN and the other inbound control lines and the bus becomes free. (Automatic time-out circuitry in the processor will disconnect the Device Address from the bus, etc. and allow the processor program to continue if no device responds.)

The processor next initiates a DATI END sequence (IOR, IORC), DATO sequence (IOW, IOWC), CONI sequence (IOS) or CONO sequence (IOC). Figure 1 depicts a one-byte data input sequence (IOR, IORC). The status input sequence (IOS, IOT) would be identical except that the processor would set the command lines to CONI instead of DATI END. Figure 2 depicts a one-byte data output sequence (IOW, IOWC). The status output sequence (IOC) would be identical except that the processor would set the command lines to CONO instead of DATO.
Figures 3 and 4 depict two-byte input and output respectively. When the first data transfer between processor and device is accomplished, the device responds and indicates that it requires (or has) an additional byte of data (or status).

When an RIO instruction is executed (or when the corresponding console switch is closed), the processor sets the outbound control lines to indicate a Reset operation and raises SYNC (one of the outbound control lines). It then waits for all devices to receive this signal (the same time-out that occurs during the address-selection sequence) and then proceeds.
4. Device-Initiated Activity

When a device finishes a task previously requested by the processor, it will set the REQ bit in its status register. The processor may test this bit (with an IOS or IOT instruction) to determine if the device has finished its task. Since the processor would have no way of knowing when the device finished unless it continually tested this bit (precluding any other computational activity), an interrupt system is made available to allow the device to signal its service request to the processor. Every device has an ENABLE bit in its status register which allows it to be connected or disconnected from this interrupt system. If this ENABLE bit is set and the device has either finished its task or has detected an unusual condition (REQ or UNUSUAL), the device will signal the central processor that it requires service by raising the REQi line corresponding to the priority level to which it is assigned (i). (Note that this does not affect any operations on the data, outbound control and inbound control lines that may be occurring.) When the processor is able to honor this service request, it raises the corresponding Priority (grant)i line. This line passes through all devices whose priority is i. The first such device that is also requesting (has REQi up), does not re-transmit this signal to the next device, but instead, connects its Device Address code and the HIGH bit from its status register onto the Data Lines and raises Address In (one of the inbound control lines). When the processor detects the rise of Address In, it reads in the Device Address and HIGH from the Data Lines. The processor then determines if a channel operation is indicated or not by examining the contents of the main memory location indicated by the Device Address and HIGH.

Figure 5 depicts the sequence that occurs when a channel operation is not indicated. Figure 6 depicts the sequence that occurs when a channel operation is indicated. In a channel operation, the device is selected. The processor does so by setting the outbound control lines to indicate an address-selecction sequence and raises SYNC. Since the Device Address is already on the Data Lines, the device becomes selected and sets the inbound control lines including RTN. The proces-
sor then examines the inbound control lines and drops Priority and SYNC. The inbound control lines 4, 5 indicate one of four initial types of operations:

0  Normal, the Byte Counter and Byte Address are updated. Response In 6-7 are then examined.

1  Inhibit BA Count. Only the Byte Counter is updated. The Byte Address is left undisturbed. Response In 6-7 are then examined.

2  Force Last Cycle After Reading in 16-bit Byte Address. The IO Bus performs 2 DATI cycles to read in a 16-bit Byte Address. The Byte Counter and Byte Address are undisturbed. Response In 6-7 are then examined.

3  Permit only BC Count. The Byte Counter is incremented and the result is re-transmitted to the device. The interrupt is then dismissed by the processor.

Modes 2 and 3 are depicted in Figures 7 and 8 respectively. Figure 8a depicts Mode 3 for a 2-byte device. After Modes 0, 1, 2, Response In 6-7 are examined to determine the type of data transfer as follows:

0  OUT
1  ILLEGAL
2  IN
3  ADD-IN

These types of transfer are depicted for a single-byte device in Figures 9, 10, 11. If the device is a multiple-byte device, channel overflow is indicated while transferring the byte in which the overflow occurred. In the case of ADD-IN, which reads in data bytes, does the addition, stores the sum in memory and transmits the sum; two types of overflow may occur:

a. channel overflow
b. add overflow

Channel overflow is indicated while transforming the byte in which the overflow occurred (DATI LAST) and add overflow is indicated during each
each output transfer. Only one or two byte transfers will be made during an ADD-IN operation. Additional bytes will not be received. Double-byte additions across word boundaries may produce unpredictable sum and overflow results.
5. Appendix

Bus Sequences
IOR, IORC INSTRUCTION (IOS, IOT with DATI
END changed to CONI)

Processor

set up Device Address
onto Data Lines 2-7;
set command 5-7 to ADDRESS;

raise SYNC

drop SYNC

set command 5-7 = DATI END;

Device

declare address on Data Lines
2-7;

decode Command 5-7 = ADDRESS;
select;
raise RTN

drop RTN

decode command 5-7 = DATI END;
load data onto Data Lines;
raise RTN

drop RTN

read in DATA
(processor may proceed)

drop SYNC

disconnect data from Data Lines;

drop RTN

Bus is free
IOW, IOWC INSTRUCTION (IOC with DATC changed to CONO)

**Processor**

- set up Device Address onto Data Lines 2-7; set Command 5-7 to ADDRESS;
- raise SYNC
- drop SYNC
- put data byte on Data Lines; set command 5-7 to DATC
- raise SYNC
- (processor may proceed) disconnect data from Data Lines;
- drop SYNC
- Bus is free

**Device**

- decode address on Data Lines 2-7; decode command 5-7 = ADDRESS; select;
- raise RTN
- decode command 5-7 = DATC; read in data;
- raise RTN
- drop RTN

**Figure 2**

**Bus Sequence**
Output One Byte
IOR, IORC INSTRUCTION (IOS, IOT with DATI END changed to CONI)

Processor
- set up Device Address onto Data Lines 2-7;
- set Command 5-7 to ADDRESS;
- raise SYNC
- drop SYNC
- set Command 5-7 = DATI END
- raise SYNC
- drop SYNC
- Read in First Byte;
- drop SYNC
- set Command 5-7 = DATI END;
- raise SYNC
- drop SYNC
- Read in Second Byte;
- (processor may proceed)
- Bus is free
- drop RTN

Bus Sequence
Input Two Bytes

Device
- decode address on Data Lines 2-7;
- decode Command 5-7 = ADDRESS;
- select;
- raise RTN
- drop RTN
- decode Command 5-7 = DATI END;
- load data onto Data Lines;
- raise More Bytes (RI 3);
- raise RTN
- drop RTN
- disconnect data from Data Lines;
- disconnect Response In;
- drop RTN
- decode Command 5-7 = DATI END;
- load data byte onto Data Lines;
- raise RTN
- disconnect data from Data Lines;
- drop RTN
IOW, IOWC INSTRUCTION (IOC with DATO changed to CONO)

Processor

1. set up Device Address onto Data Lines 2-7;
2. set Command 5-7 to ADDRESS;
3. raise SYNC
4. drop SYNC
5. put data onto Data Lines;
6. set Command 5-7 to DATO;
7. raise SYNC
8. disconnect data from Lines;
9. drop SYNC
10. put second byte onto Data Lines;
11. set Command 5-7 to DATO;
12. raise SYNC
13. disconnect data from Lines; (processor may proceed)
14. drop SYNC

Bus is free

Figure 4

Bus Sequence
Output Two Bytes

Device

decode address on Data Line; decode Command 5-7 = ADDRESS;
drop RTN

raise RTN

decode Command 5-7 = DATO;
Read in data;
Raise More Bytes (RI3);
raise RTN

disconnect Response In;
drop RTN

decode Command 5-7 = DATO;
Read in Data;
raise RTN

drop RTN
Bus Sequence
Interrupt - No Channel

Figure 5

Processor

if current priority < i (and not inhibited)
send Priority

read address from Data Lines; decode if Channel: if not; (processor may proceed)

drop Priority

Bus is free; processor doesn't respond since it is now operating at priority

Device

Send REQ

device is requesting and Priority
is up, put High and Address on Data Lines

1-7;
set Response In to indicate type of operation;
raise Address In

disconnect address, Response In;
drop Address In
Bus Sequence
Interrupt - Channel

Figure 6
Processor

if current priority < i (and not inhibited)
Send Priority_1

(read Address from Data Lines; decide if Channel
if it is;
Set Command 5-7 = ADDRESS
raise SYNC

(processor may proceed to
perform channel function
(figure 7, 8 or directly to 9, 10 or 11)
 drop Priority_i
 drop SYNC

Device

(REQ, UNUSUAL) \_ ENABLE

Send REQ_i

device is requesting and Priority_1 is up,
put HIGH and Address on Data Lines 1-7;
set Response In to indicate type of opera-
tion;

raise Address In

decode address on Data Lines 2-7;
decode Command 5-7 = ADDRESS;
select;

raise RTN

drop connect HIGH, Address;
drop Response In;
drop Address In

Bus is free
Enter from Interrupt Sequence - Channel (Figure 6)

Processor
Set Command 5-7 to DATI;
raise SYNC
Read in Byte 1;
drop SYNC
Set Command 5-7 to DATI END;
raise SYNC
Read in Byte 2;
drop SYNC
Bus is free
Proceed to
CHANNEL INPUT (Figure 9)
CHANNEL OUTPUT (Figure 10)
CHANNEL ADD-IN (Figure 11)

Bus Sequence
Channel - Read 16-bit Address
Don't Use BC,BA
Figure 7
Device

decode Command 5-7 = DATI;
connect byte 1 to Data Lines;
raise RTN
drop RTN
drop RTN;
raise Address In
decode Command 5-7 = DATI;
connect byte 2 to Data Lines;
connect Response In
raise RTN;
Processor
Set Command 5-7 to
a. DATO if BC ≠ 0
b. DATO LAST if BC = 0
Put BC byte 1 on Data Lines
raise SYNC 
disconnect data from Lines
(processor may proceed; dismis)
drop SYNC
Bus is free
Enter from Interrupt Sequence - Channel (Figure 6)

Processor
Put BC byte 1 onto Data Lines;
Set Command 5-7 to
a. DATO if BC ≠ 0
b. DATO LAST if BC = 0

raise SYNC

disconnect data from Lines
drop SYNC

Put BC byte 2 onto Data Lines
Set Command 5-7 to
a. DATO if BC ≠ 0
b. DATO LAST if BC = 0

decode Command 5-7 = DATO;
read data
raise More Bytes (RI3)
raise RTN

drop Response In
drop SYNC

drop RTN

Bus is free
Processor

Put data byte on Data Lines;
Set Command 5-7 to:
  a. DATO if no OVERFLOW
  b. DATO LAST if OVERFLOW
raise SYNC

(processor may proceed;
  dismiss interrupt)

dropsync

drop RTN

Bus is free

Bus Sequence
Channel - Output One Byte

Figure 9

Device

decode Command 5-7 =
  a. DATO - read data;
     start device
  b. DATO LAST - read
     data; start device;
     clear HIGH; clear
     REQ; drop REQ1*
raise RTN

drop RTN

*Device can signal more bytes
by raising More Bytes (Response In 3). This sequence
is the same as the second se-
quence in Figure 2 (i.e., af-
ter the device Selection Se-
quence).
Enter from Interrupt Sequence - Channel (Figure 6) or Read 16-bit Address (Figure 7)

Processor

Set Command 5-7 to
a. DATI if no OVERFLOW
b. DATI LAST if OVERFLOW

Raise SYNC
Read in Data;
(Processor may proceed, dismis)

Drop SYNC
Bus is free

Bus Sequence
Channel - Input One Byte

Device

Decode Command 5-7 =
a. DATI clear REQ, set BUSY, start device
b. DATI LAST clear HIGH
connect data to Data Lines;

Raise RTN

Drop RTN

*Device can signal more bytes by raising More Bytes (Response In 3). This sequence is
the same as the second sequence in Figure 7 (i.e., after the device selection sequence).
Enter from Interrupt Sequence - Channel (Figure 6) or Read 16-bit Address (Figure 7)

Processor

Set Command 5-7 to
a. DATI if no OVERFLOW
b. DATI LAST if OVERFLOW

raise SYNC

read data from Data Lines Form SUM
drop SYNC

Set Command 5-7 to
a. DATO if no SUM OVERFLOW
b. DATO LAST if SUM OVERFLOW

put sum on Data Lines

raise SYNC

drop RTN

disconnect data from Lines (processor may proceed; dismiss)
drop SYNC

decode Command 5-7 =
  a. DATI clear REQ, set BUSY
  b. DATI LAST clear HIGH
connect data to Data Lines
raise RTN
drop RTN

decode Command 5-7 read sum from Data Lines
raise RTN

drop RTN