Module
3

Embedded Systems I/O

Lesson
13

Interfacing bus, Protocols, ISA bus etc.

Instructional Objectives

After going through this lesson the student would learn

- Bus, Wires and Ports
- Basic Protocols of data transfer
- Bus arbitration
- ISA bus signals and handshaking
- Memory mapped I/O and simple I/O
- Parallel I/O and Port Based I/O
- Example of interfacing memory to the ports of 8051

Pre-Requisite

Digital Electronics, Microprocessors

13.1 Introduction

The traditional definition of input-output is the devices those create a medium of interaction with the human users. They fall into the following categories such as:
1. Printers
2. Visual Display Units
3. Keyboard
4. Cameras
5. Plotters
6. Scanners

However in Real-Time embedded systems the definition of I/O devices is very different. An embedded controller needs to communicate with a wide range of devices namely
1. Analog to Digital (A-D) and Digital to Analog (D-A) Converters
2. CODECs
3. Small Screen Displays such as TFT, LCD etc
4. Antennas
5. Cameras
6. Microphones
7. Touch Screens

Etc.

A typical Embedded system is a Digital Camera as shown in Fig. 13.1. As it can be seen it possesses broad range of input-output devices such as Lens, Microphone, speakers, Serial interface standards, TFT screens etc.
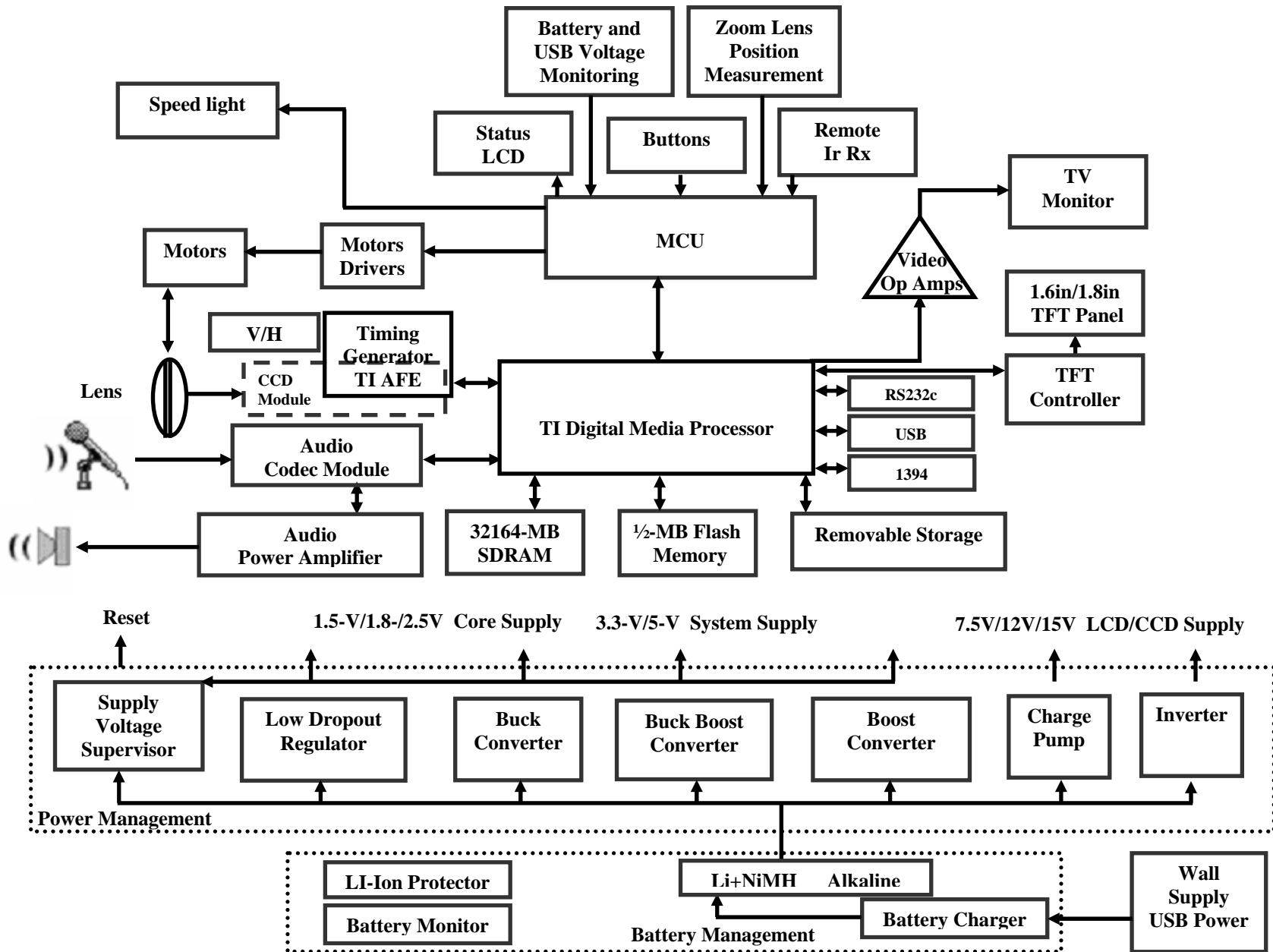
Fig. 13.1

The functionality of an Embedded System can be broadly classified as

Processing
  • Transformation of data
  • Implemented using processors
Storage
  • Retention of data
  • Implemented using memory
And **Communication (also called *Interfacing*)**
  • Transfer of data between processors and memories
  • Implemented using buses

Interfacing

Interfacing is a way to communicate and transfer information in either way without ending into deadlocks. In our context it is a way of effective communication in real time.
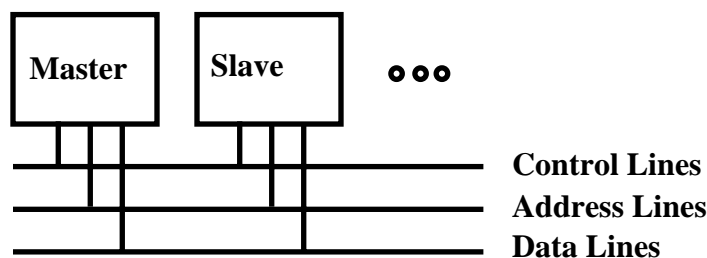This involves
  – Addressing
  – Arbitration
  – Protocols

Fig. 13.2(a) The Bus structure

Addressing: The data sent by the master over a specified set of lines which enables just the device for which it is meant
Protocols: The literal meaning of protocol is a set of rules. Here it is a set of formal rules describing how to transfer data, especially between two devices.
A simple example is memory read and write protocol. The set of rules or the protocol is
For read (Fig. 13.2 (b))
         The CPU must send the memory address
         The read line must be enabled
         The processor must wait till the memory is ready
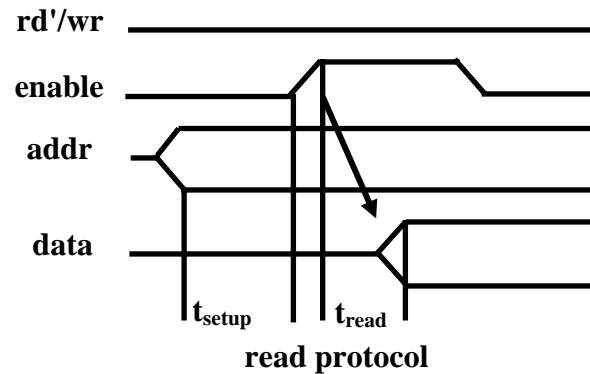   Then accept the bits in the data lines

rd'/wr

enable

addr

data

t_{setup} t_{read}

read protocol

Fig. 13.2(b)

For write (Fig. 13.2(c))

        The CPU must send the memory address

        The write line must be enabled

        The processor sends the data over the data lines
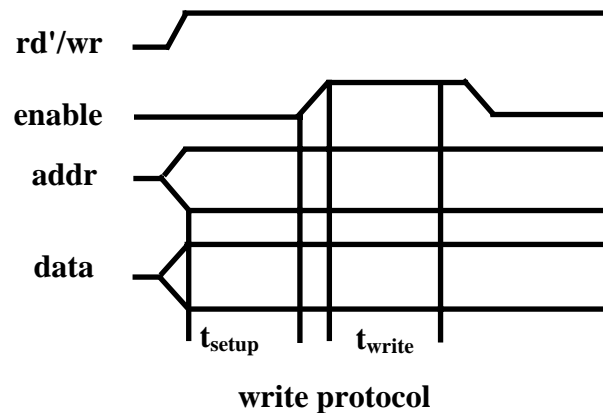
        The processor must wait till the memory is ready

rd'/wr

enable

addr

data

t_{setup} t_{write}

write protocol

Fig. 13.2(c)

Arbitration: When the same set of address/data/control lines are shared by different units then the bus arbitration logic comes into play. Access to a bus is arbitrated by a *bus master*. Each node on a bus has a bus master which requests access to the bus, called a *bus request*, when then node requires to use the bus. This is a global request sent to all nodes on the bus. The node that currently has access to the bus responds with either a *bus grant* or a *bus busy* signal, which is also globally known to all bus masters. (Fig. 13.3)
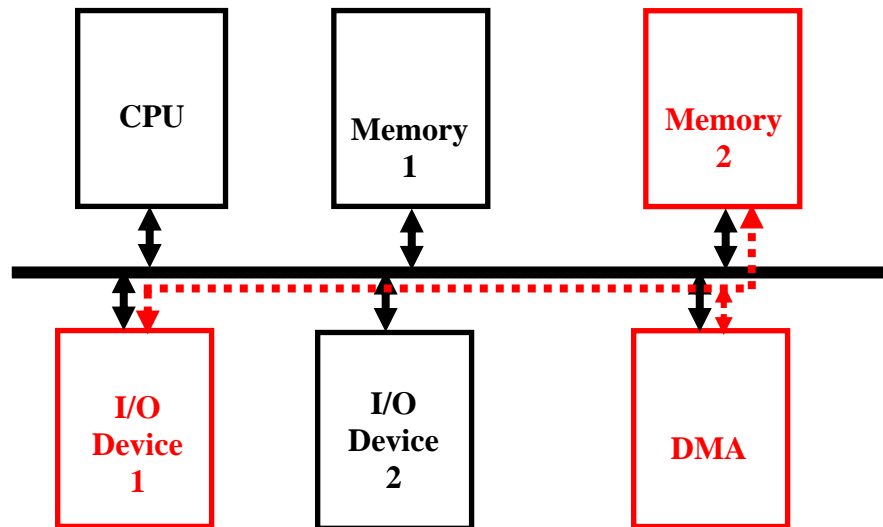
Fig. 13.3 The bus arbitration of the DMA, known as direct memory access controller which is responsible for transferring data between an I/O device and memory without involving the CPU. It starts with a bus request to the CPU and after it is granted it takes over the address/data and control bus to initiate the data transfer. After the data transfer is complete it passes the control over to the CPU.

Before learning more details about each of these concepts a concrete definition of the following terms is necessary.

Wire: It is just a passive physical connection with least resistance

Bus: A group of signals (such as data, address etc). It may be augmented with buffers latches etc. A bus has standard specification such as number of bits, the clock speed etc.

Port: It is the set of physical wires available so that any device which meets the specified standard can be directly plugged in. Example is the serial, parallel and USB port of the PC.

Time multiplexing: This is to Share a single set of wires for multiple pieces of data. It saves wires at expense of time

Time-multiplexed data transfer
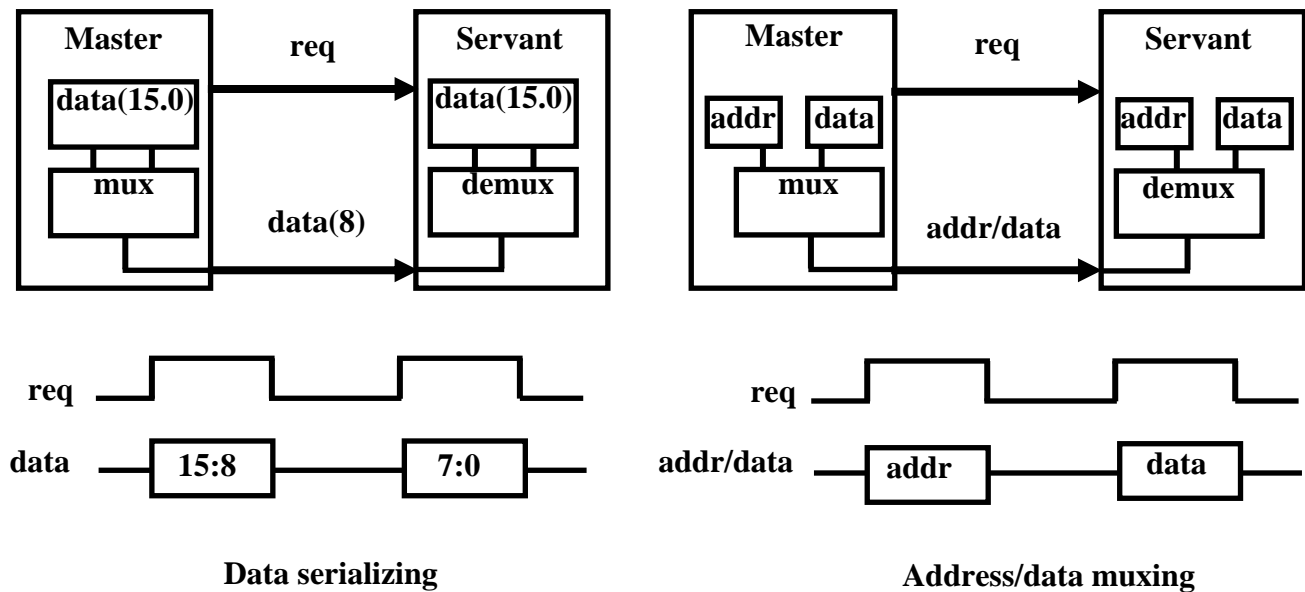

Data serializing

Address/data muxing

Fig. 13.4 The Time multiplexing data transfer. The left hand side transmits 16-bits of data in an 8-bit line MSB after the LSB. The transfer is synchronized with the *req* signal. In the example shown on the right hand side the same set of wires carry address followed by data in synchronism with the *req* signal. mux: stands for multiplexer

The Handshaking Protocol

Strobe Protocol

Fig. 13.5(a) Strobe Protocol

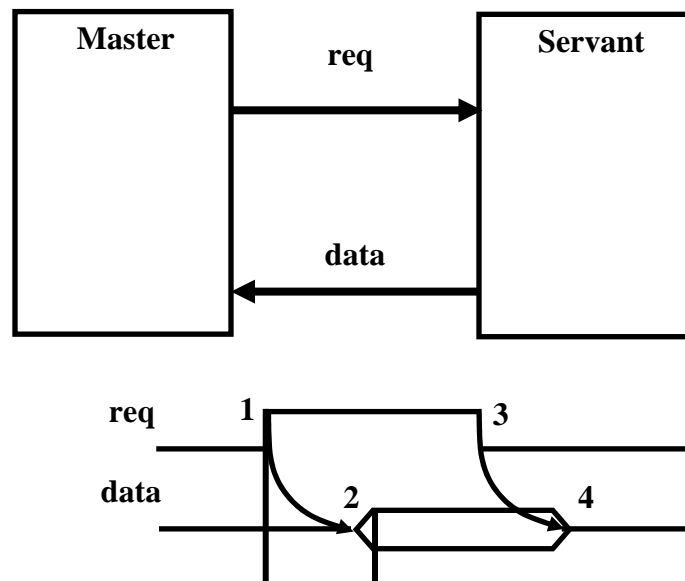1. Master asserts *req* to receive data
2. Servant puts data on bus within time t_{access}
3. Master receives data and deasserts *req*
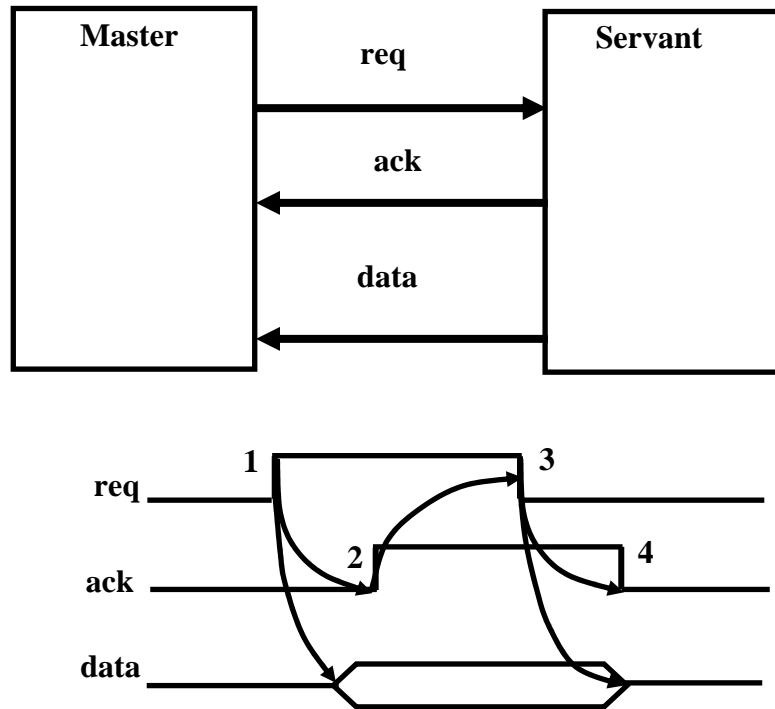4. Servant ready for next request

Handshake Protocol

Fig. 13.5(b) Handshake Protocol

1. Master asserts *req* to receive data
2. Servant puts data on bus and asserts *ack*
3. Master receives data and deasserts *req*
4. Servant ready for next request
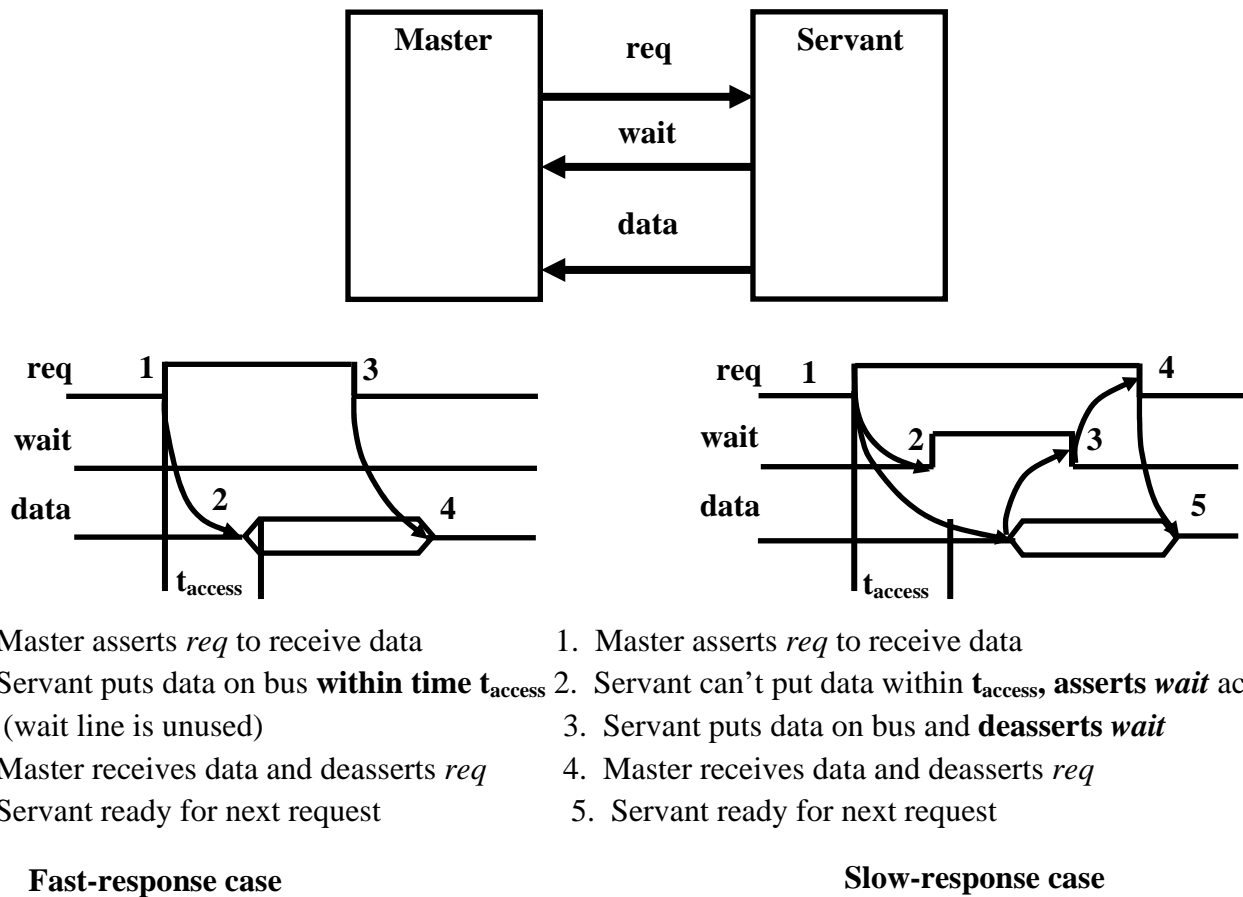
The Strobe & Handshake combined

Fast-response case

1. Master asserts *req* to receive data
2. Servant puts data on bus **within time t_{access}** (wait line is unused)
3. Master receives data and deasserts *req*
4. Servant ready for next request

Slow-response case

1. Master asserts *req* to receive data
2. Servant can't put data within t_{access}**, asserts *wait* ack
3. Servant puts data on bus and **deasserts *wait***
4. Master receives data and deasserts *req*
5. Servant ready for next request

Fig. 13.5(c) Strobe and Handshake Combined

Handshaking Example in ISA Bus

The Industry Standard Architecture (ISA Bus) has been described as below
This is a standard bus architecture developed to help the various designers to customize their product and the interfaces. The pin configuration and the signals are discussed below.
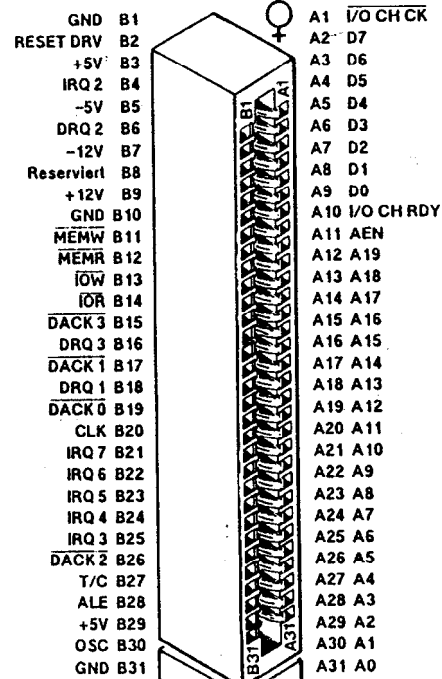
ISA

GND	B1	A1	I/O CH CK
RESET DRV	B2	A2	D7
+5V	B3	A3	D6
IRQ 2	B4	A4	D5
–5V	B5	A5	D4
DRQ 2	B6	A6	D3
–12V	B7	A7	D2
Reserviert	B8	A8	D1
+12V	B9	A9	D0
GND	B10	A10	I/O CH RDY
MEMW	B11	A11	AEN
MEMR	B12	A12	A19
IOW	B13	A13	A18
IOR	B14	A14	A17
DACK 3	B15	A15	A16
DRQ 3	B16	A16	A15
DACK 1	B17	A17	A14
DRQ 1	B18	A18	A13
DACK 0	B19	A19	A12
CLK	B20	A20	A11
IRQ 7	B21	A21	A10
IRQ 6	B22	A22	A9
IRQ 5	B23	A23	A8
IRQ 4	B24	A24	A7
IRQ 3	B25	A25	A6
DACK 2	B26	A26	A5
T/C	B27	A27	A4
ALE	B28	A28	A3
+5V	B29	A29	A2
OSC	B30	A30	A1
GND	B31	A31	A0

Fig. 13.6 The ISA bus

ISA Signal Descriptions

SA19 to SA0 (*SA for System Address*)

System Address bits 19:0 are used to address memory and I/O devices within the system. These signals may be used along with LA23 to LA17 to address up to 16 megabytes of memory. Only the lower 16 bits are used during I/O operations to address up to 64K I/O locations. SA19 is the most significant bit. SA0 is the least significant bit. These signals are gated on the system bus when BALE is high and are latched on the falling edge of BALE. They remain valid throughout a read or write command. These signals are normally driven by the system microprocessor or DMA controller, but may also be driven by a bus master on an ISA board that takes ownership of the bus.

LA23 to LA17

Unlatched Address bits 23:17 are used to address memory within the system. They are used along with SA19 to SA0 to address up to 16 megabytes of memory. These signals are valid when BALE is high. They are "unlatched" and do not stay valid for the entire bus cycle. Decodes of these signals should be latched on the falling edge of BALE.

AEN

Address Enable is used to degate the system microprocessor and other devices from the bus during DMA transfers. When this signal is active the system DMA controller has control of the

address, data, and read/write signals. This signal should be included as part of ISA board select decodes to prevent incorrect board selects during DMA cycles.

BALE

Buffered Address Latch Enable is used to latch the LA23 to LA17 signals or decodes of these signals. Addresses are latched on the falling edge of BALE. It is forced high during DMA cycles. When used with AEN, it indicates a valid microprocessor or DMA address.

CLK

System Clock is a free running clock typically in the 8MHz to 10MHz range, although its exact frequency is not guaranteed. It is used in some ISA board applications to allow synchronization with the system microprocessor.

SD15 to SD0

System Data serves as the data bus bits for devices on the ISA bus. SD15 is the most significant bit. SD0 is the least significant bits. SD7 to SD0 are used for transfer of data with 8-bit devices. SD15 to SD0 are used for transfer of data with 16-bit devices. 16-bit devices transferring data with 8-bit devices shall convert the transfer into two 8-bit cycles using SD7 to SD0.

DACK0 to DACK3 and DACK5 to DACK7

DMA Acknowledge 0 to 3 and 5 to 7 are used to acknowledge DMA requests on DRQ0 to DRQ3 and DRQ5 to DRQ7.

DRQ0 to DRQ3 and DRQ5 to DRQ7

DMA Requests are used by ISA boards to request service from the system DMA controller or to request ownership of the bus as a bus master device. These signals may be asserted asynchronously. The requesting device must hold the request signal active until the system board asserts the corresponding DACK signal.

I/O CH CK

I/O Channel Check signal may be activated by ISA boards to request than an non-maskable interrupt (NMI) be generated to the system microprocessor. It is driven active to indicate a uncorrectable error has been detected.

I/O CH RDY

I/O Channel Ready allow slower ISA boards to lengthen I/O or memory cycles by inserting wait states. This signals normal state is active high (ready). ISA boards drive the signal inactive low (not ready) to insert wait states. Devices using this signal to insert wait states should drive it low

immediately after detecting a valid address decode and an active read or write command. The signal is release high when the device is ready to complete the cycle.

IOR

I/O Read is driven by the owner of the bus and instructs the selected I/O device to drive read data onto the data bus.

IOW

I/O Write is driven by the owner of the bus and instructs the selected I/O device to capture the write data on the data bus.

IRQ3 to IRQ7 and IRQ9 to IRQ12 and IRQ14 to IRQ15

Interrupt Requests are used to signal the system microprocessor that an ISA board requires attention. An interrupt request is generated when an IRQ line is raised from low to high. The line must be held high until the microprocessor acknowledges the request through its interrupt service routine. These signals are prioritized with IRQ9 to IRQ12 and IRQ14 to IRQ15 having the highest priority (IRQ9 is the highest) and IRQ3 to IRQ 7 have the lowest priority (IRQ7 is the lowest).

SMEMR

System Memory Read instructs a selected memory device to drive data onto the data bus. It is active only when the memory decode is within the low 1 megabyte of memory space. SMEMR is derived from MEMR and a decode of the low 1 megabyte of memory.

SMEMW

System Memory Write instructs a selected memory device to store the data currently on the data bus. It is active only when the memory decode is within the low 1 megabyte of memory space. SMEMW is derived from MEMW and a decode of the low 1 megabyte of memory.

MEMR

Memory Read instructs a selected memory device to drive data onto the data bus. It is active on all memory read cycles.

MEMW

Memory Write instructs a selected memory device to store the data currently on the data bus. It is active on all memory write cycles.

REFRESH

Memory Refresh is driven low to indicate a memory refresh operation is in progress.

OSC

Oscillator is a clock with a 70ns period (14.31818 MHz). This signal is not synchronous with the system clock (CLK).

RESET DRV

Reset Drive is driven high to reset or initialize system logic upon power up or subsequent system reset.

TC

Terminal Count provides a pulse to signal a terminal count has been reached on a DMA channel operation.

MASTER

Master is used by an ISA board along with a DRQ line to gain ownership of the ISA bus. Upon receiving a -DACK a device can pull -MASTER low which will allow it to control the system address, data, and control lines. After MASTER is low, the device should wait one CLK period before driving the address and data lines, and two clock periods before issuing a read or write command.

MEM CS16

Memory Chip Select 16 is driven low by a memory slave device to indicate it is capable of performing a 16-bit memory data transfer. This signal is driven from a decode of the LA23 to LA17 address lines.

I/O CS16

I/O Chip Select 16 is driven low by a I/O slave device to indicate it is capable of performing a 16-bit I/O data transfer. This signal is driven from a decode of the SA15 to SA0 address lines.

0WS

Zero Wait State is driven low by a bus slave device to indicate it is capable of performing a bus cycle without inserting any additional wait states. To perform a 16-bit memory cycle without wait states, -0WS is derived from an address decode.

SBHE

System Byte High Enable is driven low to indicate a transfer of data on the high half of the data bus (D15 to D8).
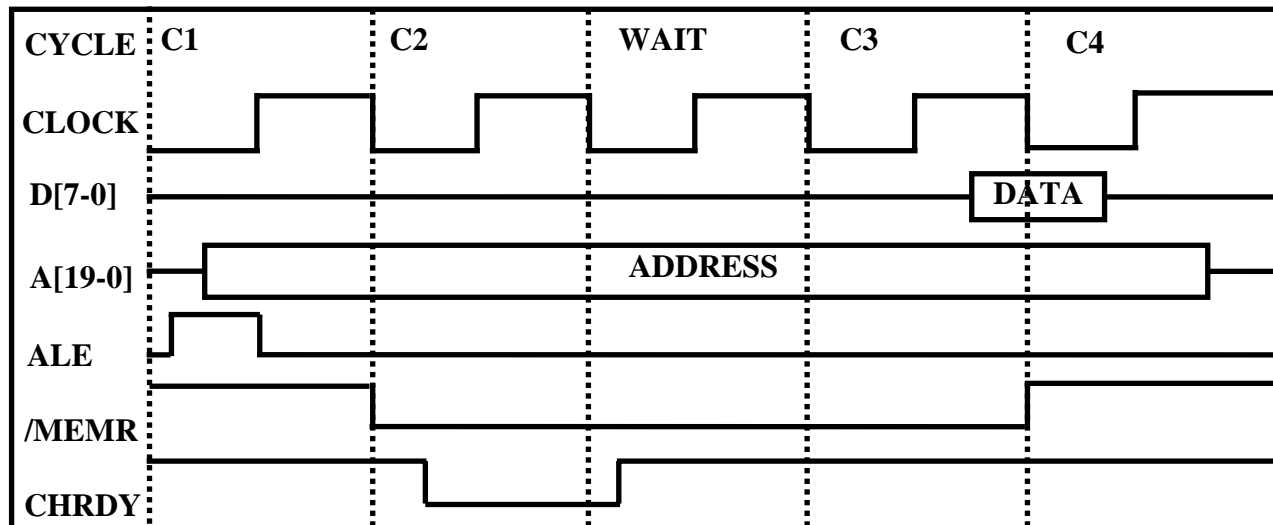
The Memory Read bus cycle in ISA bus

Fig. 13.7(a) The Handshaking Mode of Data Transfer in ISA bus
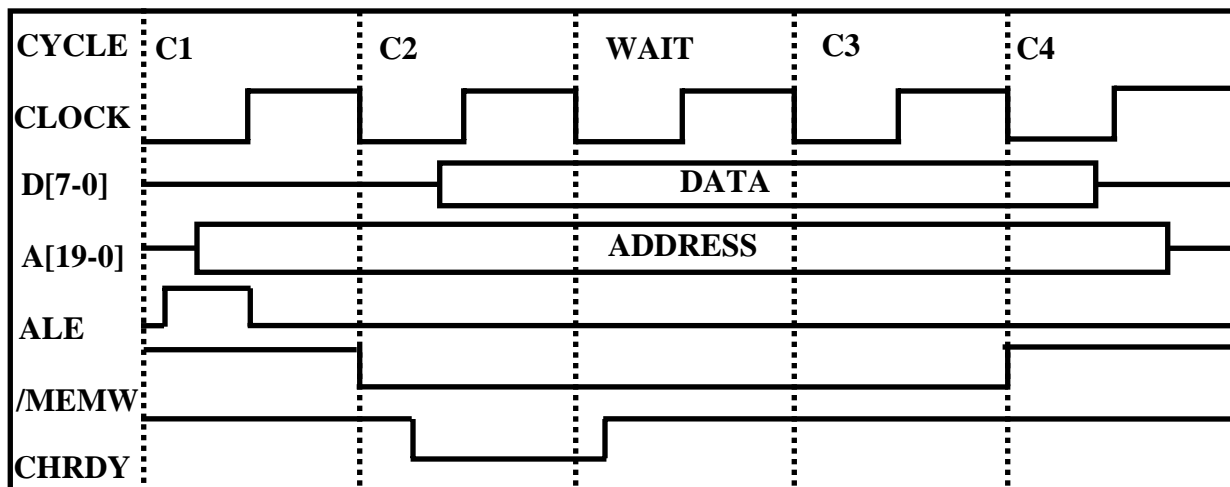
The Memory Write bus cycle in ISA bus

Fig. 13.7(b) The Handshaking Mode of Data Transfer in ISA bus

13.2 I/O addressing

- A microprocessor communicates with other devices using some of its pins. Broadly we can classify them as

- Port-based I/O (parallel I/O)
  - Processor has one or more N-bit ports
  - Processor's software reads and writes a port just like a register
- Bus-based I/O
  - Processor has address, data and control ports that form a single bus
  - Communication protocol is built into the processor
  - A single instruction carries out the read or write protocol on the bus
- Parallel I/O peripheral
  - When processor only supports bus-based I/O but parallel I/O needed
  - Each port on peripheral connected to a register within peripheral that is read/written by the processor
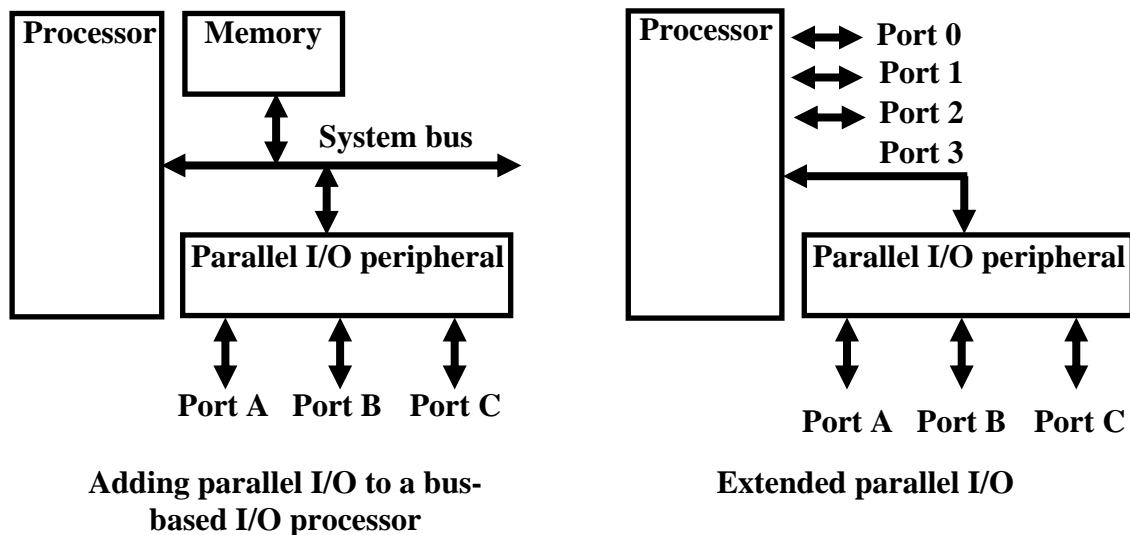

Adding parallel I/O to a bus-based I/O processor

Extended parallel I/O

Fig. 13.8 Parallel I/O and extended Parallel I/O

- Extended parallel I/O
  - When processor supports port-based I/O but more ports needed
  - One or more processor ports interface with parallel I/O peripheral extending total number of ports available for I/O
  - e.g., extending 4 ports to 6 ports in figure

Types of bus-based I/O:
Memory-mapped I/O and standard I/O
- Processor talks to both memory and peripherals using same bus – two ways to talk to peripherals
  - *Memory-mapped I/O*
    - Peripheral registers occupy addresses in same address space as memory
    - e.g., Bus has 16-bit address
      - lower 32K addresses may correspond to memory
      - upper 32k addresses may correspond to peripherals
  - *Standard I/O (I/O-mapped I/O)*
    - Additional pin (*M/IO*) on bus indicates whether a memory or peripheral access
    - e.g., Bus has 16-bit address
      - all 64K addresses correspond to memory when *M/IO* set to 0

– all 64K addresses correspond to peripherals when *M/IO* set to 1

Memory-mapped I/O vs. Standard I/O
- Memory-mapped I/O
  - Requires no special instructions
    - Assembly instructions involving memory like MOV and ADD work with peripherals as well
    - Standard I/O requires special instructions (e.g., IN, OUT) to move data between peripheral registers and memory
- Standard I/O
  - No loss of memory addresses to peripherals
  - Simpler address decoding logic in peripherals possible
    - When number of peripherals much smaller than address space then high-order address bits can be ignored
      - smaller and/or faster comparators

A basic memory protocol

Interfacing an 8051 to external memory

8051 has three 8-bit ports through which it can communicate with the outside world.
- Ports P0 and P2 support port-based I/O when 8051 internal memory being used
- Those ports serve as data/address buses when external memory is being used
- 16-bit address and 8-bit data are time multiplexed; low 8-bits of address must therefore be latched with aid of ALE (address latch enable) signal

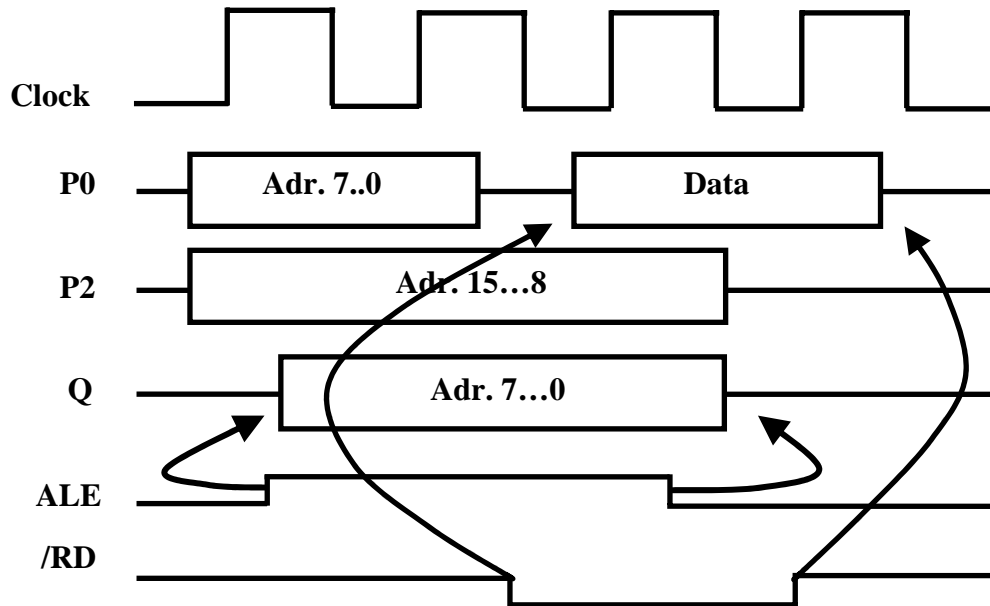
Fig. 13.9(a) A basic memory interface

Fig. 13.9(b) The timing diagram

The timing of the various signals is shown in Fig. 13.9(b). The lower byte of the address is placed along P0 and the address latch enable signal is enabled. The higher byte of the address is placed along P2. The ALE signal enables the 74373 chip to latch the address as the P0 bus will be used for data. The P0 bus goes into tri-state (high impedance state) and switches internally for data path. The $\overline{RD}$ (read) line is enabled. The bar over the read line indicates that it is active when low. The data is received from the memory on the P0 bus. A memory write cycle can be explained similarly.

13.3 Conclusion

In this lesson you learnt about the basics of Input Output interfacing. In the previous chapter you also studied about some input output concepts. But most of those I/O such as Timer, Watch Dog circuits, PWM generator, Serial and Parallel ports were part of the microcontroller. In this lesson the basics of interfacing with external devices have been discussed. The difference between a Bus and a Port should be kept in mind. The ISA bus is discussed to give an idea about the various bus architectures which will discussed in the later part of this course. You must browse various websites as listed below for further knowledge.
http://esd.cs.ucr.edu/slide_index.html
http://esd.cs.ucr.edu/wres.html
www.techfest.com/hardware/bus/isa.htm
You should be able to be in a position to learn any microcontroller and their interfacing protocols.

13.4 Questions

1. List at least 4 differences between the I/O devices for a Real Time Embedded System (RTES) and a Desktop PC?

RTES I/O	PC I/O
It has to operate in real time. The timing requirement has to met.	May take little longer and need not satisfy the stringent timing requirement of the user
The I/O devices need not be meant for the human user and may consists of analog interfaces, digital controllers, mixed signal circuits.	The I/O for desktop encompasses a broad range. Generally the keypad, monitor, mouse etc which are meant for the human users are termed as I/O. But it could have also the similar I/Os as in case of RTES
The power consumption of these I/O devices should be limited.	There is virtually no strict limit to the power in such I/Os
The size of the I/O devices should be small to make it coexist with the processor and other devices	Generally the size is not a problem as it is not meant to be portable

2. Draw the timing diagram of a memory read protocol for slower memory. What additional handshaking signals are necessary?

Ans: An additional handshaking signal from the memory namely /ready is necessary. The microcontroller inserts wait states as long as the /ready line is not inactive. The ready line in this case is sampled at the rising edge of the third clock phase. Fig.Q2 reveals the timing of such an operation.
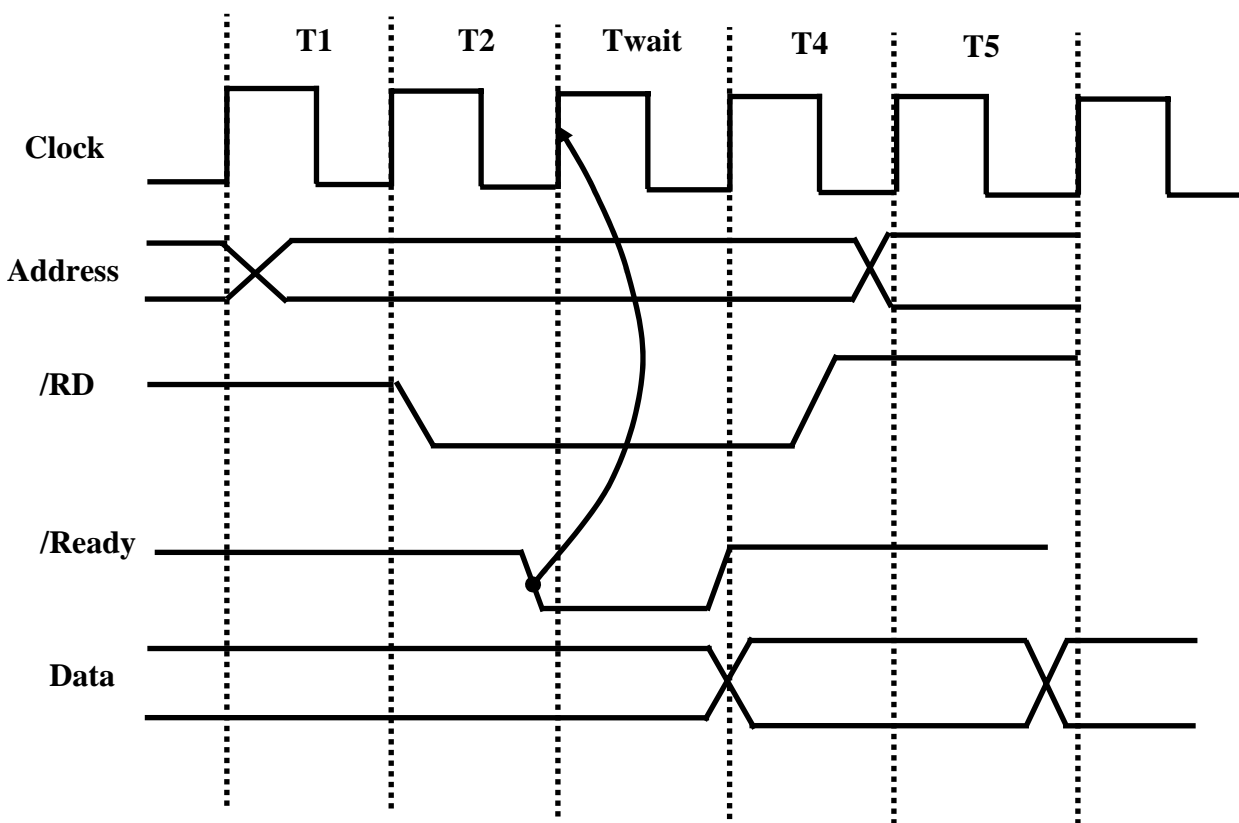

Fig. Q2 The Timing Diagram of memory read from a slower

3. Enlist the handshaking signals in the ISA bus for dealing with slower I/O devices.

Ans:

I/O CH RDY

I/O Channel Ready allow slower ISA boards to lengthen I/O or memory cycles by inserting wait states. This signals normal state is active high (ready). ISA boards drive the signal inactive low (not ready) to insert wait states. Devices using this signal to insert wait states should drive it low immediately after detecting a valid address decode and an active read or write command. The signal is release high when the device is ready to complete the cycle.

4. What additional handshaking signals are necessary for bidirectional data transfer over the same set data lines.

Ans:

For an 8-bit data transfer we need at least 4 additional lines for hand shaking. As shown in Fig.Q4 there are two ports shown. Port A acts as the 8-bit bidirectional data bus. Port C carries the handshaking signals.

Write operation: When the data is ready the /OBFA (PC7 output buffer full acknowledge active low) signal is made 0. The device which is connected acknowledges through /ACKA(PC6 acknowledge that it is ready to accept data. It is active low). The data transfer takes place over PA0-PA7.

Read operation: When the data is ready the external device makes the /STBA (PC4 Strobe acknowledge active low) line low. The acknowledgement is sent through IBFA (Input Buffer Empty Acknowledge that it is ready to accept data. It is active high). The data transfer takes place.
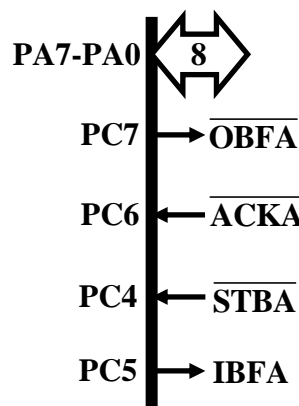

Fig. Q4 The master

5. List the various bus standards used in industry.

Ans:

ISA Bus

The Industry Standard Architecture (ISA) bus is an open, 8-bit (PC and XT) or 16-bit (AT) asymmetrical I/O channel with numerous compatible hardware implementations.

EISA Bus

The Extended Industry Standard Architecture (EISA) bus is an open, 32-bit, asymmetrical I/O channel with numerous compatible hardware implementations. The system bus and allows data transfer rates at a bandwidth of up to 33 MB per second, supports a 4 GB address space, 8 DMA channels, and is backward compatible with the Industry Standard Architecture (ISA) bus.

PCI Bus

The Peripheral Component Interconnect Local Bus (PCI) is an open, high-performance 32-bit or 64-bit synchronous bus with multiplexed address and data lines, and numerous compatible hardware implementations. PCI bus support a PCI frequency of 33 MHz and a transfer rate of 132 MB per second.

Futurebus+

Futurebus+ is an open bus, designed by the IEEE 896 committee, whose architecture and interfaces are publicly documented, and that is independent of any underlying architecture. It has broad-base, cross-industry support; very high throughput (the maximum rate for 64-bit bandwidth is 160 MB per second; for the 128-bit bandwidth, 180 MB per second). Futurebus+ supports a 64-bit address space and a set of control and status registers (CSRs) that provides all the necessary ability to enable or disable features; thus supporting multivendor interoperablity.

SCSI Bus

The Small Computer Systems Interface (SCSI) bus is an ANSI standard for the interconnection of computers with each other and with disks, floppies, tapes, printers, optical disks, and scanners. The SCSI standard includes all the mechanical, electrical, and
Data transfer rates are individually negotiated with each device attached to a given SCSI bus. For example, a 4 MB per second device and a 10 MB per second device may share a fast narrow bus. When the 4 MB per second device is using the bus, the transfer rate is 4 MB per second. When the 10 MB per second device is using the bus, the transfer rate is 10 MB per second. However, when faster devices are placed on a slower bus, their transfer rate is reduced to allow for proper operation in that slower environment.
Note that the speed of the SCSI bus is a function of cable length, with slow, single-ended SCSI buses supporting a maximum cable length of 6 meters, and fast, single-ended SCSI buses supporting a maximum cable length of 3 meters.

TURBOchannel Bus

The TURBOchannel bus is a synchronous, 32-bit, asymmetrical I/O channel that can be operated at any fixed frequency in the range 12.5 MHz to 25 MHz. It is also an open bus, developed by Digital, whose architecture and interfaces are publicly documented.
At 12.5 MHz, the peak data rate is 50 MB per second. At 25 MHz, the peak data rate is 100 MB per second.
The TURBOchannel is asymmetrical in that the base system processor and system memory are defined separately from the TURBOchannel architecture. The I/O operations do not directly

address each other. All data is entered into system memory before being transferred to another I/O option. The design facilitates a concise and compact protocol with very high performance.

XMI Bus

The XMI bus is a 64-bit wide parallel bus that can sustain a 100 MB per second bandwidth in a single processor configuration. The bandwidth is exclusive of addressing overhead; the XMI bus can transmit 100 MB per second of data.

The XMI bus implements a "pended protocol" design so that the bus does not stall between requests and transmissions of data. Several transactions can be in progress at a given time. Bus cycles not used by the requesting device are available to other devices on the bus. Arbitration and data transfers occur simultaneously, with multiplexed data and address lines. These design features are particularly significant when a combination of multiple devices has a wider bandwidth than the bus itself.

VME Bus

Digital UNIX includes a generic VME interface layer that provides customers with a consistent interface to VME devices across Alpha AXP workstation and server platforms. Currently, VME adapters are only supported on the TURBOchannel bus. To use the VME interface layer to write VMEbus device drivers, you must have the Digital UNIX TURBOchannel/VME Adapter Driver Version 2.0 software (Software Product Description 48.50.00) and its required processor and/or hardware configurations (Software Support Addendum 48.50.00-A).