

Computer Organization and Architecture

Under Graduate Course
(B. Tech-Information Technology, 2nd Semester)
Jan 2020-July 2020

By

Dr. Satish Kumar Singh



Associate Professor

Indian Institute of Information Technology, Allahabad

Email: sk.singh@iiita.ac.in

CONTENTS

Peripheral Devices

Input-Output Interface

Asynchronous Data Transfer

Modes of Transfer

PERIPHERAL DEVICES

Input Devices

- Keyboard
- Optical input devices
 - Card Reader
 - Paper Tape Reader
 - Bar code reader
 - Digitizer
 - Optical Mark Reader
- Magnetic Input Devices
 - Magnetic Stripe Reader
- Screen Input Devices
 - Touch Screen
 - Light Pen
 - Mouse
- Analog Input Devices

Output Devices

- Card Puncher, Paper Tape Puncher
- CRT
- Printer (Impact, Ink Jet, Laser, Dot Matrix)
- Plotter
- Analog
- Voice

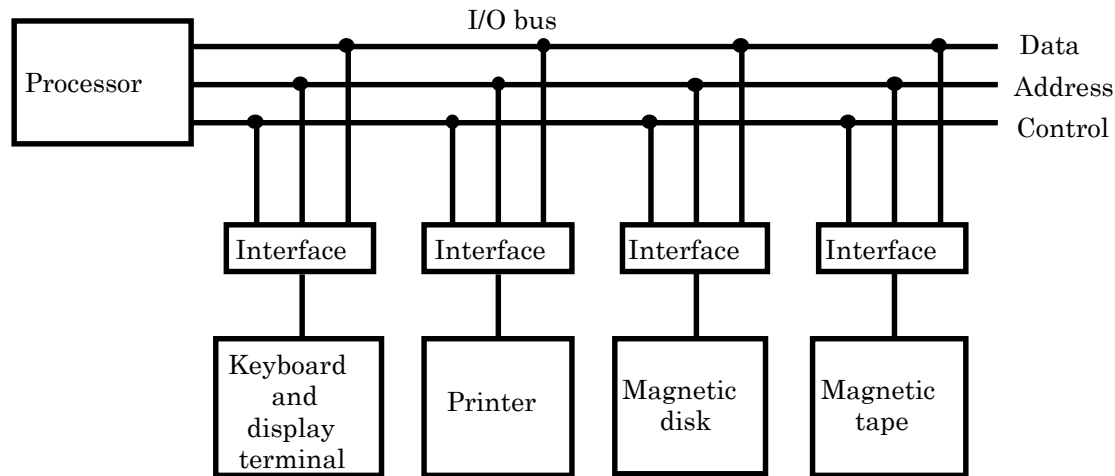


INPUT/OUTPUT INTERFACE

- Provides a method for transferring information between internal storage (such as memory and CPU registers) and external I/O devices.
- I/O devices need special communication links for interfacing them with the CPU.
- The purpose of the communication link is to resolve the *differences* between the computer and peripheral devices. The major differences are:
 - Peripherals – Electromechanical and electromagnetic Devices
 - CPU or Memory - Electronic Device
 - Therefore a conversion of signal values may be required.
 - Data Transfer Rate
 - Peripherals - Usually slower
 - CPU or Memory - Usually faster than peripherals
 - Some kinds of Synchronization mechanism may be needed
 - Unit of Information
 - Peripherals – Byte, Block, ...
 - CPU or Memory – Word
 - Data representations may differ. Operating mode of peripherals are different from each other and each must be controlled so as not to disturb the operation of other peripherals connected to the CPU.
- ***To resolve these differences, computer system includes special hardware components between CPU and I/O devices to supervise and synchronize all input and output transfers. These components are called interface units.***



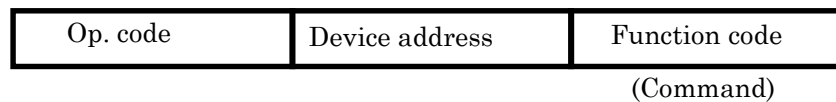
I/O BUS AND INTERFACE MODULES



Each peripheral has an interface module associated with it interface with do following:

- Decodes the device address (device code)
- Decodes the commands (operation)
- Provides signals for the peripheral controller
- Synchronizes the data flow and supervises the transfer rate between peripheral and CPU or Memory

Typical I/O instruction



COMMANDS

The I/O bus from the processor is attached to all the I/O devices interface.

To communicate with a particular device, the processor puts a device address on the address line.

When interface detects its own address, it activate the path between the bus line and the device that it controls.

At the same time that the address is made available in the address lines, the processor provides a function code in the control lines.

The interface selected respond to the function code and proceeds to execute it.

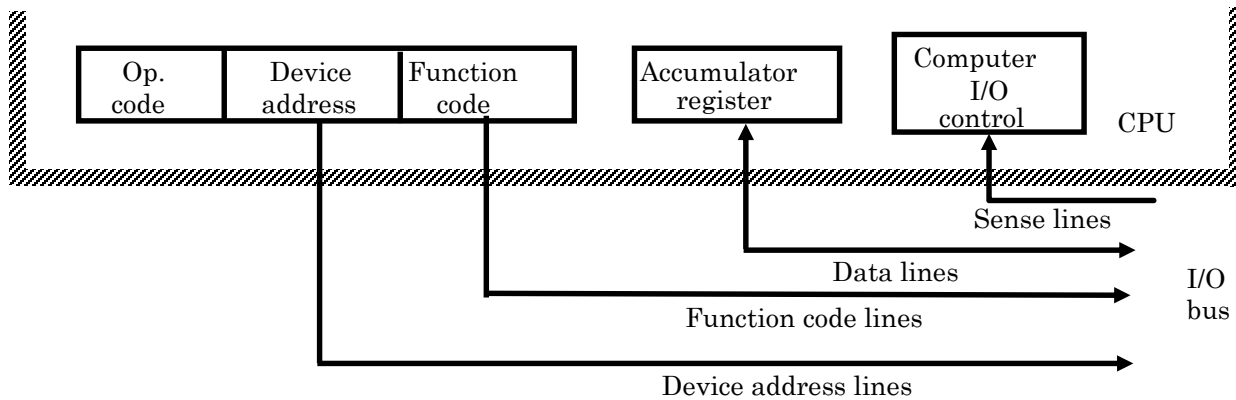
The function code is referred to as an I/O command and is in essence an instruction that is executed in the interface and its attached peripheral unit.

There are four types of command that an interface may receive.

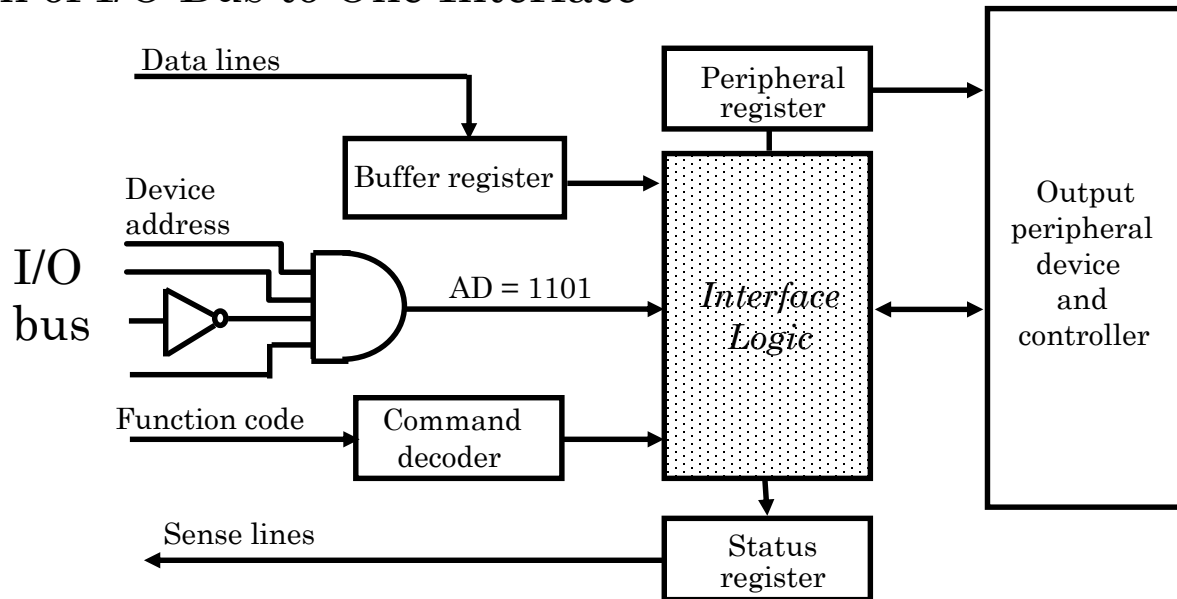
- Control,
- Status,
- Data Output,
- Data Input.

CONNECTION OF I/O BUS

Connection of I/O Bus to CPU



Connection of I/O Bus to One Interface



I/O BUS AND MEMORY BUS

In addition to communicating with I/O, the processor must communicate with the memory unit. Like the I/O bus, the memory bus contains data, address, and read/write control lines.

Functions of Buses

- *MEMORY BUS* is for information transfers between CPU and the MM
- *I/O BUS* is for information transfers between CPU and I/O devices through their I/O interface

Physical Organizations

- Many computers use a common single bus system for both memory and I/O interface units
- Use one common bus but separate control lines for each function
- Use one common bus with common control lines for both functions
- Some computer systems use two separate buses, one to communicate with memory and the other with I/O interfaces



ISOLATED vs MEMORY MAPPED I/O

Isolated I/O

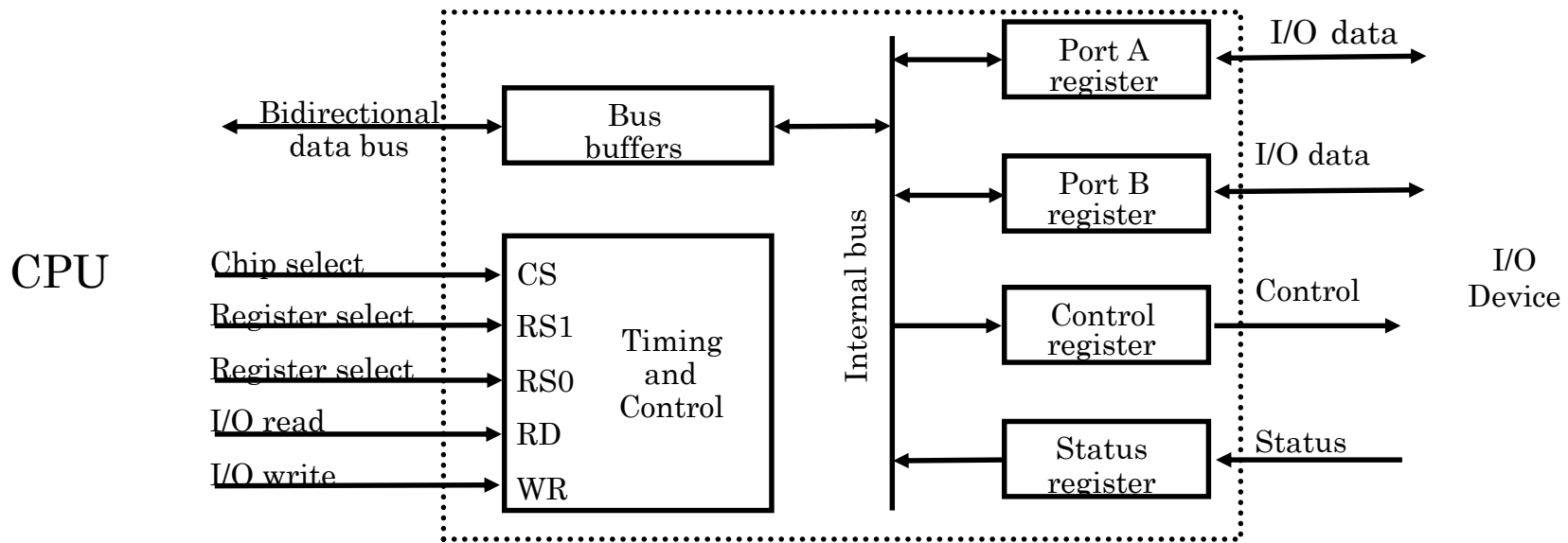
- Separate I/O read/write control lines in addition to memory read/write control lines
- Separate (isolated) memory and I/O address spaces
- Distinct input and output instructions

Memory-mapped I/O

- A single set of read/write control lines
(no distinction between memory and I/O transfer)
- Memory and I/O addresses share the common address space
 - > reduces memory address range available
- No specific input or output instruction
 - > The same memory reference instructions can be used for I/O transfers
- Considerable flexibility in handling I/O operations



I/O INTERFACE



The I/O data to and from the device can be transferred into either port A or port B.

CS	RS1	RS0	Register selected
0	x	x	None - data bus in high-impedence
1	0	0	Port A register
1	0	1	Port B register
1	1	0	Control register
1	1	1	Status register

The transfer of data, control, or status information is via a common data bus. The distinction between data, control, or status information is determined from the particular interface register with which the CPU communicates.

ASYNCHRONOUS DATA TRANSFER

Synchronous and Asynchronous Operations

Synchronous - All devices derive the timing information from common clock line

Asynchronous - No common clock

Asynchronous Data Transfer

Asynchronous data transfer between two independent units requires that *control signals* be transmitted between the communicating units *to indicate the time at which data is being transmitted*. One way to achieving this is by means of a STROBE pulse method. Other way is HANDSHAKING method.

In general case we consider the transmitting unit as the source and receiving unit as the destination.

ASYNCHRONOUS DATA TRANSFER METHODS

- Strobe pulse

- A strobe pulse is supplied by one unit to indicate the other unit when the transfer has to occur.

- Handshaking

- A control signal is accompanied with each data being transmitted to indicate the presence of data.

- The receiving unit responds with another control signal to acknowledge receipt of the data.

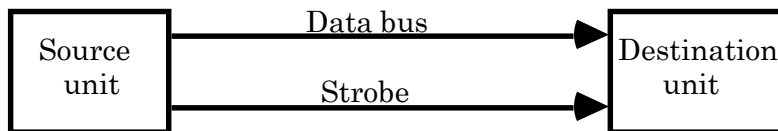


STROBE CONTROL

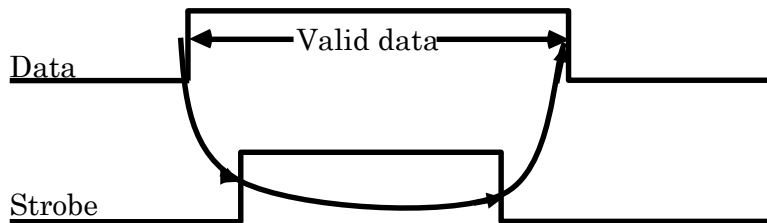
- * Employs a single control line (STROBE) and a data bus .
- * The strobe may be activated by either the source or the destination unit.

Source-Initiated Strobe for Data Transfer

Block Diagram

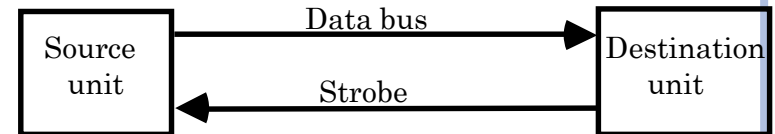


Timing Diagram

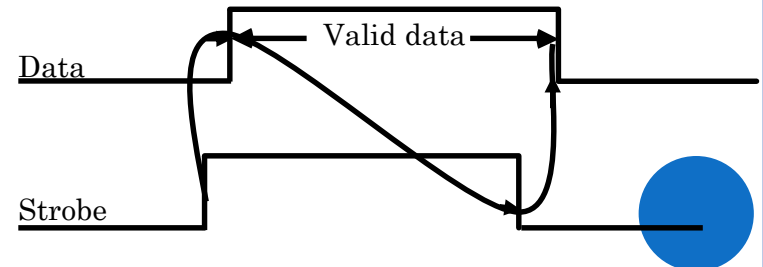


Destination-Initiated Strobe for Data Transfer

Block Diagram



Timing Diagram



HANDSHAKING

In Strobe Methods

Source-Initiated

The source unit that initiates the transfer has no way of knowing whether the destination unit has actually received data

Destination-Initiated

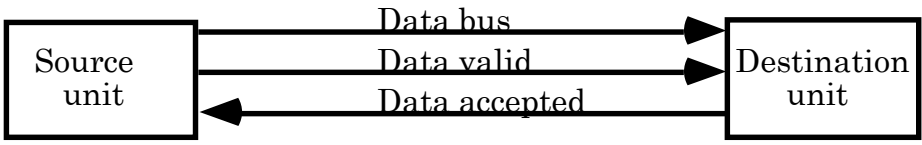
The destination unit that initiates the transfer has no way of knowing whether the source has actually placed the data on the bus

To solve this problem, the *HANDSHAKE* method introduces a second control signal to provide a *Reply* to the unit that initiates the transfer

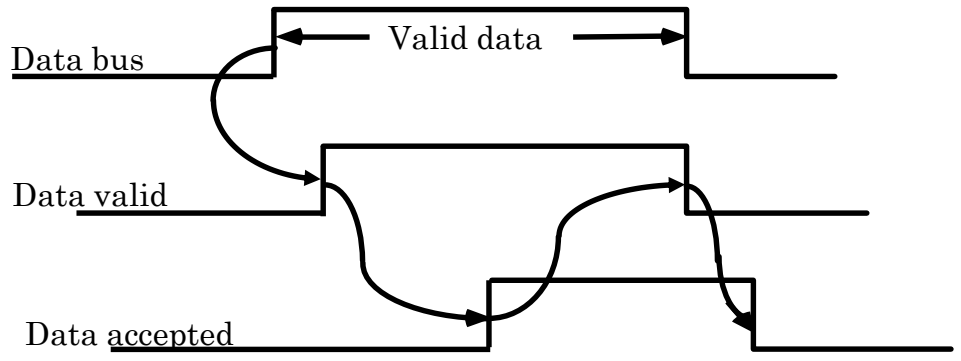


SOURCE-INITIATED TRANSFER USING HANDSHAKE

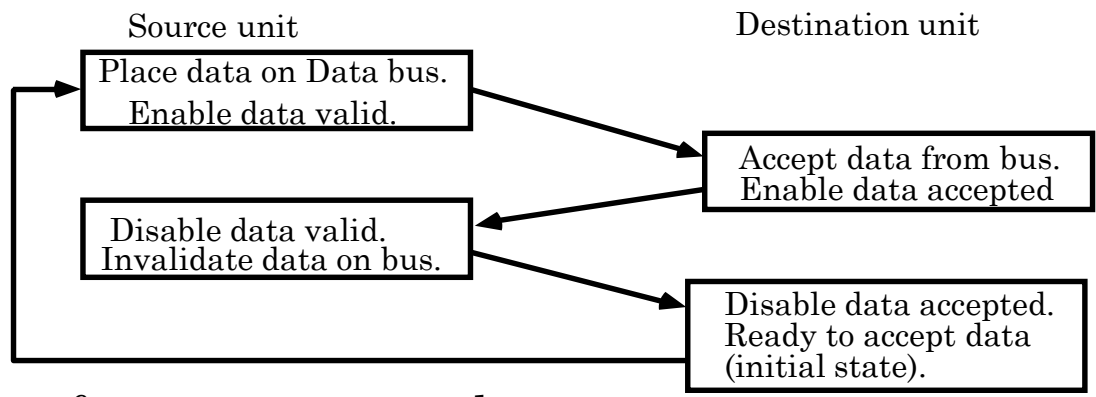
Block Diagram



Timing Diagram



Sequence of Events

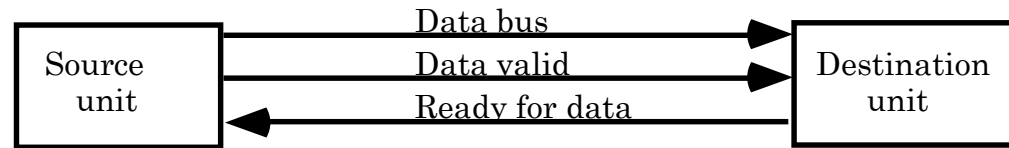


- * Allows arbitrary delays from one state to the next
- * Permits each unit to respond at its own data transfer rate
- * The rate of transfer is determined by the slower unit

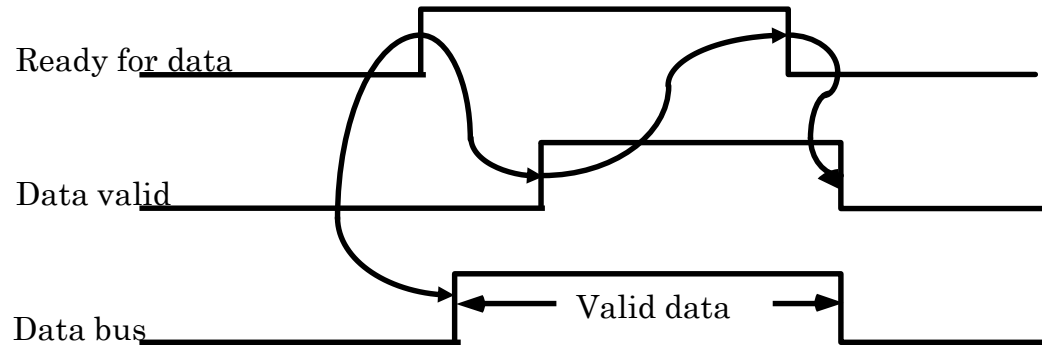


DESTINATION-INITIATED TRANSFER USING HANDSHAKE

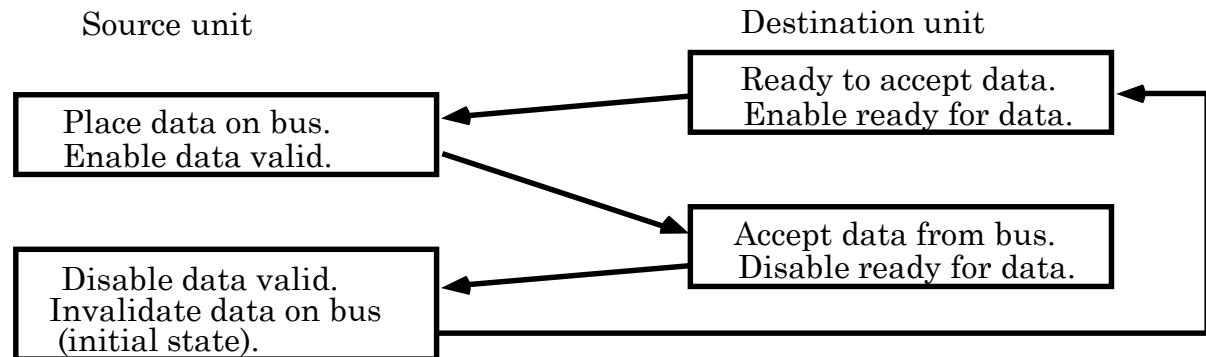
Block Diagram



Timing Diagram



Sequence of Events



- * Handshaking provides a high degree of flexibility and reliability because the successful completion of a data transfer relies on active participation by both units
- If one unit is faulty, data transfer will not be completed -> Can be detected by means of a *timeout* mechanism, which produces a alarm if data transfer is not completed in time.

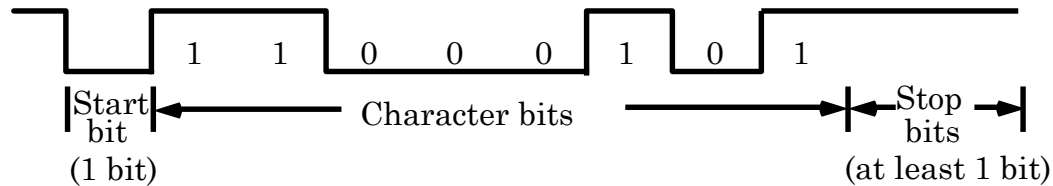
ASYNCHRONOUS SERIAL TRANSFER

Four Different Types of Transfer :->>>

Asynchronous serial transfer
Synchronous serial transfer
Asynchronous parallel transfer
Synchronous parallel transfer

Asynchronous Serial Transfer

- Employs special bits which are inserted at both ends of the character code
- Each character consists of three parts; Start bit; Data bits; Stop bits.



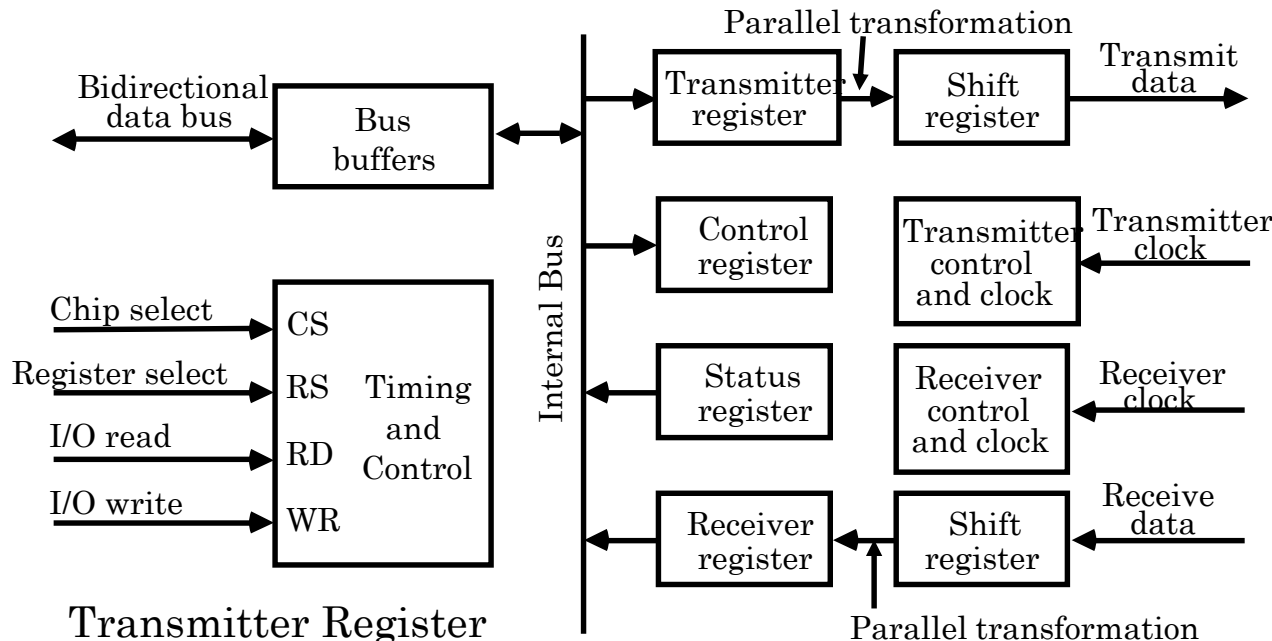
A character can be detected by the receiver from the knowledge of 4 rules;

- When data are not being sent, the line is kept in the 1-state (idle state)
- The initiation of a character transmission is detected by a *Start Bit*, which is always a 0
- The character bits always follow the *Start Bit*
- After the last bit of the character, a *Stop Bit* is detected when the line returns to the 1-state for at least 1 bit time



UNIVERSAL ASYNCHRONOUS RECEIVER-TRANSMITTER - UART -

A typical asynchronous communication interface available as an IC



It functions as both as a transmitter and receiver.

CS	RS	Oper	Register selected
0	x	x	None
1	0	WR	Transmitter register
1	1	WR	Control register
1	0	RD	Receiver register
1	1	RD	Status register

Transmitter Register

- Accepts a data byte (from CPU) through the data bus
- Transferred to a shift register for serial transmission

Receiver

- Receives serial information into another shift register
- Complete data byte is sent to the receiver register

Status Register Bits

- Used for I/O flags and for recording errors (parity, framing, overrun error)

Control Register Bits

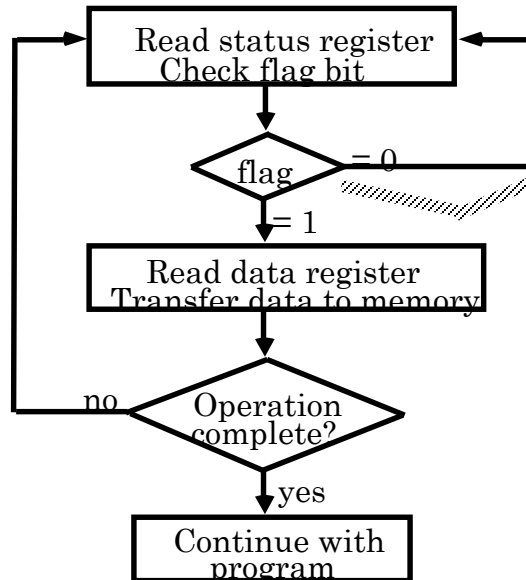
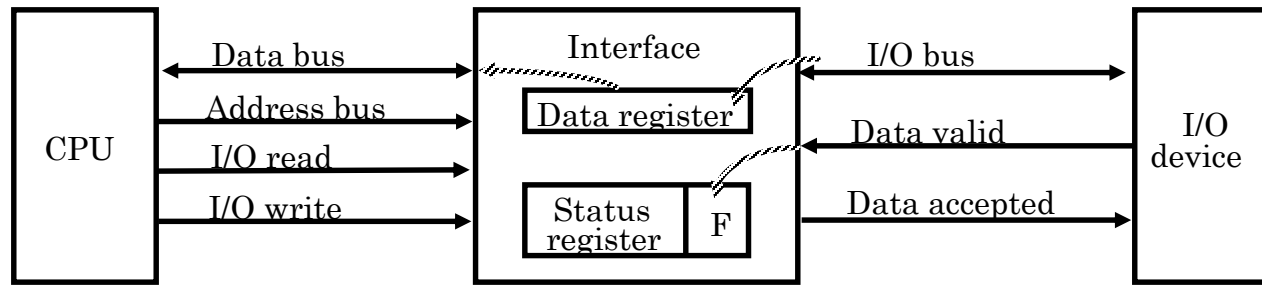
- Define baud rate, no. of bits in each character, whether to generate and check parity, and no. of stop bits, used for initialization.

MODES OF TRANSFER - PROGRAM-CONTROLLED I/O -

3 different Data Transfer Modes between the central computer(CPU or Memory) and peripherals;

- Program-Controlled I/O
- Interrupt-Initiated I/O
- Direct Memory Access (DMA)

Program-Controlled I/O(Input Dev to CPU)



Polling or Status Checking

- Continuous CPU involvement
- CPU slowed down to I/O speed
- Simple
- Least hardware

Transferring data under program control requires constant monitoring of the peripherals by the CPU.(CPU stays in a program loop). Useful for small low speed computers or in systems that are dedicated to monitor a device continuously.

MODES OF TRANSFER - INTERRUPT INITIATED I/O & DMA

Interrupt Initiated I/O

- Polling takes valuable CPU time
- Open communication only when some data has to be passed -> *Interrupt*.
- I/O interface, instead of the CPU, monitors the I/O device
- When the interface determines that the I/O device is ready for data transfer, it generates an *Interrupt Request* to the CPU
- Upon detecting an interrupt, CPU stops momentarily the task it is doing, branches to the service routine to process the data transfer, and then returns to the task it was performing
- CPU responds to the interrupt signal by storing the return address from the program counter into memory stack and then control branches to a service routine that processes the required I/O transfer.

DMA (Direct Memory Access)

- Large blocks of data transferred at a high speed to or from high speed devices, magnetic drums, disks, tapes, etc.
- DMA controller is a Interface that provides I/O transfer of data directly to and from the memory and the I/O device
- CPU initializes the DMA controller by sending a memory address and the number of words to be transferred
- Actual transfer of data is done directly between the device and memory through DMA controller --> Freeing CPU for other tasks

Thank you