Interfacing (Ch-5)

Date. No.

communication Basics

Basic terminology

wire

- connecting line of two terminals in communication system
- either uni-directional / bidirectional
- single line to represent multiple wire with small angled line as ⇒ —/—

BUS

- set of wires with single function
  (i) Address bus, (ii) data bus
* system bus consists of entire collection of wires: address, data and control lines.

Port / port

- medium through which signal is input to / out put from the processor.
- is conducting device to communicate peripherals by a processor.
- Also ~~Known~~ known as pin - that can be plugged into socket on PCB.

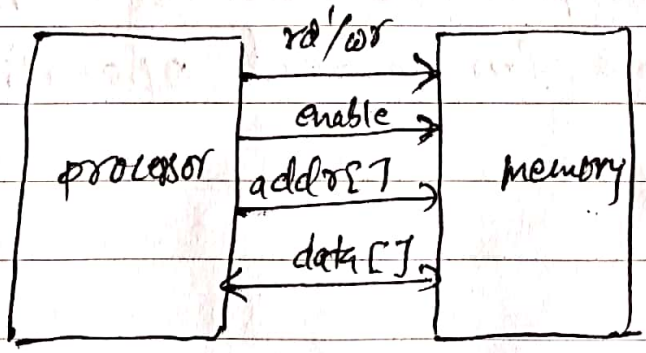


Fig Simple BUS example.

Date. No.

Timing diagram

- pictorial representation of hardware protocol.
- time proceeds along x-axis
- represents state of control lines & data lines.

* control line either high or low
* data line/address line can be either valid or not valid.

* active high
  - high (1) on the line make it active
* active low
  - low (0) on the line make it active.

* asserting a line
  - making line active

* de-asserting
  - the line deactivates the line.

Ex.: Timing Diagram for read protocol

- The processor must set rd'/wr line low for a read operation.

Date. No.

- address of the memory must be placed on addr line at least for t_{setup} time before setting the enable high.
- setting enable high will cause memory to place data on data line after time tread.

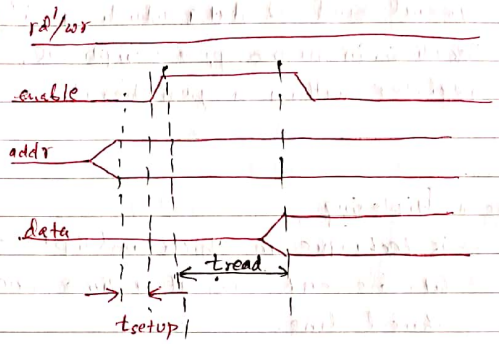


Fig. Timing Diagram: memory read protocol

Basic Protocol Concepts

Actor
- A device that can be processor or memory that takes part in data transfer.
- Actor can be master or slave.

* master initiate the data transfer whereas slave responds to the initiation request.

Date. No.

Data Direction

- represents movement of data among actors.
- data direction is independent of type of actor. Either master or slave can send or/and receive data.

Address

- special kind of data which specify a location in memory, a peripheral, or a register within a peripheral.

Time multiplexing

- is technique in which multiple sets of data are sent one at a time over the shared line.

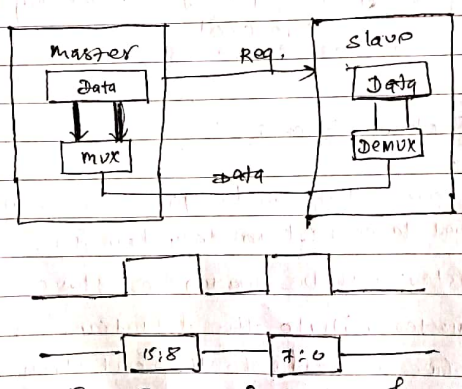


Fig. Time multiplexing for data transfer.

Date. No.

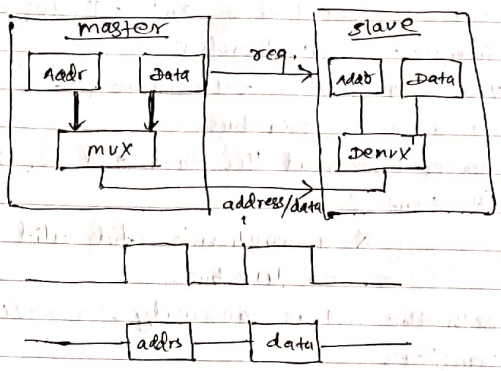


Fig. Address and data multiplexed transfer

Control method # ~~Data transfer protocol~~

- is scheme for initiation and end of the data transfer.

① strobe protocol

② handshake protocol.

① strobe protocol

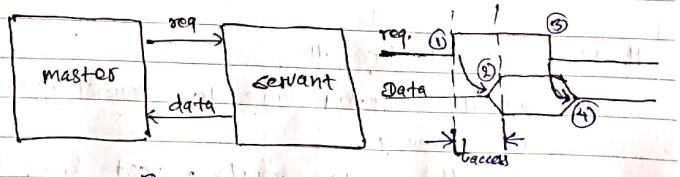


Fig. Strobe protocol.

Date. No.

- masters uses control line and activates it to initiate the data transfer.
- Slave/servant takes certain time (t_{access}) to put data on data bus.
- master then read the data and deactivate its control line
  * assumption is ~~valid~~ valid data on data bus.
- Both the actors are ready for next data transfer.
  * disadvantage is master does not have knowledge whether slave received the data request.

Time flow in timing diagram

① master asserts req. to receive data.
② servant puts data ~~and deasserts req~~ on data line within time t_{access}.
③ Master receives data and deasserts req.
④ servant ready for next request.

Handshake Protocol

- servant uses extra line to acknowledge that data is ready.
- Initially, master asserts request line to start the data transfer.
- servant, taking its time to put data on data line, asserts acknowledge line to inform the master the data is ready.
- master reads the data ~~line~~ from data line and deasserts the request.
  * it follows deasserting acknowledge line by slave/servant.
- data transfer complete; both actors are ready for next transfer.
  * complex, somehow, but reliable - compared to strobe protocol.

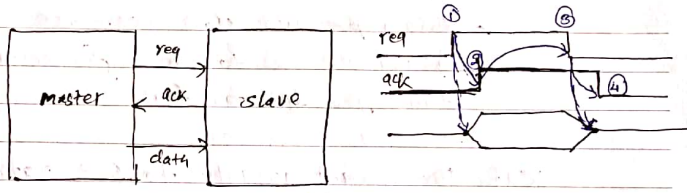


Fig - Handshake Protocol

Timing Diagram can be explained as:

① Master asserts req line to receive data
② servant puts data on data line & asserts ack.
③ Master receive data from data line/bus and deasserts req.
④ servant read for next transfer

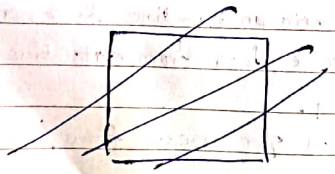
Date. No.

strobe / Handshake compromise

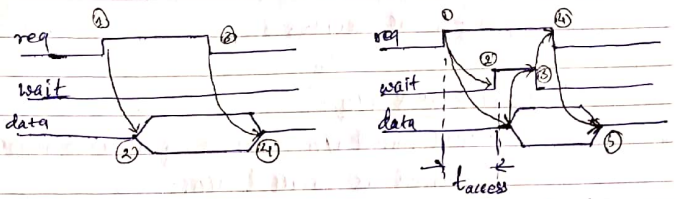
- inteded to achive the speed of strobe protocol,
- intended to handle varying response time of handshake protocol.

* If the servant is ~~unable to~~ unable to put data within time t_{access} then it asks master to wait longer by asserting wait line.
  - after the data is ready, wait line is deasserted by servant and master receives data.
  - this represents slow response as master has to wait for certain time for data transfer.

* If the servant is able to put data within time t_{access} then it follows strobe protocol
  - this represents fast response
  * wait line is not used at this time.



Date. No.



A strobe / handshake compromise : Fast and slow response

* Fast response timing diagram
  1. master asserts ~~request~~ req line to receive data
  2. servant puts data on data bus within t_{access} wait line remains unused.
  3. master receive data and deasserts req.
  4. servant ready for next request.

* slow response timing diagram
  1. master asserts req to receive data.
  2. servant can't/unable to put data within t_{access}, asserts wait line.
  3. servant puts data and deasserts ~~req~~ wait.
  4. master receive data and deasserts req.
  5. servant is ready for next request.

Date. No.

Microprocessor Interfacing

A) I/O Addressing

- A microprocessor uses its pins defined for input/output purpose to communicate peripheral devices.

* Port based I/O

- port based i/o addressing is also referred referred as parallel I/O.
- Generally the processors may be provided with one or more (4-bit) ports to facilitate port based I/O.
  - * each port is addressable.
- the port can be read from or written into directly with processor's instruction like a register.
  - * For example: In 8051 AVR µP, it has 8 bit I/O ports.

P1 = 0XFF statement will write into port 1.

P2.4 = 1 set the pin 4 of port 2

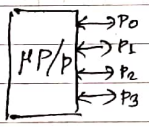


Fig. Port Based I/O

* bus-based I/O

- processor has address, data and control ports that for I/O addressing which form a single bus.
- The communication protocol is built

Date. No.

into the processor.

- Single instruction causes the hardware to write or read data.

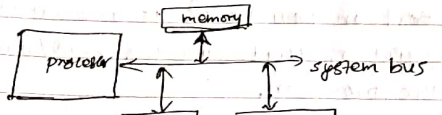
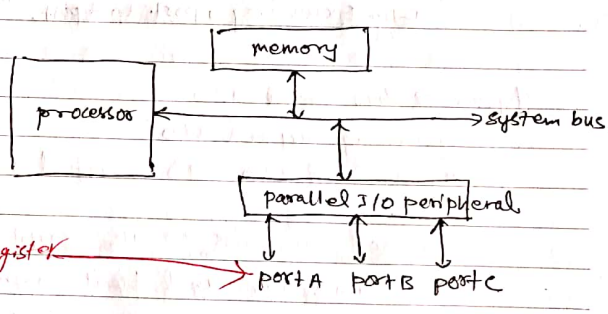


Fig. Bus based I/O addressing

parallel I/O peripheral (extended bus-based I/O)

* When a system with bus based I/O needs parallel I/O then parallel I/O peripherals can be connected to the system bus.



- each port on peripheral connected to a register within peripheral that is read/written by the processor

Date. No.

Extended parallel I/O

- For the processor with limited port-based I/O, one can extend the available port with parallel I/O peripheral.
- each port on peripheral is associated with a register that can be read/written by the processor.

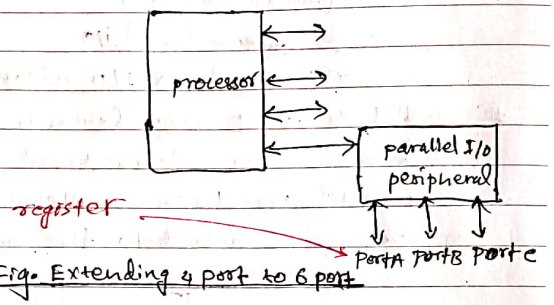


Fig. Extending 4 port to 6 port

Types of bus-based I/O

(i) memory-mapped I/O
(ii) standard I/O. (I/O-mapped I/O)

- A processor can talk to its peripherals using same bus and to its memory too.
- above are two ways to talk its peripherals

memory-mapped I/O

- Since memory as well as peripheral uses same bus, total address space is divided into memory address and peripheral address;

→ there is loss of memory address to peripherals.
- because of same address space for both memory and peripherals, similar instruction like mov for memory can be used to work with peripherals.

Eg:- A bus with 16 bit have total 65536 addresses; some upper addresses may be used for memory addressing while lower rest could be used to address peripheral addressing.

standard I/O

- additional pin (M/IO) on the bus indicates whether the address represents memory location or peripherals.

* possible unused addresses in either address spaces.
- For this kind of addressing, different instructions are needed to work with peripherals than memory

* mov, load → for memory
* In, out → peripherals.

Eg: Bus has 16 bit address (65536 addresses)
* When M/IO set to '0' → memory addressing
* When M/IO set to '1' → peripheral ".

Date. No.

* Interrupt
- peripherals might need services unpredictably from the processor.
→ so, there is issue of how to serve the peripherals by the processor while it remains busy on its own task.

(i) polling (ii) interrupt

(i) polling :
- periodic or in any sequential check for service requirement of every peripheral is done by the processor.
* this is easy to implement, but repeated checking could waste many clock cycles without doing certain useful work.

(ii) Interrupt :
- It's a feature of processor through which peripherals can request for service even when the processor is busy in its own task.
* for external interrupt, there is always a pin available to implement interrupt service / feature.
* when interrupt pin is asserted, the processor jump to a particular address at which the routine for the

Date. No.

interrupt is stored.
* though interrupt seems to overcome the limitation of poll, the interrupt pin is to be checked after the execution of every instruction; it does not require extra clock cycle.

Interrupt address vector
- represents the addresses in which the interrupt service routine (ISR) resides.
(i) Fixed interrupt (ii) vectored interrupt
(iii) interrupt address table.

Fixed interrupt
- address of subroutine is built into microprocessor and remain fixed.
- programmer simply has to store the ISR at that location or can put jump instruction to move to actual location of ISR where programmer has saved the ISR.

* Let data from sensor (peripheral-1) is to be read, processed and then a motor (peripheral-2) is controlled based on calculated data.
- following actions are to be carried
(i) peripheral-1 has its data in its register; meanwhile the processor is executing its main program.

Date. No.

(2) peripheral-1 asserts INT to request service from the processor

(3) after execution of each instruction, the processor checks INT pin. So, processor detects the service requirement.
- it saves its present context and sets the PC to the fixed ISR location.

(4) The ISR is executed which reads data from peripheral 1; process it and sends the resulting data to peripheral 2.
- peripheral-1 deasserts INT after data is read from it.

(5) the processor retrieves its state and resumes its work.

vectored interrupt
- peripheral must provide the address to the processor.
- In this method, both INT and INTA pins are also required. INTA acknowledge the interrup has been detected and the peripheral can provide the address of relevant ISR using system bus.
- The peripheral provides the address through the data bus which is read by processor.

Date. No.

* example with a two peripherals as in previous case can be carried out as:

(1) peripheral-1 has data in its register; at the same time the processor is executing its main program.

(2) Peripheral-1 asserts INT to request service from the processor.

(3) After execution of each instruction, the processor checks INT pin. So, processor detects the service requirement and it asserts INTA.

(4) peripheral-1 detects INTA and puts interrupt address vector on the databus.

(5) processor jumps to the address read from data bus and executes its corresponding ISR.
* it reads data from peripheral-1; processes it, and results to the peripheral-2.
* meanwhile peripheral-1 deasserts INT after data is read from it.

(6) The processor retrieves its state and resumes its work.

Interrupt address table
- is compromise between fixed and vectored interrupt.
- A table with ISR address is stored in processor memory.

Date. No.

- A peripheral instead provides the number rather address of ISR, corresponding entry in the table
* one major advantage is that the bit requirement to address the table is very small compared to number of bits of real address of ISR.
* It also provides flexibility to assign and change the location of ISR.

* Additional Issue in Interrupt ~~#~~

#i) - external interrupts may be maskable or non maskable.
- In maskable, the programmer can use specific instruction to disable the interrup by configuring certain bits of interrupt register.
- It is important when more critical works need to be executed first.

- Non-maskable interrupt can not be disabled by programmer. It requires a separate pin for drastic situations.
eg: - If power fails, the non maskable interrupt can cause a jump to subroutine that stores critical data in non-volatile memory before power is completely gone.

Date. No.

#ii) another issue
- is jump to ISR in which the microprocessor either saves complete context or partial state before jump to ISR.
* some processors save PC, registers which consumes many cycles, while others save the content of PC only.
- The ISR, however, must ~~notify~~ not modify registers if its content is not saved.

#c) Direct memory Access (DMA) controller

Introduction
- If communication between memory and peripherals involves microprocessor, there will be waste of ~~time~~ processor's time.
- For speed unmatched between processor speed and peripherals, data must be stored temporarily before processing which is referred as buffering - impacts on system performance.
- Additionally, Interrupt service ~~route~~ routine execution needs (storing and restoring the state of processor) many clock cycles.
- so, regular program stalls during transfer of data cause problem in the system performance.

Date. No.

~~X to relief~~

* to relieve processors from all data transfer involving memory and peripherals, a separate single-purpose processor called DMA controller can be used.

DMA controller

- used, specially, to transfer data between memories and peripherals.
- the peripherals request the service from DMA controller which then requests control of the system bus from processor.
- After that, processor releases the system bus.
- data transfer is initiated by DMA controller without the involvement of processor.
- this, storing & restoring of the state is eliminated.

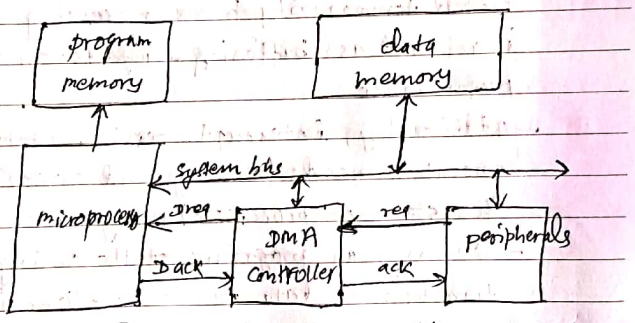


Fig. Simple system with DMA Controller

Date. No.

Operation

(1) initially, processor is busy executing its main program.
(2) once peripheral has data within its register, it asserts request line for service from DMA.
(3) DMA asserts request signal to the system from processor.
(4) Processor releases the system bus after seeing the request from DMA, and ack to DMA about the request.
(5) DMA asserts ack signal to peripheral; and starts transfer of data as requested.
(6) after completion of data transfer, all control lines are deasserted and processor retakes the control of the system bus.

Arbitration

- is mechanism through which a service or shared resources is provided to particular requesting device, out of many contenting devices for service.

(i) priority Arbiter (ii) Daisy-Chain Arbiter
(iii) Network oriented Arbiter

Date. No.

A) Priority Arbiter

- Single purpose processor to arbitrate among various requests from peripherals.
- Peripherals connected to the arbiter can make request for the service.
- Based on certain priority mechanism defined, arbiter selects the peripheral to permit the required service.

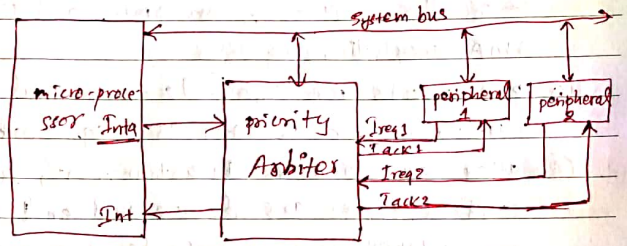


Fig. Arbitration using a priority arbiter

Above block diagram shows priority abiter to provide service to the peripherals connected with priority arbiter. (vectored interrupt)

* Arbiter is connected to system bus for configurations - may include setting priority of the peripherals.

* main advantage

- it can support advanced priority scheme.
- failure of a single peripheral does not have any impact on the operation of whole system.
- However, the system must be redesigned if new peripherals are to be added.

* so, this method is less flexible if new peripherals are required to be added or removed.

Operation

1. initially, microprocessor is busy in its own operation.
2. Both peripherals can assert request to priority arbiter which interrupts processor when at least one request is available from peripherals.
3. processor stops its current operation, stores its state and asserts interrupt acknowledge signal.
4. After acknowledgement by the processor, priority arbiter asserts acknowledge signal to any one peripherals based on priority.
5. the selected peripherals puts its interrupt address vector on the system bus.
6. microprocessor reads ISR address from data bus and jumps into it and execute the ISR.
7. After execution of requested ISR, processor retrives its state and resumes its operation.

# # #

Date. No.

types of priority arbiter

① Fixed Priority ⑪ Rotating priority or round-robin priority

Fixed Priority

- each peripheral is assigned a unique priority or rank value.
- when two peripherals request simultaneously for service then arbiter choose the one with higher priority value
* efficient for distinct priority among peripherals.
* But it can cause high-ranked peripherals to get much more service than other peripherals.

Round-robin priority

- In this method, each peripheral gets almost equal time for service from the arbiter.
- is efficient when there is not much difference in priority among peripherals.
- priority changes based on history of servicing of those peripherals.
- so, complex in rotating priority.

# # #

Date. No.

B) Daisy-Chain Arbitration :

- peripherals are connected in daisy-chain manner.
- the configuration is as follows:

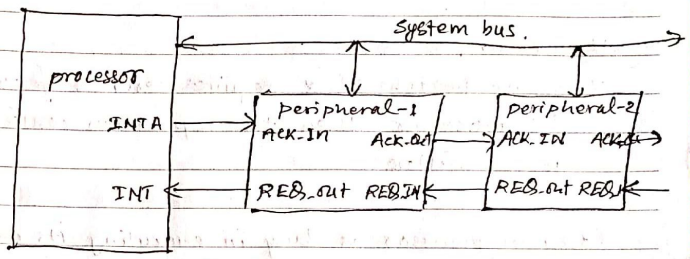


Fig. Daisy chain configuration.

- The request signal and acknowledge signals flow through the peripherals.
- peripherals request signal flows downstream to processor while processor's acknowledge signal flows upstream to requesting peripherals.
* peripheral connected first to the processor has the highest priority while the peripheral at the end of the chain has lowest priority.

* main advantage

- one can easily add or remove peripherals from the system without requirements of system redesign.
- does not support rotating priority.

Date. No.

- If one peripheral is damaged in the chain, other peripherals beyond that broken point will remain inaccessible as signal can not pass through the chain.

operation

* Suppose peripheral 2 requires services from the processor then the operation could goes as follows:

(1) microprocessor is busy in executing its own program.
(2) request signal from peripheral 2 is send to processor through the peripheral 1 and interrupt pin is asserted.
(3) processor stops its current work, stores its state, and asserts acknowledge signal
(4) ACK signal reaches to peripheral-2 through peripheral-1, since the request is generated by peripheral 2, not by peripheral-1; peripheral-1 passes the ack signal to peripheral-2,
(5) peripheral-2 puts its interrupt address vector on the system bus.
(6) microprocessor reads ISR address from data bus and jumps into it; execute the ISR.

Date. No.

(7) After execution of requested ISR, processor retrieves its state and resumes its operation.

Daisy chain aware peripherals:

- Generally, peripherals have ack in and request out line; but, in daisy chain aware peripherals must have additional ack out and request in lines.

* If peripheral does not have/contain ack out and request in, they should be made daisy chain aware by certain logic which adds complexity to the system.

eg:

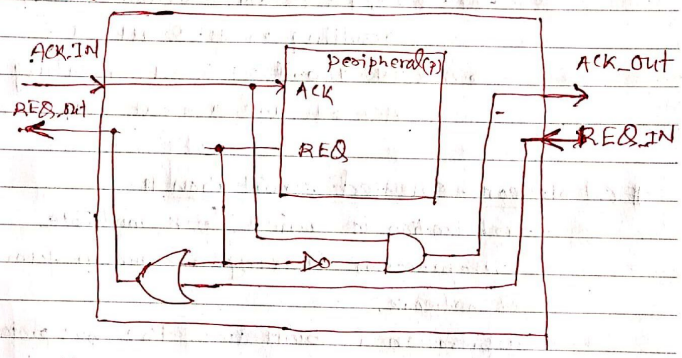


fig: Simple logic to make Daisy chain Aware

Date. No:

case1
- when request is from down stream peripherals
  → peripheral(p) does not participate in the flow of signal.

case2
- when request is from upstream peripherals beyond the peripheral(p)
  → REQ_IN = 1, but REQ = 0, resulting REQ_out = 1,
  → ACK_IN = 1, and REQ = 0, resulting ACK_out = 1,

case3
- when request is made from peripheral (p)
  → REQ = 1, REQ_IN = x (don't care) resulting in REQ_out = 1
  → ACK_IN = 1 and REQ = 1, resulting ACK = 1 and ACK_out = 0,

C) Network-Oriented Arbitration
- arbitration is carried for multiple microprocessor sharing a common form of network.
- arbitration is built into the bus protocol being bus a medium to connect multiple processors.
- for simultaneous access of the bus by multiple processor may cause data collision.

Date. No.

- the protocol must be designed in such a way that the contending processor do not start sending at the same time.
  * some kind of statistical methods could be applied to reduce the chance of data collision.

multi level Bus Architecture
- multilevel bus architectures are implemented in the system to improve the overall performance of the system.
- A single high speed bus is not enough to improve the overall performance of the system regarding communication.
- This is because of various drawbacks of using high speed bus.

* drawbacks are
  (i) inefficient interface
  (ii) slower bus interface of peripherals.

Inefficient interface
- Use of high speed bus requires peripherals with high speed bus interface
- However, all the peripherals do not need such high speed bus.

Date. No.

- High speed bus causes peripherals consume extra power, increase in number of gates, and so higher cost.
- ~~Being~~ Being high-speed ~~peripherals~~ bus ~~be~~ processor specific leading to non portable interfaces of peripherals.

#slower bus

- when many peripherals are connected to a single bus, all peripherals may not get the access to the bus when required.
- so, slow down of data transfer and performance lag.

two level bus system

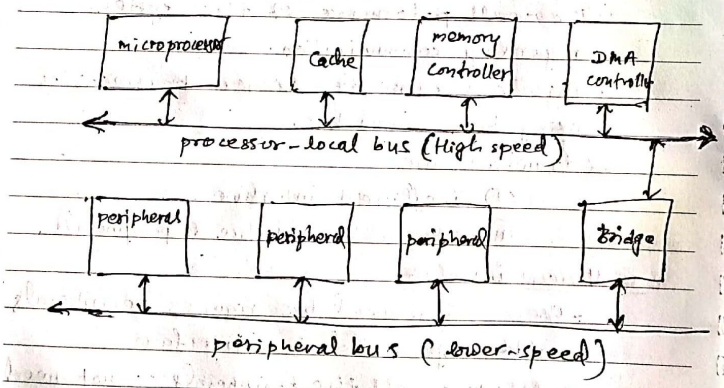


Fig. two-level bus architecture.

Date. No

① high speed processor-local bus ② slower speed peripheral bus

Generally, two level bus system consists of high speed & slower speed buses with bridge to connect the ~~bus~~ two buses.
- High speed bus is processor local bus while slower speed bus is peripheral bus.

- ~~the~~ processor local bus connects very high speed devices such as microprocessor, cache, memory controller, and certain high speed coprocessors.
- processor local buses are as wide as memory word and frequent communication takes place through it.

- peripheral bus connects peripherals which do not have access to processor-local bus
- it promotes portability, lower power, lower number of gates.
- It is often narrower and slower than ~~a proc~~ a processor local bus.
- frequency of communication through the bus is less compared to that of processor local bus.
- peripheral bus interfaces is comparatively efficient in terms of number of pins, gates and power consumption.

Date. No.

- A bridge is a single purpose processor that connects two buses of the system
- It makes various conversion required.
- speed synchronization is one important function of the bridge.
- data speed and data formate of processor local bus is different to that of peripheral bus; the compatability is resolved by bridge using various mechanism.

Three level bus hierarchy
(i) processor local bus
(ii) system bus
(iii) peripheral bus.

- processor local bus connects the processor to a cache
- may support one or more high speed devices.

- sytem bus can act as high speed bus.
- ~~offload~~ offload much of the traffic from processor local bus.

- peripheral bus is used to various peripherals to the system.

Date. No.

Advanced Communication principles
(1) parallel communication
(2) serial communication
(3) wireless communication
- infrared wave
- radio frequency

layering of communication
error detection and correction

(1) parallel communication #
- physical layer carries multiple bits of data at a time.
- Along with each wire carrying a single bit, the bus consists of data ~~wires~~ wire in addition to control and power lines.

Advantage :
- high data throughput : many bits are ~~transfom~~ transferred at a time.
- less complexity : easily implemented in hardware requiring only a latch to copy data onto a data bus.

Disadvantage :
- long parallel wires can result in Ferranti effect.
↳ * there is voltage build up due to capacitance.

Date. No.

* voltage at receiving end could become more than that at sending end.
- little variation in wire length may cause data misalignment because of different reach time of data signal.
- more costly and bulky and increased cost for insulation when interference is to be reduced.

usage
- this kind of connectivity ~~made~~ is made when devices resides on the same circuit board or same IC.

serial communication
- physical layer carries one bit at a time
- all bit transmission over single data wired bus takes place bit by bit
- the bus consist data wire along with control and power lines.

Advantage
- significant reduction in the size, complexity of connectors and corresponding costs.
- distant device communication could have better throughput compared to parallel communication of two distant devices.

Date. No.

- It does not exhibit Ferranti effect and data misalignment.

Disadvantage
- complex interfacing logic and communication protocols
- Data are decomposed into bits ~~and~~ at sending end and should be assembled at receiving end properly.
- For short distant communication, its ~~through~~ throughput is very less compared to that of parallel communication.

usage
- used to connect distant devices.
- It can connect short distant devices however, it is more efficient for distant communications.

wireless communication
- there is no physical medium connecting devices in communication,
- instead infrared and radio frequency channels are used as physical layer.
(i) Infrared wave (ii) radio frequency.

Date. No.

Infrared wave

- use electromagnetic wave frequencies that are below the visible light spectrum (3×10^{11} Hz – 4×10^{14} Hz)
- Infrared waves are generated using infrared diode whereas infrared transistors are used to detect the wave.
- infrared transistors ~~get~~ conducts when exposed to the wave.
- Advantage is cheap to build transmitter and receiver
- disadvantage - is line & sight requirement between communicating devices.
  * Also range & communication is low; so, inefficient method & communication for distant devices.

Radio Frequency

- uses EM wave in the range of 3 KHz to 300 GHz frequencies.
- for the communication, antenna and analog circuitry are required at communicating devices.
- main advantage : no need & line & sight (LOS)
  * longer distance communication is possible
- the range & communication distance is transmission power dependent.
- Disadvantage : transmitter and receivers are complex and costly.

Date. No.

layering

- hierarchical organization & communication protocols.
- lower level protocols provide services to higher level protocols.

* the main objective & layering is to simplify the study and design.
* the physical layer provides the lower level services to both sending and receiving bits or words & data.
* generally, physical layer signal is electromagnetic wave or electrical signal (voltage or current).
* Application layer provides the high level services to the user.

⇒

Date. No.

Error detection and correction

- error detection is the process to detect errors in bits during the data transmission.
- errors could be of two types
  (i) bit error (ii) burst of bit error

bit error

- single bit in the received data is invalid or changed

burst of bit error

- more than one bit get error or changed.

- error correction is process of correcting the bits that were detected during communication process.
  * parity and checksum are two basic error correction methods.

(i) parity check (ii) checksum error.

In parity check

- extra bit is send along with data to provide addition information about the data.

Date. No.

* when extra bit added makes the number of 1s in the data word ~~then~~ add then it is ~~before~~ refered to as odd parity else it is even parity.
- parity of data at sending terminal should be qual to that of parity of data at receiving end/terminal otherwise there is bit error.
- This method of checking parit is efficient for single bit error but for burst of bit error is not applicable to detect the change.

In checksum

- multiple of words of data in packets are checked for error.
- the extra word which represents XOR sum of all data words in a packet.
- transmitter sends the packet along with checksum words; the word is checked at bot sending and receiving end;
- If checksum is error free, the transmission is successful

Date. No.

#① serial protocol
② Parallel protocol
③ Wireless protocol

serial protocol

① Inter-IC or I2C or I^2C

* serial protocol for two-wire interface
* connects low-speed devices like ~~microcontr~~ microcontroller, EEPROM, A/D and D/A converter, I/O interfaces and other similar peripherals in embedded systems
* it has 7 bit or 10 bit address space.
* 7-bit addressing can address 128 devices ~~(bits)~~ over shared I^2C bus.
* common speed is 100 Kbit/s in standard mode / 10 Kbit/s in low-speed mode
* However, recent upgrade of I^2C can host more nodes and can run even faster speed
  * 400 kbit/s in fast mode
  * 1 Mb/s in fast mode plus
  * 3.4 mb/s in High speed
* uses only two wires
  - serial clock (SCL)
  - serial data (SDA)

Date. No.

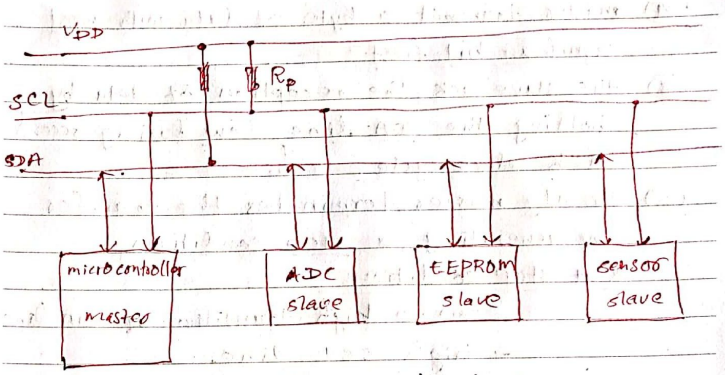


Fig. I^2C bus structure.

Typical ~~rea~~ write operation

① master initiate the transfer with a start condition
  * at start condition: * SDA line transition is high to low
  * SCL is high

② Then address of the device to which data is to be written is sent with most significant bit down to the least significant bit

③ For write operation
  - master send zero '0' after sending address
  - slave ~~ack to~~ acknowledge the ~~transition~~ transmission by holding SDA line low during first ACK clock cycle.

ACK

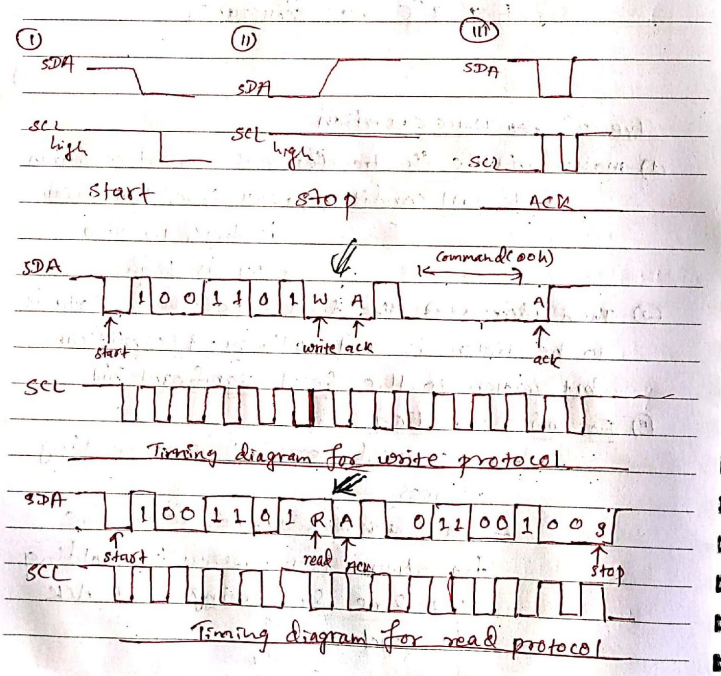
Date. No.

(4) master transmit a byte of data with most significan bit first

(5) the slave ack the reception of data by holding the SDA line low during second ACK clock cycle.

(6) Finally master terminates the transfer by generating a stop condition.

* at stop condition
  - low to high transition of SDA line
  - high SCL line.



Timing diagram for write protocol

Timing diagram for read protocol

Date. No.

(2) Serial Peripheral Interface (SPI)
- is synchronous serial communication interface specification.
- used in short distance communication
- sends data between processor/controllers and small peripherals.

* it uses separate clock and data line along with select line to choose the device that should be communicated with.

* consists four logic signal
  (i) serial clock (SCLK)
  (ii) (MOSI) Master output slave input
  (iii) (MISO) master input slave output.
  (iv) ss - slave select.

* SPI bus can operate with a single master device and with one or more slave devices

(3) Control Area Network (CAN):
- is ISO defined serial communication bus.
- originally targeted for automotive industry replacing complex wiring harness with only two wire bus.
- has high ~~impurity to~~ immunity to electrical interference and repairs data error.

Date. No.

Characteristics

* high-integrity serial data communication.
* real time support
* data rate up to 1 Mb/s
* error detection and confinement capabilities
  - balanced differential signaling
    - reduces noise coupling
    - high signaling rate over twisted pair cable.
* error detection forces the sending node to resend the message ~~to~~ until correct receive.

(4) # Firewire

- is serial bus for high speed data transfer.
- initiated by Apple and developed by IEEE P1394 group; so, this is IEEE 1394 protocol
- supports peer to peer device communication without the involvement of system memory or CPU.

* characteristic

- transfer rate up to 400 Mb/s.
- plug and play, and hot swapping
- packet based ~~layed~~ layered design structure

Date. No.

- power provision through cable.
* 64 bit addressing allows a local-area network to consists of 1023 sub-networks, each consists of 63 nodes.
- device orgnized at the bus in tree or daisy chain topology.

* two types of data transfer :
  (i) asynchronous
  (ii) isochronous

* asynchronous data transfer
- data transfer can be initiated as a given length of data arrives in a buffer.

* isochronous data transfer
- data flows at a pre-set rate.

(5) # Universal serial Bus

- designed to connect wide range of peripherals to a computer
  * displays, data storage, communication devices and other devices.
- it standarized the connection of computer peripherals to personal computers,
- it supports communication and power supply.

Date. No.

* USB 1.0 → has data transfer rates of
  * 1.5 Mb/s for low data rate devices.
  * 12 Mb/s for high speed devices
* USB 2.0 → 480 Mb/s
* USB 3.0 → 5 Gb/s

* some USB p can serve as connection ports for other USB peripherals
* USB hubs and standalone hubs can be used to provide convenient USB port.

Parallel protocols

① PCI Bus
- is peripheral component interconnect
- high performance bus to attach hardware to computer.
- It is synchronous bus architecture
- data transfer takes plance to the range of system bus.

* Max clock rate 66 MHz; use only up to 33 MHz (data rate 132 - 512 Mb/s)

- PCI implements a 32-bit multiplexed address and data bus
- it supports rigorous auto configuration mechanism.

② - ARM Bus (self)

③ # wireless protocols #

① Infrared Data Association (IrDA)
- is international organization that creates and promotes infrared data interconnection standards
- provides complete set of protocols for wireless infrared communications.
- IrDA - implemented in portable devices like smart phones, laptops, cameras
- support communication between device to device over point to point infrared at speeds (9.6 kbps - 4 Mbps)
- low cost and simple architecture
* line of sight, very low bit error rate and physically secure data transfer

② Bluetooth
- wireless technology standard, for data exchange over short distances from fixed and mobile device
- frequency of operation 2402 - 2480 MHz

Date. No.

- uses radio based link.
- does not need Line of sight communication
* Bluetooth 4.0 → upto 25 Mbps data rate
- packet based protocol with master-slave structure.
- one master can communicate up to seven slave devices.
- feature
  - low power and short range communication
  * power range and communication distance depends on the class of radio
  (A) class 3 radio
    - 1 mW (power) & up to 1 m (distance)
  (B) class 2 radio
    - 2.5 mW (power) & about 10 m (distance)
  (C) class 1 radio
    - 100 mW (power) & about 100 m (distance)

IEEE 802.11
- is set of media access control (MAC) and physical layer (PHY) specification for wireless local area network (WLAN).
- often known as Wi-Fi,
- Data rate about 1-2 Mbps.
* has variety of standards with ~~suffix~~ letter suffix.

Date. No.

ex: 802.11a, 802.11b, 802.11g, 802.11n.
- 802.11 wi-Fi standards operate within ~~the~~ Industrial, scientific and medical (ISM) frequency band.
- the ISM band cluster ~~around~~ 2.4 GHz reserved for Industrial, Scientific and medical equipment that uses RF.
- Data rate up to 54 Mbps for some standard while few standards (latest) goes up to 6.75 Gbps.

* PHY layer defines physical link transmitting bits from one node to another node in a network.
* Modulation, line coding, synchronization are few functions
* MAC layer provides addressing and channel access control mechanism that ensures the communication of several nodes ~~with~~ within a shared medium.
* Each device is given unique serial number known as MAC address
* multiple access protocol enables several stations to share the same physical medium.
  Eg: CSMA/CD.

Chapter-7 (control system)

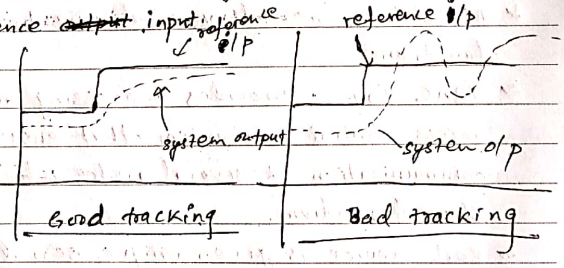
Date. No.

Control system

- A class of embedded system,
- focuses on tracking the reference input that is provided to the system.
- initially reference input is ~~more likely to be~~ set to certain value and then output is more likely to track same input level.

* tracking might get difficult when there is disturbance.
* however, the system must be able to adjust to external factors for optimum performance.

objective of a control system is to track the reference ~~output~~ input/reference i/p



Open-loop and closed-Loop Control system overview.

Open loop control system

- output has no influence on the control action of the i/p signal.
- it is feed-forward system or no-feedback system. (ie output is not fed back for control)
- controller is not aware of reference input → resulting no optimization.

* this mode is best in the case when system model is accurate and disturbance effect is minimal.

Components of open loop control system?

① plant ② output ③ reference ④ actuator ⑤ controller ⑥ disturbance.

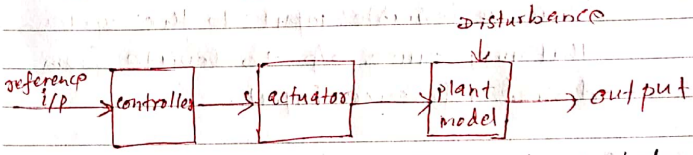


Fig. block diagram of open loop control.

 # plant

- is process to be controlled
- is physical system, eg: fan, heater.

 # output (is parameter / measure)

- is attribute of the physical system that we are about to control. eg: speed, temperature.

 # reference

- is desired input to get desired output of the physical system.

Date. No.

actuator
- is device to control the input to the plant, eg: motor can be taken as an example of an actuator.

controller
- is the main processing part of the system
- compute the input to the plant such that desired o/p is achieved on the given reference input.

Disturbance
- is an undesirable input to the system that may cause o/p to deviate from desired reference input.

closed loop control system
- operates on feedback principle.
- output is fed back, compared with reference input and error signal is produced.
- the controller process the error signal, and reduces the error to obtain the desired output.

* controller is aware of output variations,
→ optimization can be done minimizing the error.

Date. No.

components of closed loop control system
① plant ② output ③ reference ④ controller ⑤ actuator ⑥ disturbance ~~⑦ closed loop control system~~ ⑦ sensor and ⑧ error detector

* sensor is used to sense the output and is fed to the input; calculating error, error signal is input to controller.

* error detector: determines the error ~~pro~~ produced in the system. Error is, in general, difference between output of the system and reference input.

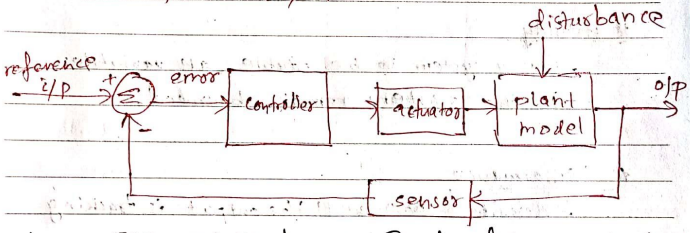


Fig: Block diagram of closed loop control.

comparison of open-loop & closed loop control

open loop control	closed loop control
① feed forward; no o/p feed back.	① feed back; o/p is feed back and compared with i/p
② simple and economic	② complex and expensive
③ good calibration needed and no optimization	③ reduced error increases accuracy & optimized
④ slow & unreliable but stable	④ fast & more reliable but unstable.

Date. No.

General control system and PID controllers

Control objective

- main objective is to make output track the reference input;
  - *even in the presence of measurement noise, model error, and disturbance.
- * parameters for study and analysis if objective fullfilled:
  - (i) stability, (ii) performance (iii) Disturbance rejection (iv) Robustness.

stability

- for a system to be stable, all variables in the system remain bounded.

performance

- describes how well the output is tracking the change in reference input.

* different aspects of performance are
(1) Rise time (T_r) (2) Peak time (T_p)
(3) Overshoot (M_p) (4) settling Time (T_s)

(1) Rise Time (T_r):
- time required to change from 10% to 90% of its final value.
- it measures the ability of a system to fast input signals.

(2) Peak Time (T_p):
- time to reach the first peak of the response.

(3) Overshoot (M_p):
- o/p exceeding its final, steady-state value.
- It is the percentage amount by which the peak of the response exceeds the final value.

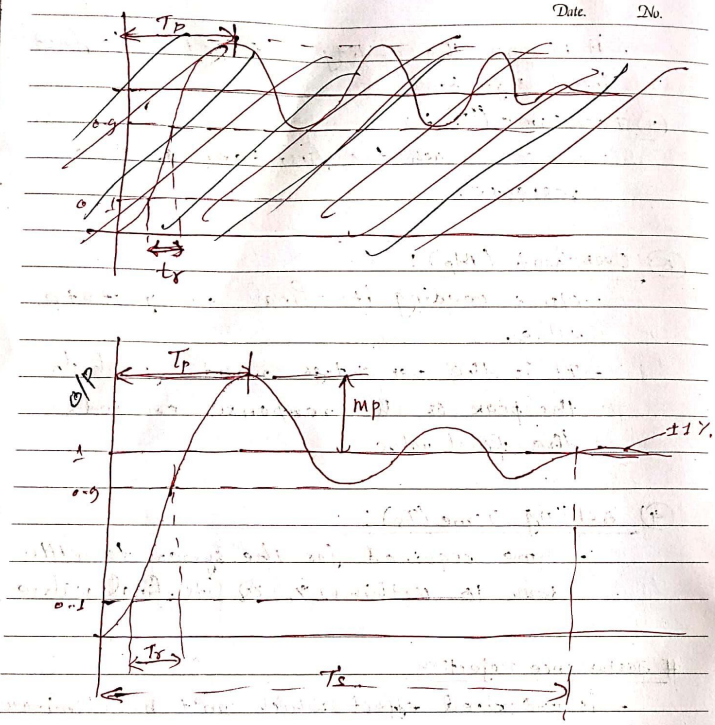
(4) Settling Time (T_s):
- Time required for the system to settle down to within ±1% of its final value.

Disturbance Rejection

- is undesired effect which can't be eliminated but impact can be minimized

Robustness

- designed system must tolerate the modeling error.
- model error should not effect much to the stability and performance of the system.



Aspect of performance metrics in control system response.

Transient Response

* it occure when there is undesired condition or the system starts — just after the system starts.
* the response during settling time is transient response.

steady state response

- is response after settling time.
- steady state error is difference between the actual output and desired output when there occures steady state response.

Modeling Real Physical System

* the accurate modeling of behavior of the plant is essential.
* controller is designed based on the plant model; so, the plant model must be as accurate as possible.

Features of Real System

* Continuous in nature :
  - System response is continuous function of time.
  - Physical system continuously reacts, so plant model can be represented by differential equation; for more than or variable — partial differential equation.
  - for equivalent discrete time model, response should be sampled at smaller time than the reaction time of the system.

Date. No.

* Complexity
  - complex model than general design.
  - In general design, non linear effect, all system interactions are excluded.
  * Generally, model designed is linear.

Controller Design

(i) proportional control (P) ~~(ii) proportional~~
(ii) proportional and derivative (PD) control
(iii) Proportional and Integral (PI) control
(iv) Proportional Integral and Derivative (PID) control.

Proportional control
  - controller multiplies the tracking error by a constant.

$\therefore u(t) = P * e(t)$

  - $u(t)$ is output of controller,
  - P is proportional constant
  - $e(t)$ is measure of error = difference between input and output reference input.

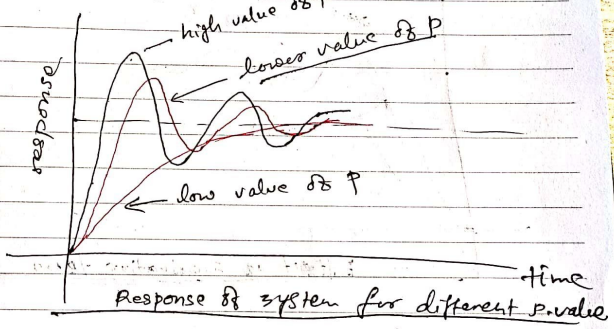
* Proportional constant (P) affects transient response, steady state tracking error and disturbance rejection.

Date. No.

* high value of proportional constant (P) can cause system to become unstable by resulting in high overshoot and oscillation.

* low value of proportional constant (P) will cause the system to be less response to less sensitive, because rise time is high for low value of P.

* steady state error will be high for low value of P.



Response of system for different P.value

proportional and Derivative (PD) control
  - derivative predicts the system behavior
  - improves the settling time and stability of the system.
  - Derivative allows the transient response to be optimized without affecting the steady state response and disturbance rejection.

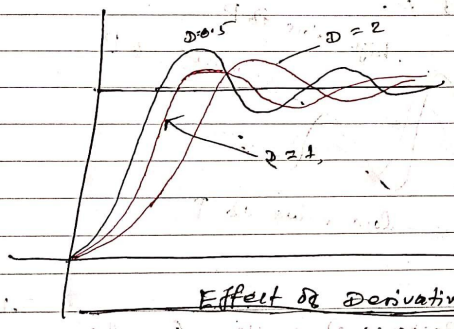
Date. No.

- Hence, transient response and steady state error independently can be adjusted by using appropriate value of P and D in PD controller

$$u(t) = P * e(t) + D * [e(t) - e(t-1)]$$

characteristics of PD control

- rise time reduces, improves damping, overshoot reduces, response is stable



Effect of Derivative term.

proportional and Integral (PI) control

- integral is the sum of instantaneous error over time and the ~~accumulated~~ accumulated error is multiplied by integral constant -

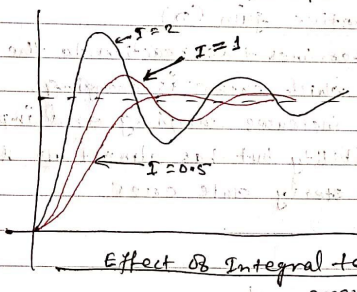
Date. No.

$$u(t) = P * e(t) + I * [e(0) + e(1) + e(2) + \dots + e(t)]$$

* PI controller is used to eliminate the steady state error resulting from P-controller.
* However, it has undesirable impact on speed and stability of the system.

characteristics of PI control

- steady state accuracy improves.
* rise time increases, response is oscillatory



Effect of Integral term in system response

Proportional Integral and Derivative (PID) control

- it helps to attain a set point irrespective of disturbance.
- It calculates its output based on the measured error.

Date. No.

① The proportional gain (P)
- simply multiplies the error by factor P. It reduces the ~~steady~~ steady state error while reduced effect of external disturbance.

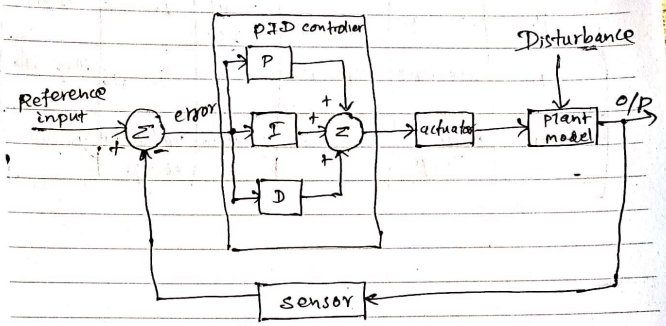
② integral gain (I)
- multiplies the sum of the recent errors.
- It eliminates the steady state error.

③ Derivative gain (D)
- determines the reaction to the rate at which error is changing.
- it increases damping and improves stability but has almost no effect on steady state error

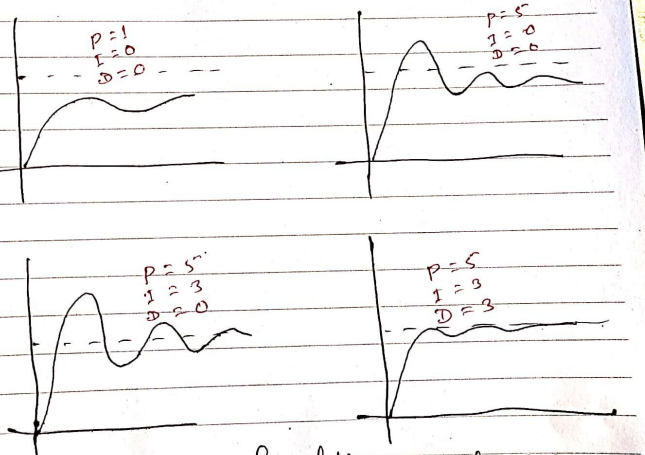
controller output

$$u(t) = P * e(t) + I * [e(0) + e(1) + e(2) - \cdots + e(t)] + D * [(e(1) - e(0)) + (e(2) - e(1)) + \cdots + (e(t) - e(t-1))]$$

Date. No.



General Block Diagram of PID Controller



response for different value of P, I & D

Date. No.

PID control summary

Type	Rise Time	Maximum Overshoot	Settling Time	Steady state error	Stability
P	Decrease	Increase	small change	Decrease	Degrade
I	Decrease	Increase	Increase	eliminate	Degrade
D	small change	Decrease	Decrease	NO / small change	Improve