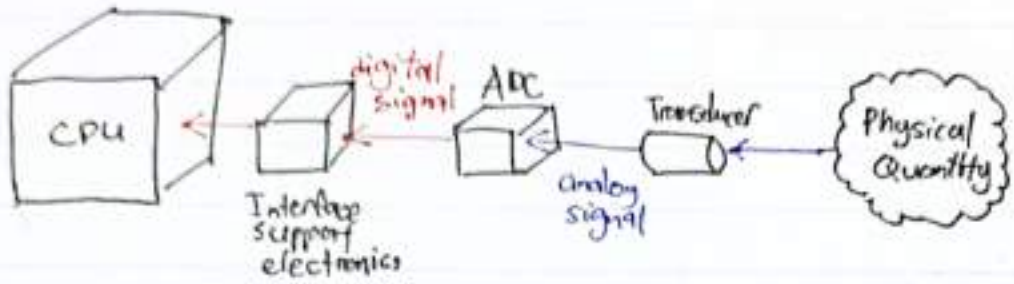
No. CINTER - 1
Date 21 . 5 . 22.

PC Interfacing.



CPU → represents the personal computer

ISE → The digital signal is then sent to be interface module, which will then be connected to the personal computer

ADC → The analog signal will have to be converted using an ADC so that it will get a digital equivalent.

Transducer → Convert the raw data into it's corresponding electrical signal, and very frequently, is in analog form.

Physical Quantities → External devices, the physical quantity, raw data is usually not in any electrical form.

No. CINTER -2
Date 21 . 5 . 22.

The Personal Computer.

8088 micro-processor → original PC → 20-bit address bus.
For compatibility, the pentium and its successors can still be operated in 8086 or 8088 real mode, as well as the more up-to-date protected and virtual 8086 mode.

REAL MODE

8086 real mode → 2 CPU registers → segment register.
↳ offset register.

16-bit

	1	2	3	4 ← segment
+	5	6	7	8 ← offset.
1	7	9	B	8 } 20-bit → thus, it has 1MB.

The addition of the register produces a sum of more than 1MB, the address wrap-around to the lower memory area.

	F	F	F	F
	F	F	F	F
0	F	F	E	F

ans suppose to be 10FFEF. ← the carry is not generated.

- 80386 processors (i386) → 32-bit micro processor, → this and its successors don't have the 1MB restriction when ~~operated~~ ~~in~~ they emulate the 8086 real mode
- The segment & off-set register are kept at 16-bit each, during the addition to form the resultant memory address, the carry generated from the 20th position to the 21st position will not be lost, this is because the address buses of these processors are more than 20-bit each.

PROTECTED MODE

At protected mode, 8086 & 8088 have 1MB limit, to get around this while allowing backward compatibility.
- 80286 CPU can work in real & 16-bit prot mode
- 80386 CPU can work in real & 32-bit prot mode

Address range for real & prot mode.

	Real	16-bit prot	32-bit prot.
Address Line	20-bit (1MB range)	24-bit (16 MB range)	32-bit (4 GB range)

Protected mode was first implemented to the 80286 to protect the diff task in multitasking OS from invalid or incorrect accesses. To do this, the processor hardware checks all memory access (code & data) made by a program and uses 4 privilege level to provide access rights for such accesses.

No.

Date

RING ARCHITECTURE

An 80386 or higher processor has 4 privilege levels, known as rings that control things such as memory access, and access to certain sensitive CPU instructions (e.g. those related to security).

Ring 0 → kernel mode → most privilege level, with complete access to all memory and CPU Instructions.
Low lvl OS code execute in kernel mode/space (linux)

Ring 1 & 2 → not used.

Ring 3 → User mode → least privilege level.
User app code runs in user mode/space (linux).

Intel 80386 & higher processors Ring architecture

① Power-Up → CPU started at 8086 real-mode executing the routines in the Boot ROM.

② Then OS takes over control of the PC before any app software is started.

③ Since OS has initial control, it set its kernal or core to the highest privilege level → Ring 0 (PL=0).

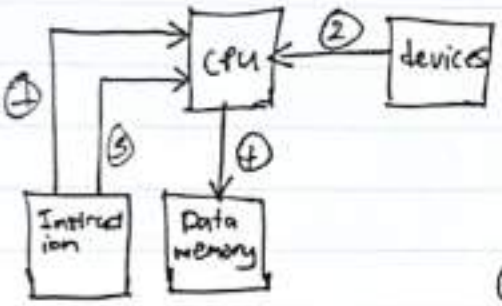
④ App programs are given the lowest privilege level → Ring 3 (PL=3), because although the app programs use the resources of the PC, it is the OS that manages the resources. Thus, it's important that the app program should not interfere with how the resources are managed.

CINTER -2.
21.5.22.

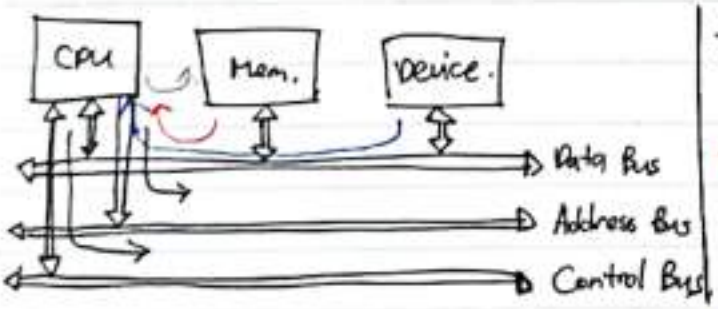
PROGRAM CONTROLLED I/O OPERATIONS

Normally when I/O operations are performed, they are under program control, i.e. according to the program instructions as and when the need arise.

- Works well when the data transfer is small over a given period. and do not require memory buffer.
- If the data is large and required to be transfered within a short period of time and required storage space, the I/O operations under program control may not be satisfactory.



1. CPU fetches instruction from program memory
2. Data is transmitted to CPU register. Data has to go through an intermediate store, CPU register, before finally settled into the data memory
3. CPU fetches next instruction from program memory.
4. Data is transmitted from register to data memory



time t=0 → CPU sends out address & control signal to fetch instructions.

t = 1 → Program memory returns instruction

t = 2 → CPU executes instructions by sending address & control.

t=3 → Input device place data on data bus and places it in a register.

t=4 → CPU sends out addres & control signals to fetch next instructions

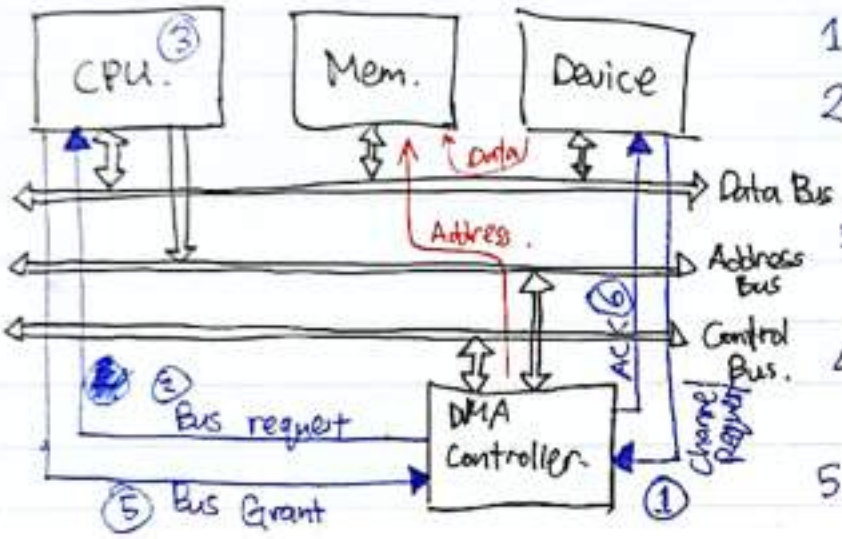
t=5 → Program memory returns next instructions

t=6 → CPU executes instructions by sending address & control

t=7 → CPU places data on data bus, memory stores data

DIRECT ~~ACC~~ MEMORY ACCESS.

- DMA is ~~where~~ a scheme whereby the I/O operations don't involve the CPU; data is directly transferred between the device and the consecutive memory location.
- To coordinate transfer, a DMA controller is needed. (a secondary bus master). A bus master is a unit that controls a given bus by generating address and control signals on the bus.
- A Bus Exchange protocol is carried out by the controller to supervise the data transfer, also have the control of the system bus so that the CPU would release the system bus to the DMA controller.



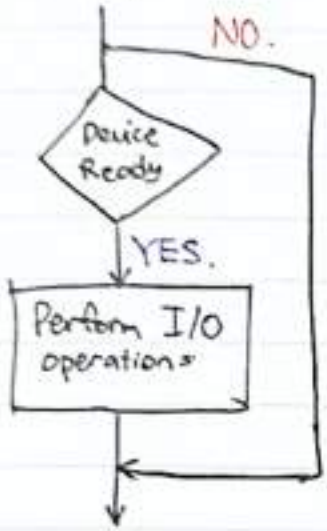
1. Device generate DMA channel request
2. DMA controller sends bus request to CPU.
3. CPU would first complete its current instruction execution.
4. Pending on no interrupt request, CPU releases system bus.
5. CPU sends bus grant to DMA controller.
6. DMA controller sends ACK to device.

Once the DMA controller has possession of the system bus, it would generate a ~~pre~~ predetermined starting memory address for the data transfer tgt with appropriate control signal (I/O_Read to the device and MEM_WRITE to the memory)

The process is repeated with the next unit of data stored directly from the device into the next memory location until the whole block of data has been transfered.

POLLED I/O

Some devices have a signal to indicate their readiness for data transfer. Under program - controlled I/O, the program would, from time to time, check the ready status of the device.

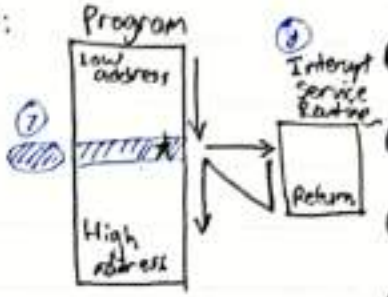


- If the device is now ready for data transfer, the program then performs the I/O operations with the device.
- If the device is not ready for data transfer, the program then carries on with its other tasks until it's time to check the device's status again.

It is also possible that the program really wants to perform I/O operation with the device at certain point in time, and would wait for the device to be ready, rather than doing sth else first.

INTERRUPT REQUEST.

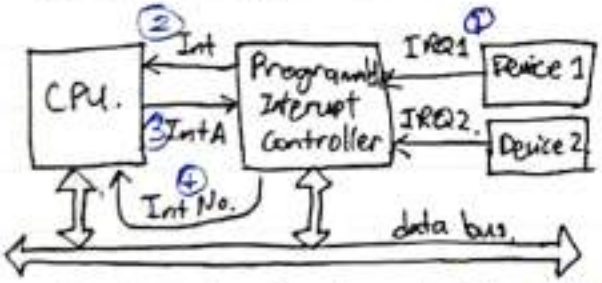
An interupt is an occurance of exception that cause:
- Program execution to temporarily deviate from it's normal cause of operations.
- Another task being performed to handle the situation that cause the interuption (interupt service).



- After the interupt had been serviced, the program resume its previously interupted operations.
- There are 2 catagories of interuptions that could happen to the program execution flow:
  - Hardware interuptions.
  - Software interuptions

Hardware interupts

- For hardware interupts from external devices, there is no need to identify which device actually generated the request.
- In the case of 2 or more interupts occurs, there needs to be a interupt controller to decide which one to attend to first, it is placed between the devices and the CPU.



1. If a device wants immediate attention from the CPU, it being sending / recieving data, It would generate a interupt request to the PIC.
2. If the interupt channel of the PIC is not masked, and is the highest priority pending interupt, the PIC would activate the Interupt Line to the CPU.
3. If interupts are not masked in the CPU, CPU completes the current instruction and then activates the interupt acknowledge line to ask for an interupt number, 8-bits from the PIC.
4. The PIC sends the interupt number of the corresponding device that made the interupt via the data bus.
5. The 8-bit number recieved by the CPU is skewed by 2-bit to the left to form the entry address of the interupt table e.g. $00000010_2 \rightarrow \underbrace{0000000000001000}_{16\text{-bit offset}} = 008H$
6. From the interupt table, the interupt vector is read by the CPU for the starting address of the interupt service routine.
7. The CPU saves the basic context register.
8. CPU jumps to the starting address of the ISR and executes instructions from there. The service routine may furthur save additional context before starting its own task.
9. Once the ISR is complete, it replaces back context previously saved by it & then executes a return instruction.
10. Return instruction would cause the CPU to replace back the previous context registers.
11. The CPU is now back to whr it was interupted, and continues like nothing happened

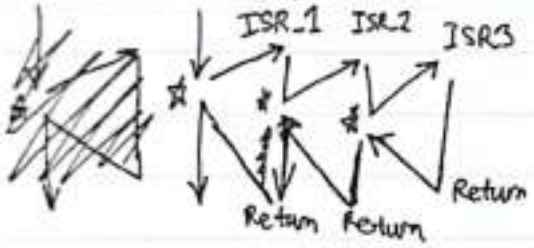
No. CINTER -2.
Date 22 . 8 . 22.

INTERRUPT REQUEST

Software Interupts

- Like hardware interupt, but it is initiallized ~~by~~ / initiated by interupt instruct code, which is pre-planned within the program.
- ~~Also~~ It behaves as if it has been interrupted but instead of getting an Int No. from the PIC, the instruction actually includes the Int. No.
- So with this number, CPU performs the same operations as ~~mentioned~~ in the hardware interupts' step 5 all
- A Software Interupts is a powerful feature of the CPU. This allows app program to call system-level subroutines without knowing the address of the subroutines.

Nested Interupts Services.



- When CPU performs an interupt service routine, further interupts from devices having lower or the same priority as the current device being serviced would be barred.
- However, a device with a higher priority can still ~~[illegible]~~ interupt. service routine causing the CPU to ~~[illegible]~~ switch over and service its request.
- This is known as nested interrupt service.

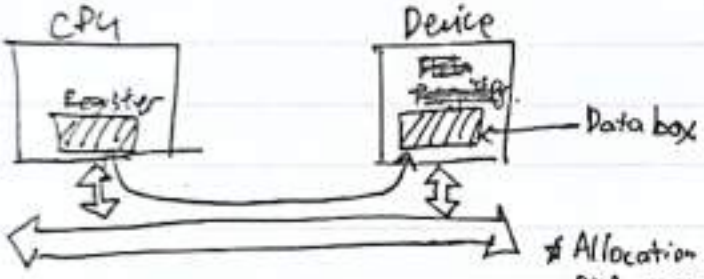
I/O DEVICES

Accessing I/O Devices

When a CPU performs an I/O operation, it requires the following to carry out a successful in/out transaction between the CPU and an external device:
- Source & destination address.
- Size of data (number of bytes)
- Direction of transfer (reference to CPU)
- Actual instance for transfer (timing)

The above is based on the assumption that the data is alr available & waiting to be transfered.

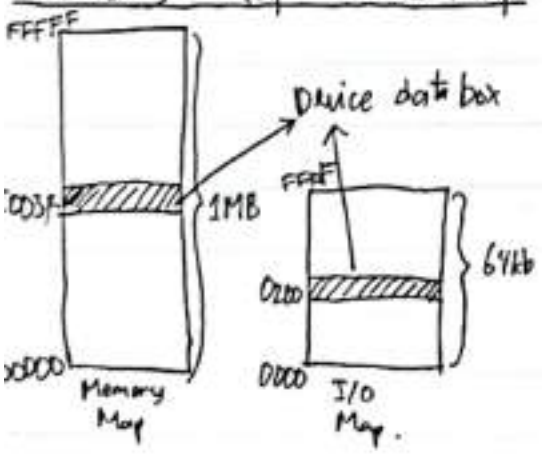
Since the I/O operation involves the transfer of data between the CPU and an I/O device, a CPU register and the address of the I/O device data box would be specified in the machine instruction that cause the transaction to occur.



The address of the I/O device data box may be allocated either in the memory map or in the I/O map. This is done by hardware arrangement.

* Allocation is permanent & cannot be switched from one map to the other by means of software.

Memory Map & I/O Map.



In the case where the I/O device data box is allocated in the memory map, the data box is treated just like any other memory location; Any instruction that moves data in or out of a memory location may be used to perform I/O operation with the data box.

The address used in the instruction for the data box will be a 20-bit address formed by a segment register and an offset, both specified in the machine instruction.

I/O DEVICES

Accessing I/O - Mapped I/O

For I/O device data box that is allocated in the I/O map, there are only 2 types of instruction that may be used for I/O operations:

OUT [port address], [acc]
IN [acc], [port address]

[acc], or accumulator, is the only CPU register involved in the holding of the data, the data size depends on whether AL, AH, AX, or EAX is specified. (AL, AH = 8-bit, AX = 16-bit, and EAX = 32-bit)

CPU is able to differentiate between the 2 maps based on the instruct it recieves:

- Memory referenced instruction generates read or write control signal, activating the selected address in the memory map.
- I/O referenced instruction generates the I/O read or I/O write control signal, activating selected address in the I/O map.

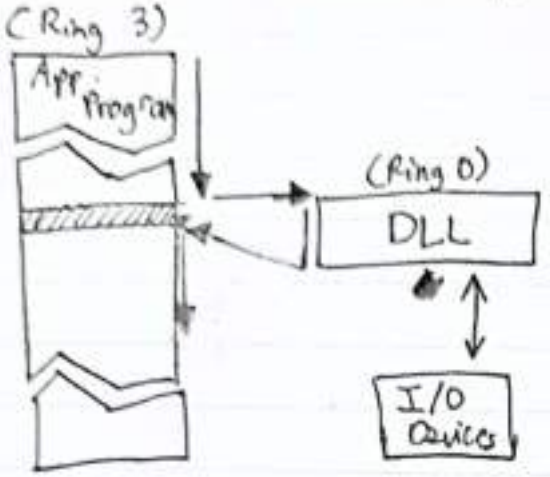
Accessing I/O - using High-Level Programming Languages.

- Most app program use High-level programming languages, as it provide standard functions for the programs to interact with standard I/O devices and / or peripherals.
- An App program is given the lowest privilege in the protected mode (ran is Win OS)
- Within the program, no codes are allowed to directly access the I/O systems and memory (which are not in the legal range, but whr the memory mapped I/O are). Although there is an instruction to switch from protected to real mode, this can only be done when the process has the highest privilege level ie. OS core. An app program cannot switch itself over to operate under the real-mode.

No.

Date

Dynamic Linked Library (DLL)



A DLL is a device driver containing code tht a win. app program call can on. To do this, the app program must have declaration statement indicating the DLL needed, so tht it will be loaded in with the app.

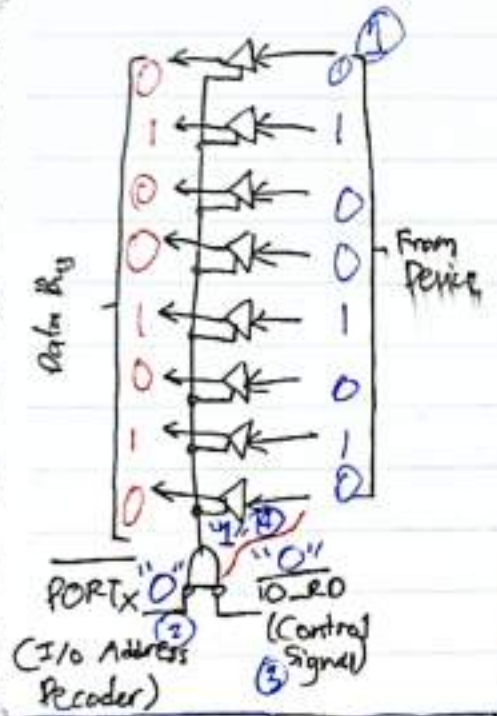
DLL executes at Ring 0 (highest privilege lvl), therefore they are able to access any resources directly (subject to availability - and usually under Win.'s control)

PARALLEL I/O INTEFACING

PARALLEL INPUT PORT.

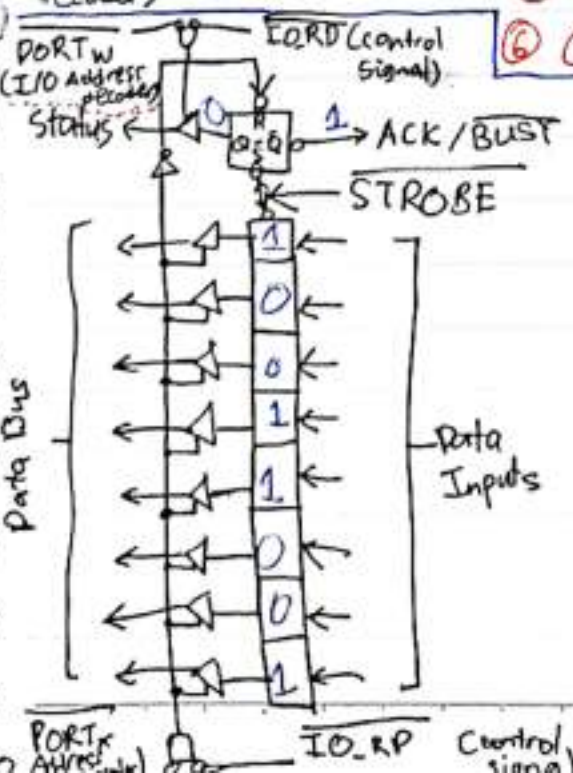
Parallel input port is the interface through which parallel data from external device can be input into the personal computer.
By parallel ~~it means that it consist of~~ it meant the unit of data, which consist of number of bits, typically 8 bits can be transfered in one go.
To construct a PIP, the 2 basic DE component need to be understood.
- Tri-stat buffer
- Data Latch.

TRI-STATE® BUFFERS & DATA LATCHES.



This type of parallel input port is suitable for those devices such as on/off switches and sensors whose output are readily available

① CPU first sends the I/O Adress of the port onto the address bus.
② I/O address decoder recognizes the I/O address and assert the select signal $\overline{PORTx}$
③ CPU generates the _IORD control signal
④ Tri-stat buffers are enabled
⑤ Device's data appears on the data bus.
⑥ CPU loads data to accumulator (acc)



Parallel Input Port using Tri-state® Buffers and Data ~~Lat~~ Latches.

Tri-state Buffer.

E	IN	OUT
HI	LO	LO
HI	HI	HI
LO	X	Hi-Z

Data Latch

$\overline{LE}$	D	Qt
1	X	Q
0	0	0
0	1	1

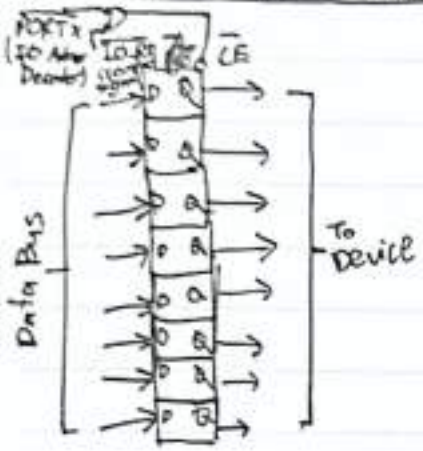
No.

Date

PARALLEL OUTPUT PORT.

- Parallel output port is the interface through which parallel data from the PC can be output to an external device.
- To construct a POP., we still need to use either one of the 2 components, tri-state buffer and data latches.

USING DATA LATCHES.



This type of output port is suitable for those output devices such as LEDs and control relays, which are ready to accept any output from the port at any time

The ports ~~are~~ outputs are typically at TTL logic levels:

"Lo" = 0V ~ 0.2V.

"Hi" = 3.6V ~ 5V

It may require additional drivers depending on the output device connected to it.

USING TRI-STATE BUFFER ← If need use handshake protocol

① Assume that previous output data had been taken by the device, so device set: _BUSY → "Hi" Indicating the CPU it is ready to accept another data.

② CPU reads from Status port PORTw BUSY = "Lo"

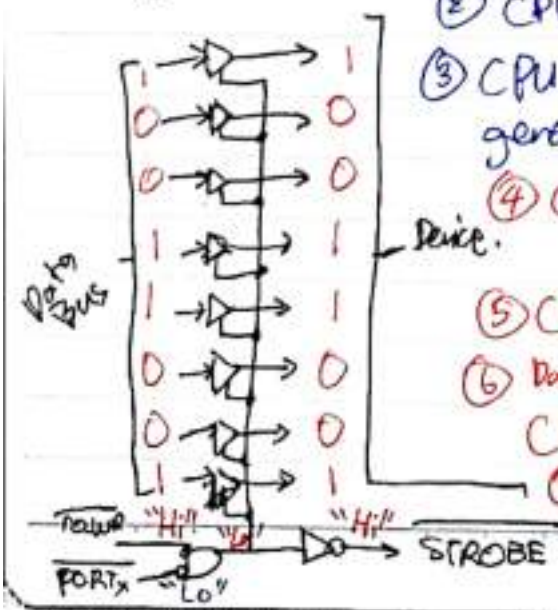
③ CPU sends address of PORTx I/O address decoder generates _PORTx

④ CPU ~~sends address of PORTx, I/O address dec~~ places data onto the data bus

⑤ CPU generates /IO_WR control signal

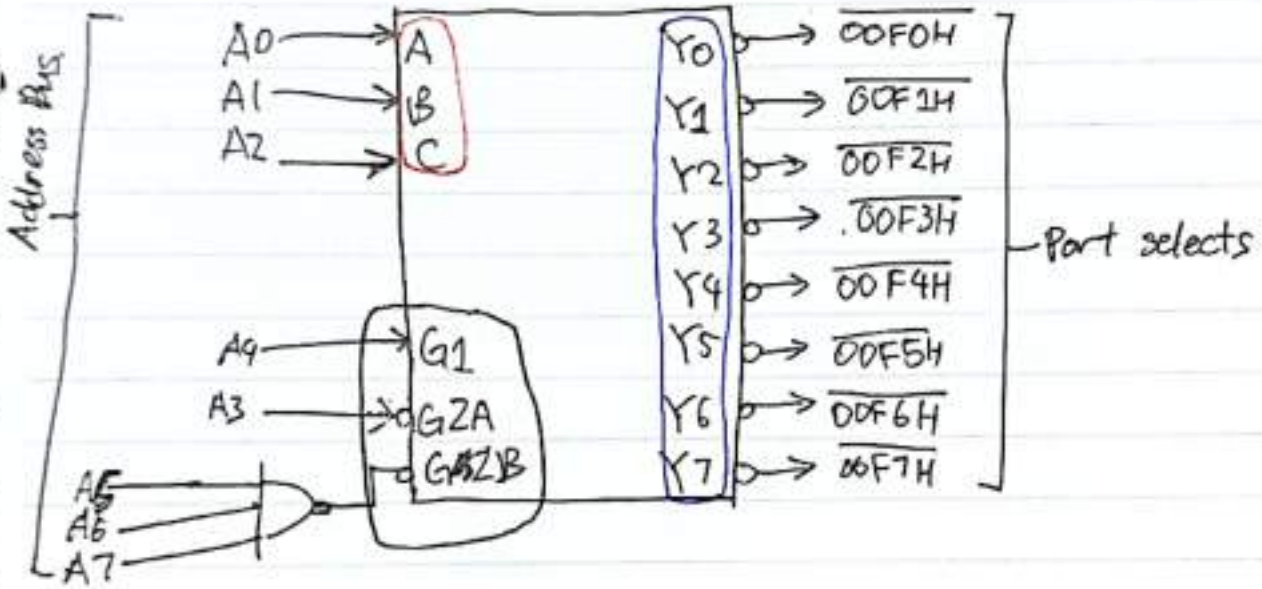
⑥ Data is strobed into device's data register when CPU removes /IO_WR.

⑦ The device having been strobed, immediately brings the signal _BUSY to "Low" to indicate that it is busy with new data.



I/O ADDRESS DECODER

The I/O Address Decoder is basically a recognition logic circuit which is designed to recognize specific I/O address that appear on the address lines. Once an I/O address is recognized, the decoder would activate an appropriate output line which, in turn, would select a corresponding I/O port.



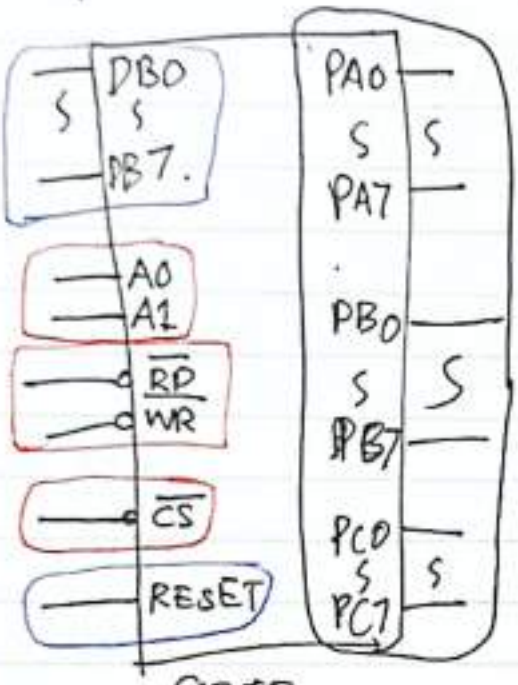
Y0 ~ Y7 → active low (normally stay at "Hi")

A, B, & C (A0 ~ A2) → responsible for selecting one out of the 8 outputs to be activated.
(Y0 ~ Y7)

In order to activate any one of the outputs.
G1, G2A, G2B (A4, A3, and A5 ~ A7) → needs to be active

LSI INTERFACING CHIPS.

There are many LSI interfacing chips that can be used for setting up parallel in/out ports on the PC.
These ICs come with necessary Tri-state buffer / data latches together with internal logics. To configure the data direction of the port pins as well as the operating modes.



8255
Parallel
Peripheral IC.

DB0 ~ DB7 → bidirectional data pins that connect to the system data bus. Data can be transfered to or fro, through these pins, between CPU and one of the 4 internal boxes in the 8255

A1 & A0 address pins → connected to system address bus: $(A_1, A_0) = 00$, select port A
$(A_1, A_0) = 01$, select port B
$(A_1, A_0) = 10$, -|- port C
$(A_1, A_0) = 11$, -|- Control register

(Chip select)
↓
CS → active Low → allows the 8255 to respond only to correct I/O address when it is being selected. Usually the appropriate out of an I/O address decoder is connected to this input

RESET is active high → normally connected to the system reset signal. → Function is to reset the 8255 when power-up, makes sure all the I/O lines started out as input lines and the operation mode is set to Mode 0.

$\overline{RD}$ & $\overline{WR}$ → recieve the IOR & IOW control signals from the system bus. These signals not only indicate the direction of transfer but also coordinate the timing of the transfer.

PA0 ~ PA7, PB0 ~ PB7, & PC0 ~ PC7.
The 8255 is a 40-pin IC with 3 I/O ports Port A (PA0 ~ PA7), Port B (PB0 ~ PB7) and Port C (PC0 ~ PC7) which gives a total of 24 I/O Lines.
These ports may be configured in the 3 different modes of operation.

#5.

No. CINTER-3
Date 22.5.22.

SWITCH

Switches → Limit switch
→ Toggle switch
→ Push button.

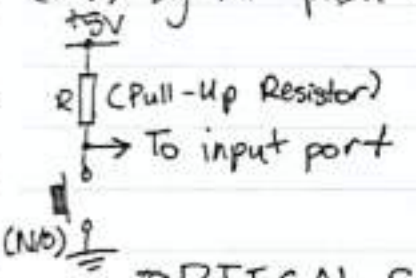
} are input devices which provide indication (signal) to the PC in terms of whether certain situation has taken place.

Types → mechanical, optical, etc.

MECHANICAL SWITCH

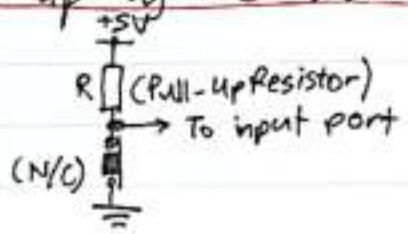
A - push to make connection.

Before pressed → "1" due to the pull-up resistor, when pressed, → "0", cuz it is connected to the gnd (0V) by the push button.



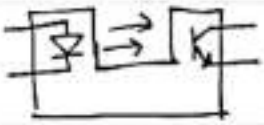
A - Push-to break connection.

Before pressed → "0" due to the normally closed push button. When pressed → "1", cuz it is released from gnd connection and is pulled up by the resistor.



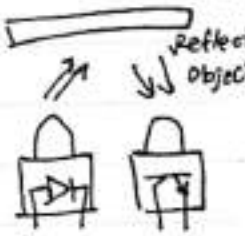
OPTICAL SWITCH.

Transmissive Sensor / Interupt Sensor



Contains an IR emitter & a photo detector f2f.
When a object is located between the emitter & detector, it interupt the beam, the amount of light reaching the detector is reduced.
This change in light energy or photo current is used to affect the app

Reflective sensor



Contains an IR emitter & a photo diode. next each other
When a reflective object is sensed the IR is reflected back to the photo-detector, the amount of light energy reaching the detector ↑, this is used as an input signal in the app.

Connect an optical sensor to an input port

Forward current I_F to the IR emitter diode.
$I_F \times R_1 + V_F = +5V$ ①

Pull-up resistor at output. so out can be "1" when light not sensed
② → $I_C \times R_2 + V_{CE(sat)} = +5V$

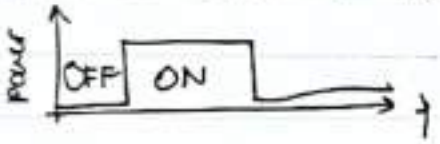
OUTPUT DEVICE / ACTUATORS.

Actuator → mechanism to introduce motion, or to clamp an object to prevent motion. (In engineering).
→ A subdivision transducer (In electronics)
→ These are devices which transform an input signal (mainly an electrical signal) into motion.

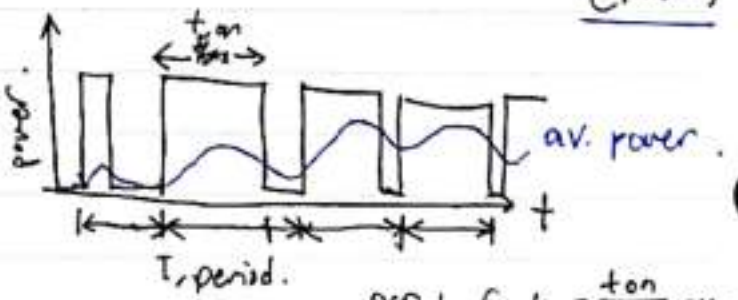
Driver Circuit for Output Device / Actuators.

The purpose of a drive is to deliver adequate power to the output device / actuator when required.
The mode of delivery could be by:

1. On/off control → a switch is used to turn on the power or turn off power to the device.

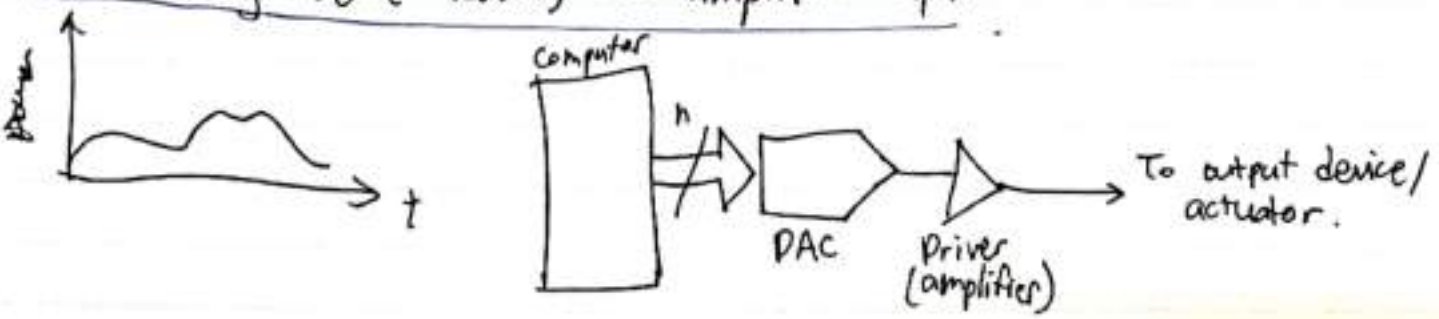


2. In modulated pulse width mode of control, power is delivered to the output device/actuator in pulse form, at a fixed period. The amount power can be controlled by adjusting the duty cycle of the output pulses.
Also known as Pulse Width Modulation (PWM)



$$\%\text{Duty Cycle} = \frac{t_{on}}{T} \times 100\%$$

3. Continuous Variable Control → power is delivered in an analog form, and the driver would normally be an amplifier.
The signal produced by the PC (our digital controller) is however, in digital form, thus the digital control output from the PC need to be converted into analog before feeding the amplifier input.



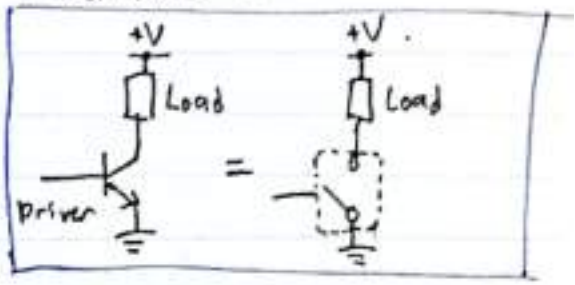
7.

No. CINER-3
Date 22.5.22

OUTPUT DEVICES / ACTUATOR.

Design of Driver Circuit

There are a number of factors needed to be considered when designing the driver circuit.
- Required V_s to the driver and output device / actuator.
- Max. I/P to be delivered (avg and/or instantaneous)
- Type of loading presented to the driver.
- Response time.
- Protection / Isolation.

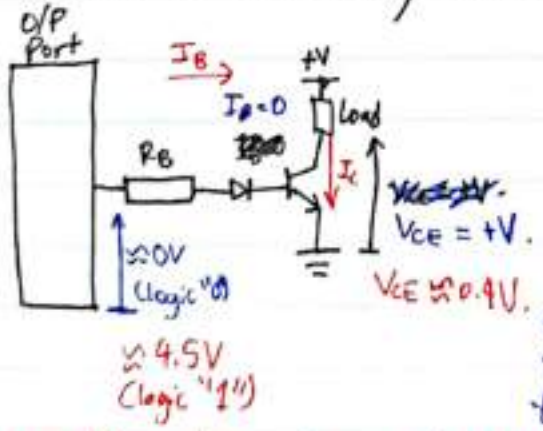


Using Bi-Polar Transistors as Drivers.

When an output device/ actuator uses DC supply, we may eithe use the On/ OFF or PWM control mode (depend on the app).
When using Bi-Polar transistors as a driver, we are using it as a switch, by turning the transistors ON or OFF, power is delivered or cut.

How to select the transistor?

1. Choose the max V_{CE} of the transistor larger than the V_{DC} power supply
2. Choose the max I_C of the transistor larger than the I_L.
   (once I_C is known, the I_B can be decided).



When the O/P port pin is at logic zero, it's not enough to overcome the V needed to turn on the diode and the base emitter junction of the transistor.
Therefore, $I_B = 0$.
Since there is no load current, the V across the load = 0. Thus $V_{CE} = V_{DC}$.

When the output from the O/P port pin is at logic high (>4V), I_B flows, the transistor is turned on. $h_{FE} \times I_B > I_{LOAD}$
the dc current gain of the transistor, β

Thus $I_B > I_{LOAD} / h_{FE}$
$V_o = V_{RB} + V_o + V_{BE}$
$V_o = I_B R_B + 0.7V + 0.7V$.
$V_o = I_B R_B + 1.4V$. $\rightarrow R_B = (V_o - 1.4)/I_B$.

Use opto-isolator (also known as an opto-coupler) to provide voltage isolation between the control port output pin and high DC supply (+V). The opto-isolator can provide voltage insolation up to a few kV.

No.

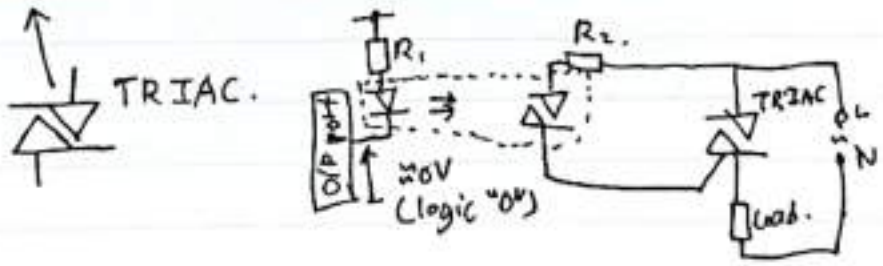
Date

Turning ON inductive Load, the inductance of the load will generate a back e.m.f. to oppose the turning on of the current. (Because of the opposing nature of the generated e.m.f, we called it back e.m.f).

Turning OFF Inductive Load. If we turn off the driver, load current will become 0 instantly, and energy in the inductance will have to be discharged instantly. This is not going to happen; because the inductor will oppose to this by generating a back e.m.f. to keep the current flowing at all cost!

Driving AC Loads.

To control power deliver to AC loads, we need diff types of drivers. This is cuz the I_L is alternating in 2 directions. 2 of the most common devices that can control current in both directions are triacs and solid state relays.



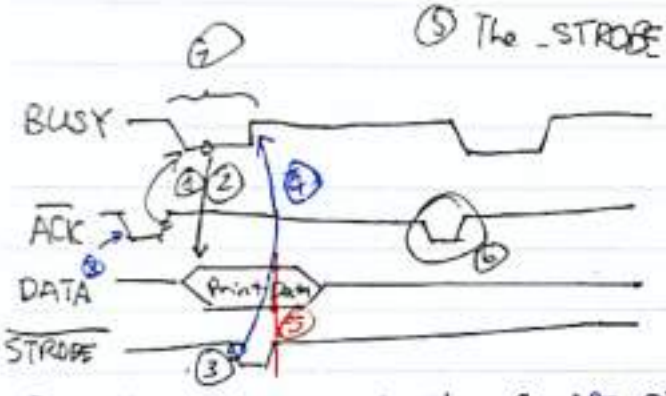
LINE POINTER (LPT) PORT & OTHERS.

Before PC, "Centronics" interface was the industry standard for connecting printer to a computer which used a connector with 36-contacts.
Today, the PC however uses a 25-pin D connector.
Meaning that some of the original (non-essential) signals are sacrificed and some of the contacts have also been assigned new functions.

Printer Data Transfer Protocol.

To send data to the printer, the following protocol is observed.

① PC monitors the BUSY signal to see if the printer is ready to receive
② If BUSY = "0", the PC place (8-bit) data on DB0~7.
③ Aft approx. 0.5 ms, PC assert _STROBE signal.
④ The _STROBE signal causes printer to assert its BUSY signal high
⑤ The _STROBE signal is returned to high aft approx. 0.5 ms



① Printer acknowledge the receipt of the prev. data and now ready for next data
② App prog. monitors BUSY line and found it low (not busy), so next print data is output to the data lines
③ Aft. ~5us, PC asserts the _STROBE signal.
④ the falling edge of the _STROBE signal causes printer to take the data and raise BUSY signal
⑤ The _STROBE signal is returned to "Hi" aft. ~5μs. Most printers choose this raising edge to take the data
⑥ Having taken the print data, printer acknowledge receipt of the data aft. some time when it's ready for next data.
⑦ Some printer drivers on the PC monitor Busy handshake signal for the sending of printer data, this is of course under prog control, ie polling.
⑧ Others use _ACK for interrupt-driven printing functions.

Using Parallel Port for I/O.

To use the parallel port for I/O interfacing, there are registers being used, there are 3 altogether for us to manipulate.

- Data register (Base address) → holds 8-bit data DB0 ~ DB7.
- Control register (Base address +2) → Control signals at control lines. (may also use as output data lines)
- Status register (Base address +1) → reflects the status of external devices and the handshaking (may be use as input)

PARALLEL PORT OPTIONS..

There is no standard method for configuring a parallel port on the PC. Some ports use jumper blocks or switches to select diff. options such as base address, IRQ lvl. and DMA channel. Other allows software config. using the ~~diff~~ utility disk provided. A port on the motherbord may hv config options in the CMOS setup.

In the case of a MultiMode Port, in addition to the config. options mentioned above, users also need to select a port type to emulate:

- Standard Parallel Port: based on centronic printer interface, 8 bit in output operation, 4 bit parallel input.
- PS/2 Port: introduced by IBM, supports parallel output operations, and simple bi-directional data transfer. 8-bit out, 4bit in.
- Enhanced Parallel Port: Bi-directional, can in/out a byte of data in 1 cycle of the ISA expansion bus including handshaking, compared to 4 cycles. It can also switch direction quickly. It can transfer at ISA bus speed, cuz of buffer and support for DMA transfer & data compression.

Addressing the Parallel Port

The PC has some parallel port support firmware built into its BIOS. Basic Input Output Service
a set of program routines tht perform many common tasks.

- When PC powers on, a BIOS routine automatically test for parallel ports at each of the 3 addresses in order → 03BCH, 0378H and 0278H.
- To determine the existance of a port, the BIOS simply sends a data ~~type~~ byte to the port and then reads back from the port what it sent. If the read-back tallies, the port exists.
- The routine stores the port addresses in the BIOS variables data area. To select LPT1 ~ LPT3.

Direct I/O using the Parallel Port.

Getting & sending data directly to and from the parallel port registers allow us to hv complete control over the parallel port signals. However, unlike other methods, this method does not automatically generate handshaking or control signals.

OTHER PARALLEL INTERFACES ON THE PC

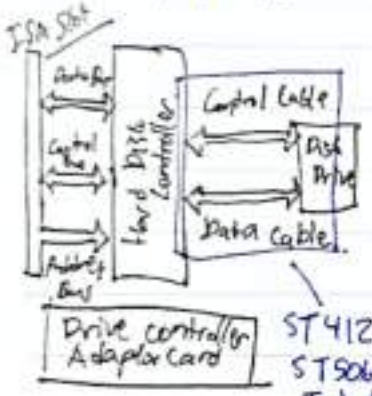
There are 3 other parallel interfaces on the PC, including:

① IDE (Intelligent Drive Electronics)
② SCSI (Small Computer Systems Interface)
③ PCMCIA (Personal Computer Memory Card International Association).

These are ~~made for~~ developed for attachment of secondary storage devices such as hard disk drives, and optical disk drives to the PC.

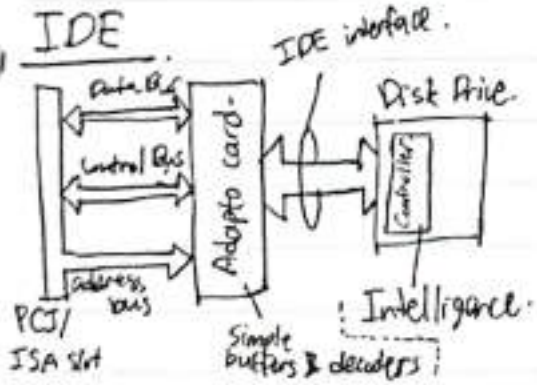
Seagate technology → ST506 Hard Disk Interface (5MB) in 1980 & ST412 (10MB) in 1981.

The interface designs were used by IBM for PC and become known as the ST412/ST506 interface.



The disk drive itself had only those electronic parts needed to drive the motors and gates of the disk drive.
The more extensive control is done by the disk controller on an adaptor card.
Eg. to rd a sector on a particular track, the controller would perform a head seek, read encoded signals, separate the data & clock signal, transfer data into random access memory with the DMA scheme, and so forth.

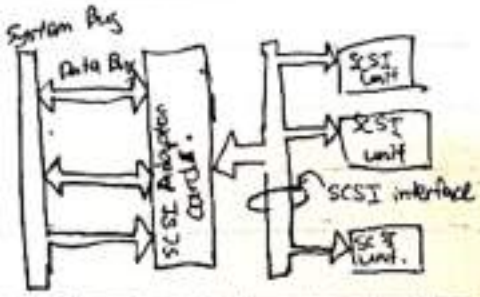
IDE



IDE is a further developement of the ST 412 /ST506 interface with the device controller being integrated into the disk drive and the adaptor card merely consists of simple electronics.

SCSI

- SCSI is v. flexible & powerful alt way to connect hard disks and other devices such as CD ROM drives, removable disk drives and scanners to a PC. For connection to a PC, an adaptor card is required.
- The original 8-bit SCSI interface defined a bus between max of 8 units.
- The adaptor card itself is also a SCSI unit, therefore connections are available ~~only~~ for 7 SCSI units only.



No.

Date

SCSI

SCSI interface serves for data exchange among the SCSI units, and up to 2 units may be active at any one time. Data can be exchanged b/w 2 units, other than the adapter, w/o the involvement of the CPU at all. This is done with one of the unit acting as the bus master initiating the data transfer, and the other unit acting as the target.
There are several diff. variants of SCSI standards:
- SCSI, SCSI II, Fast SCSI, Wide SCSI, Ultra SCSI, and more. The original 8-bit SCSI interface uses 25-pin connectors. With the 32-bit & 64-bit bus systems, the width of the SCSI bus has also been increased to 16 as well as 32-bits

PCMCIA

- With the growth of PCs, to deliver higher performance, this generates the need for smaller, lighter & less-power-hungry peripheral devices.
- Founded in 1989, and the first PCMCIA standard was introduced in 1990.
- All PC cards hv the same planar or outline dimensions (54mm x 85mm).

(Standard PC card diagram). Thickness of each card type in the substrate area is: Type 1 cards = 3.3mm
Type 2 cards = 5.0mm
Type 3 cards = 10.5mm.

- Regardless of the functionality and the type of the PC card, PCMCIA systems are designed to allow automatic detection & configuration when it's inserted either before or aft the system is powered up.
- The configuration option information of a PC card is kept within the card itself in a non volatile memory area known as the Configuration Information Structure (CIS). When a PC card is inserted (hot/cold), the PC enabler (a driver running on the system whenever the PC is powered up) would detect the insertion of the PC card and read the CIS of the card. The Enabler would then configure the PC card for accessing.

This following image shows the connection of the PC card to the PC or laptop, where the host bus adapter bridges the system expansion bus to the PC card socket or sockets.

SERIAL INPUT / OUTPUT INTERFACING

- Communications between ~~comp~~ users can be done via parallel, but when the distance is far between the 2 communicating parties, parallel is impractical as cost ↑ since all data bit is transfered at the same time, so need more wire/cables

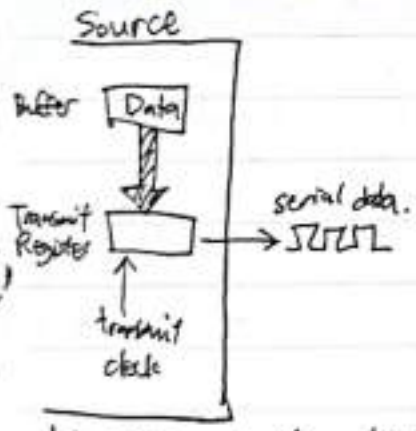
  Data skew → they might start at the same time but due to the line characteristics and ~~the~~ propagation delay, they won't arrive at the same time

* The solution to long distance data transfer is to employ only one ~~transm~~ transmission path, where cost is minimum, and since the data bit are transfered over a single channel, data skew don't exist any longer.

SERIAL TRANSMISSION FORMATS

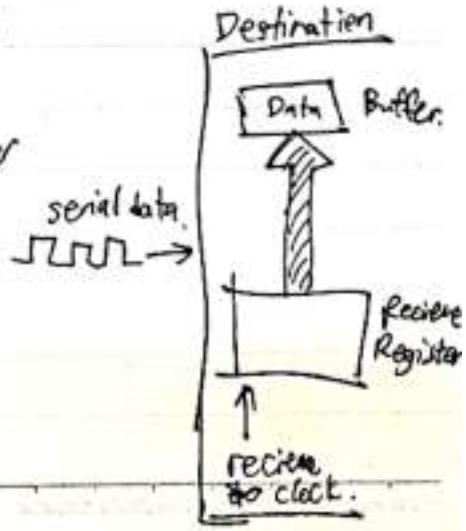
To send data out from the ~~trans~~ source:

- Data must be loaded from a holding buffer into a transmit register (shift register)
- Data is then shifted out of the transmit register one bit at a time by a transmit clock at regular intervals.



It should be apparent that the rate of the data bits being transmitted depends on the transmit clock rate. The transmit data may have to undergo certain transformation depending on the medium of the channel

Likewise, at the destination, the recieved signals need to be demodulated back to the digital form. The serial data arriving is shifted into a ~~re~~ recieve register by the recieve clock one bit at a time. When the recieve register is filled up, its data is transfered into a recieve buffer.



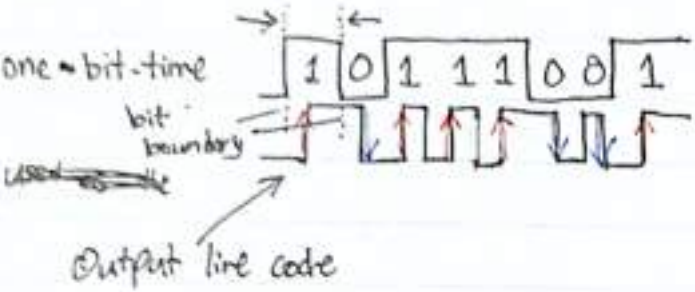
It should be obvious that the recieve clock at the destination must be in step with the data bits that arrive, in order to shift them into the recieve register correctly.

No.

Date

SYNCHRONOUS TRANSMISSION.

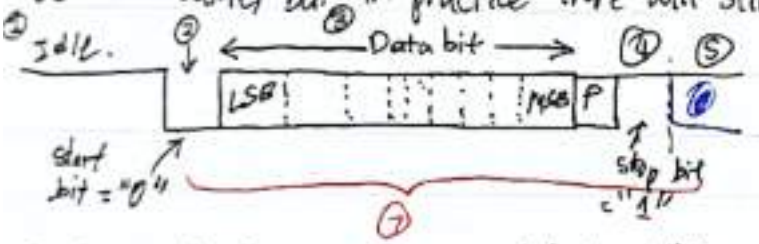
The transmit clock is embedded in the data stream using an encoding scheme. Example, one of them is Manchester Encoding, was used for storing data on the magnetic drums of the Manchester Mark 1 computer.



The output line code has positive transitions at the middle of the bit when the data bit is a "1". The output line code has negative transitions when the data bit is "0".
This arrangement ensures at least a transitions for every data bit transmitted, which can be used as the clock ref. at the dest. such that the reciever clock & the transmit clk can be exactly in step, i.e. synchronized.

ASYNCHRONOUS TRANSMISSION.

In the transmit clock is not embedded to the data stream and hence the transmit clk and reciever clk are independent to each other. Though the clk are arranged to be the same, but in practice there will still be diff between the clk (not synchronized).



① When the transmission channel is on but no data are being transmitted, the line signal remain idle at logic "1".
② At the beginning of transmission, a Start bit (logic "0") is sent.
③ At the beginning of transmission, a Start bit (logic "0") is sent following by either 6, 7, or 8 data bits and then followed by an optional Parity (P) bit.
④ A Stop bit (logic = "1") is appended immediately at the end.
⑤ After the stop bit, the line signal either remains at "1" (idle) - no immediate data to send
⑥ ~~After the stop bit~~ or go to "0" - start of next data word.
⑦ The start and stop bits tgt serve as a frame that holds a chunk of data bits. Typically an ASCII char. Therefor, the frame is also known as a char frame.

If we allow the transmission to continue for some time, both the transmit clock and reciever would be totally out of steps since their clock rates are not excatly the same.

① With the start bit serving as the indicator for the start of another chunk of (char)
② It's also used by the reciever clk gen. to re-sync its output with the transmit clk
③ With the detection of the mark-to-space transition of the START bit, at half a bit-time delay, the Rx Clk is started and the first clock appears at the center of the START bit
④ If the Rx clk freq. is excatly the same as the Tx's, then the Rx clk would always clk the incoming data bit at the center
⑤ ~~In the~~ IRL, the Rx Clk freq. is nvr excatly the same as Tx's.
⑥ Hence, subsequently Rx clk pulses will gradually drift away from the center of the data bits.
⑦ Despite so, since the total number bits in one frame is not more than 11 bits, the transmit clk and the reciever clk, shld only slightly out of sync if ~~the~~ their clk rate diff only +/- 5%, at the end of a frame, and data would still be clk into the reciever register properly by the reciever clk.

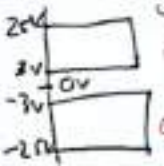
No. CINTER -5
Date 15 . 8 . 22.

SERIAL INTERFACE STANDARDS

There are a number of serial interface standards such as RS-232 C/S, Firewire, USB 2.0 as well as a number of others.
RS-232 interface standard is the standard use for legacy serial port(s) on the PC, also known as the "Communication Port(s)" or "Com Port" developed by Electronic Industries Association (EIA) of USA.
It has gone through changes till RS-232D → split into 2 groups
- Data: transmit data, TXD.
- Recieve data, RXD.
- Control: which consists of signals needed for cons protocols.

DTR = Data Transmit Ready
DSR → Data Set Ready
RI → Ring Indicator

CD → Carrier Detect
RTS → Request to send.
CTS → Clear to send.



Control → Control "ON"
Control "OFF"
Data → Logic "0"
Logic "1"

The signal volt. lvl is diff from TTL or CMOS logic circuits.
Also, the volt lvl for data are opposite to the control signals.

CONNECTORS

0-25 Pin Connector

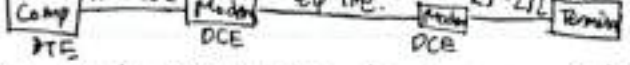
D-9 connector.

25-Pin	9-Pin.	Signal	Direction
1		Prot Gnd	–
2	3	TXD	Out
3	2	RXD	In
4	7	RTS	Out
5	8	CTS	In
6	6	DSR	In
7	5	GND	–
8	1	CD	In
20	4	DTR	Out
22.	9	RI	In.

DTE → Data Terminal Equipment, serial data original from or terminate at (generate data or consume them. eg. comp, printer, etc.

DCE → Data Communication Equipment, responsible for the transmitting and recieving data through some sort of communication channel eg modems, multiplexers and concentrators.

DTE to DCE



→ use 25-pin.

~~① DCE turns to DTR~~
① DTE turns to DTR to tell DCE it is ready to communicate.
② DCE turns to DSR to tell DTE its ready to communicate.
③ DTE has data to send to other DTE, so it turns to RTS. → Dial modem on other end to reply with carrier sign then connection established. after, DCE asserts CTS.
④ On seeing CTS from DCE, DTE sends data to TXD.
⑤ DCE transforms data from DTE into analog signal & send them through telp. line.

Answering side (assume DTR and DSR are on):
① It turn on RI to tell DTE hv incoming call
② DTE ready → DTR is alr on PCE ans the call by putting the phone line then sends carrier signal through phone line to tell the initiating PCE tht the ans side is ready
③ The carrier is turned off aft a certain time-out.
④ The answering side now awaits the calling side to transmit, first the carrier and then the modulated signal down the phone line.
⑤ On recieving modulated signal, DCE demodulates the analog signal obtains the digital data the demodulated data is passed to the DTE

No.

Date

DTE to DTE

Although RS-232 was designed to be used with a modem for a DTE to communicate with another DTE, there are times, the dist between the 2 may be v. close and therefore do not require the telp. line and modems between them.
But since they are both DTE, it's not possible to connect them via RS-232 pin for pin.
Special arrangement of the connecting cable have to be made in order to fool each DTE to think that it's connected to DCE. The cable is known as a null-modem cable since it replaces the modems at the 2 ends.
The simplest null-modem cable has only 3 wires: one each for the data transfer in the 2 directions and the 3rd wire is the common gnd.
In this arrangement, the software tht performs the data transfer simply assumes that both sides are always ready to recieve data and thus don't require handshaking protocol.

$\overline{DTR}$, $\overline{RTS}$, $\overline{OUT1}$, $\overline{OUT2}$ → Active lo out, according to Modem control register.

UNIVERSAL ASYNCHRONOUS RECEIVER AND TRANSMITTER.

Since the PC manipulates data in parallel format within itself, in order to transmit data serially, parallel-to-serial transformation has to be carried out by the serial port.
Likewise, serial data recieved by the serial port has to be converted back to parallel format. Furthermore, control signals have to be generated and monitored for the serial port to work with a modem in a long-distance data communication.
Thus, the functions performed by the serial port are many & complex. To ease the hardware design of the serial port UART 16450 or UART 16550 is used.

UART 16450

INTR → Hi if int and bit enabled in the int enable register

$\overline{RI}$, $\overline{DCD}$, $\overline{DSR}$, $\overline{CTS}$ → Modem status Active Lo input

BAUDOT → clk out for transmitter baud rate, usually connected to RCLK inp such that the receiver and transmitter operate at same baud rate

D0 ~ D7 → Data pins to data bus

A0 ~ A2 → Address Pin to connect to system.

XTAL 1/2 → connect to crystal or external oscilator of 1.8432MHz

$\overline{AS}$ → Strobe address bus signals A0 ~ A2 & chip sel sig. CS0 ~ CS1 and $\overline{CS2}$ to the UART.

DINS → for the I/O Read control signal input IOR.

DOUTS → for the I/O write control signal input

DDIS → driver disable goes low whenever the CPU is reading data from UART

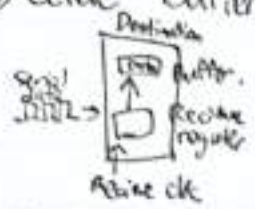
CS0/CS1 & $\overline{CS2}$ → only select it all are active

SIN → serial data input

SOUT → serial data output

UART 16450 Register.

① Recieve Buffer Register (RBR).

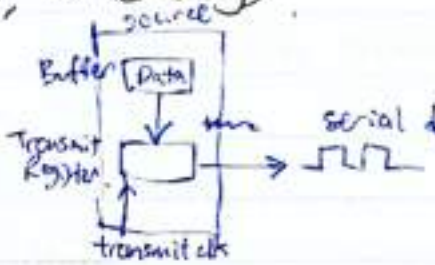


Whenever the recieve register recieve a full frame of data the data is transferred to the RBR.

(@ Base address, DLAB = 0, write-only)

② Transmitter Holding Register (THR)

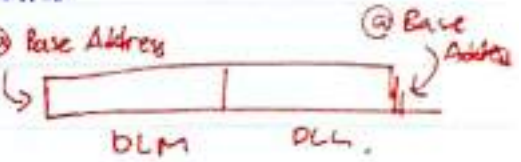
The data to be transferred is loaded to the THR (8-bits).



(@ Base Address, DLAB=0, write-only)

③ Divisor Latch (LSB), & Divisor Latch (MSB).

use a crystal at XTAL 1/2 → produce clk freq. 1.8432MHz. This freq is divided by 16 internally to give 115200Hz. If lower clk, set the division ratio in the 16-bit divisor



LSB : (@ Base Address, DLAB = 1, read/write), MSB : (@ Base Address + 1, ——1——)

④ Interupt Enable Register (IER) (@ Base Address, DLAB=0, read/write)

7	6	5	4	3	2	1	0
0	0	0	0				

0 → recieve data avail, 1 → Transmitter holding register empty
2 → Reciever line Status Regis. chg. 3 → Modem Status Regis. chg.

⑤ Interrupt Identification Register (IIR) (@ Base Address + 2, read/write)

to provide min software overhead during data transfer, the UART prioritize int into 4 lvls and records these in the IIR.

⑥ Line Control Register (LCR) (@ Base Address + 3, write-only)

This register allows the format of the asych data coms to be specified & also provides the access to the DLL & DLM. via the DLAB (Divisor Latch Access Bit).

⑦ Modem Control Register (MCR) (@ Base Address + 4, read/write)

7	6	5	4	3	2	1	0
0	0	0					

0 → Data terminal Ready, 1 → Req. to send. 2 → OUT1.
3 → OUT2, 4 → Loop-back Mode

⑧ Line Status Register (LSR) (@ Base Address + 5, read/write)

This register provides status information concerning data transfer, for read operations only. Although it can be written to (during factory testing), but writing to this register is not recommended during normal operations.

7	6	5	4	3	2	1	0
0							

6 → Recieve Data Ready (reset by reading RBR)
5 → Overrun Error (reset aft regis read)
4 → Parity Error (—1—), 2 → Break Int Indicator (—1—), 0 → Transmitter Empty (reset when data at THR & TSR
3 → Framing Error (—1—), 1 → Transmitter Holding register Empty (reset by writing data to THR)

⑨ Modem Status Register (MSR) (@ Base Address + 6, read-only).

Provides current state of control lines from the modem & in addition, indicates whether these control lines hv chged status since the last rd from CPU.

7	6	5	4	3	2	1	0

0 → ΔCTS (reset aft the MSR is rd)
1 → ΔDSR (—1—), 3 → DCD (—1—), 5 → DSR, 7 → DCD.
2 → TERI (—1—), 4 → CTS, 6 → RI

⑩ Scratchpad Register (SCR) (@ Base Address + 7, read/write).

The register don't control the UART in any way, It's just a register for holding data temporarily.

No.

Date

LOGIC STRUCTURE OF THE SERIAL PORT.

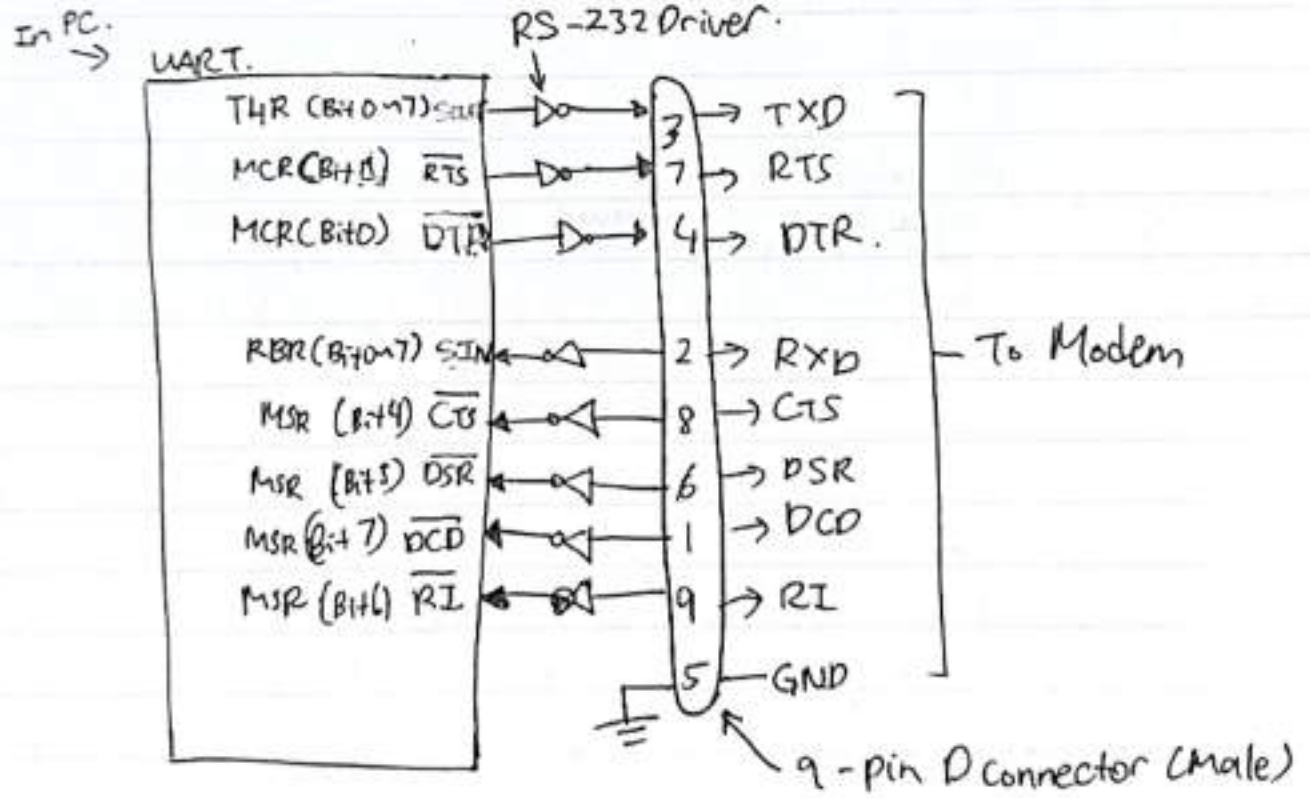
Signals at RS-232 interface operate at diff. volt. lvl. from normal digital logic sig. lvl. inside the PC's UART.

PC → TTL (Transistor-Transistor Logic) → Logic "0" = ~0V, Logic "1" = ~5V.

RS-232 → Logic "0" = +3V ~ +25V, Logic "1" = -3V ~ -25V.

The status & control signal for RS-232 operate in reverse logic as the serial data signal.
Logic "1" = +3V ~ +25V, Logic "0" = -3V ~ -25V.

∴ Hence, we need voltage translators to translate the incoming RS-232 signals with appropriate logic inversion into TTL logic lvls for the UART. We also need another type of to tranlate the TTL sig. lvl. from the UART to RS-232 volt lvl for the outgoing signals.



USB.

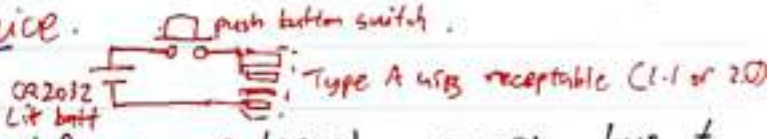
Universal Serial Bus is a set of interface specifications for high speed wired coms between electronic system peripherals and device with PC/computer.

Traditional way → parallel printer port. → can connect a few LEDs & code in visual C++.

USB → more complex → harder to implement the parallel printer port.
→ not all comp hv PPP so use USB.

Connect USB cable from comp to a USB interface board will not start working after soldering. Before computer can detect, need to load a microcontroller program code (usually a small hex file). into the PIC 18F4550 (microcontroller). Computer will then detect it as a new external plug-and-play hardware, then install the driver in computer.

USB is the most successful interconnection device.



CONCEPTS OF USB

The major goal of USB was to define an external expansion bus to add peripherals to a PC. in an easy & simple manner.

The new external expansion architecture would offer:

① PC host controller hardware & software ② Robust connections & cable assembly
③ Peripheral friendly master-slave protocol ④ Expandable through multi-port hub

USB offer simple connectivity. It eliminates the mix of diff. connectors for diff devices like printers, keyboard, mice, and other peripherals.

Allows many peripherals to be connected using a single standardized interface socket.

It support all kind of data from slow mouse input to digital audio and compressed video

USB allows hot-swapping, means the devices can be plugged and unplugged without rebooting the computer or turning of the device. This means that the USB can be plugged in and everything configure automatically, and they can unplug the cable. The host will detect its absence and automatically unload the driver → makes USB "plug-and-play" interface

USB sends data in serial (parallel data is serialized the deserialized)

① Low cost, ② Expandibity, ③ Auto-Configuration, ④ Hot-plugging ⑤ Outstanding performance

It provides power to the bus enabling many peripherals to operate without the added need for an AC power adaptor.

No.

Date

USB VERSIONS.

Jan 1996 → USB 1.0 → transfering 12 Mbps, support up to 127 devices

Sep 1998 → USB 1.1 → help rectify adoption problems tht occured with 1.0, mostly those relating with hubs

Apr 2000 → end of 2001 → USB 2.0 → released 2000, standarized 2001 → Standarization of new-device-specs made backward compatibility possible.
↖ Hi-speed USB.

Aug 2008 → USB 3.0 → by intel & partners → latest version → super-speed USB at 4.8 Gbit/s data transfer (600 MB/s).

USB SYSTEM.

made of host, multiple numbers of USB ports, and multiple peripheral devices connected in a tiered-star topology

The number of USB port can expand, the hubs can be included in tiers to branch into a tree up to 5-tier lvls.

The USB is actually an addressable bus system, with a 7-bit address code. So it can support to 127 diff devices or nodes at once. But only 1 host, the PC itself. So a PC and its peripherals connected via USB, forms a star local area network (LAN).

USB can have a number of other nodes connected to it in daisy-chain fashion. to form the hub for a mini-star sub-network.

On a USB hub device, the single port used to connect to the host PC either directly or via another hub is known as the upstream port, while the ports used for connecting other devices to the USB are downstream ports.

USB specification recognizes 2 kinds of peripherals:
- stand alone (single function units, like a mouse).
- compound devices (like video camera with separate audio processor).

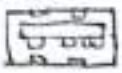
The logical channel connection host to peripheral-end is called pipes in USB. A USB device can hv 16 pipes coming in to the host controller and 16 going out the controller.

The pipes are unidirectional. Each interface is associated with single device function and is formed by grouping endpoints.

USB CONNECTORS & THE POWER SUPPLY

USB connect to machiere, if its a new device, the OS auto-detects it and asks for driver disk. If driver is installed, the comp activates it and starts talking to it.

2 standard cable and connectors,

A →  B →  A USB cable have A on one end and B on the other end.

USB A socket on the computer.

Inside the USB cable there are 2 wires tht supply the power to the peripherals -- + 5V (red) and gnd (braun) -- and a twisted pair (yellow & blue) of wires to carry the data

Type - C connector.

- Use for more than data transfer, can store data, and can have enough power for laptops.

HOW THEY COMMUNICATE?

When a USB peripheral device is first attached to the network, a process called enumeration is started.

Host communicates with the device to learn its identity & discover which device driver is required, it starts by sending a reset signal to the newly connected USB device. / the process is initiated both when the host is powered up & a device is connected. / removed from the network.

USB DESCRIPTORS.

All USB devices have a hierarchy of decriptors, to describe to the host information such as what the device is, who makes it, USB version, it supports, ways to be configured, no of end point, etc

Common USB decriptors: - Device descriptor - Interface descr - Config descriptors - En & point descr - String descpricter.

The host controller polls the bus for traffic (usually in round-robin fashion), so no USB device can transfer any data on the bus without an explicit request from the host controller. → list are 4 data transfer types. USB supp

① Control → transfer xchange config, setup command into between device & host.

② Isochronous → transfer is used by critical, streaming device such as speaker & vid cam.

③ Bulk → transfer is used by printer & scanners,

④ Interupt → transfer is used by peripherals exchanging small amt of data tht need immediate action. Devices like a mouse or a keyboard comes in this category

No.

Date

USB PACKETS AND FORMATS.

→ Serial → start from LSB → transfer in form of packets of data, sent back & front b/w the host and peripheral devices.

→ Each USB data transfer consist of:

* Token Packet → Header defining what it expects to follow
* Optional Data Packet → Containing the payload.
* Status Packet → used to acknowledge transactions & to provide a means of error connection.

FURTHUR USB.

One of the biggest problem with USB. is that it is host controlled. If we switch off a USB host, nth else work. USB don't support peer to peer coms. Eg. Cam can download data to PC but can't connect to USB printers.

To combat these problems, a standard was created to USB 2.0

USB 2.0 USB On-The-GO (OTG) was created in 2002.

To supplement to the USB 2.0. specs USB OTG defines dual-role device, acts either a host or peripheral & can connect to a PC or other portable devices through the same connector.

Min & Micro USB → The OTG specs hv 2 additional connectors. Min A/B connectors

A dual-role device is required to be able to detect whether a mini A or Mini-B is inserted. The Micro USB connector is intro on Jan 2007. main intention was to replace Mini-USB plugs in new phones & PDAs.

Serial RS-232 ports fast erd direct mapping to mem. & worked at high prio. Data practically flowed directly to & from the software that was accessing the serial port. USB devices have high throughput, but they use a shared data bus, USB driver is needed, must be installed and working, need to identify the chip in cable & locate driver on internet.

Challenges in USB interfacing

DRIVER.

The struct usb_device_id is devised with

- __u16 idVendor → The USB vendor ID for the device. This num is assigned by the USB forum to its members & can't be made up by anyone else.
- __u16 idProduct → The USB product ID for the device. All vendors that hv a vendor ID assigned to them can manage their product IDs however they choose to.

To get vendor ID must go to USB Implementers Forum, Inc.

USB bit wacker → small board with a command interpreter for basic input & output control.

No. CINTER -7
Date 16 . 8 . 22

OTHER DEVICES.

STEPPER MOTORS.

Or step motor → used where high deg. of positional control is required. This true in computer & instrumentation applications, whr they are used in printers robots & HDD eg. with a series of pulses, the motor will rotate a fixed no. of steps. By chging the order of app. the motor can be reversed. To move at a given speed, the pulse can be issued at given rate.

3 type of stepper motors in use:

1. Permanent Magnet Stepper Motor (PM) → hv a permanent magnet on the rotor. The rotor resides in a casing that physically surrounds it. In the casing, hv several coils of wire called the stator
2. Variable Reluctance Stepper Motor (VR)
3. Hybird Stepper Motor.

Step Motor terms.

- Step size → how much the motor rotates in 1 step
- Phase → No of set of wire winding on the stator. (stator coils are wired usually 4)
  → determines the no. of wires required. Another way to see phase is the num of electrical signals which hv to applied to the motor for one electrical cycle → the num. of electrical signals which hv to be applied to the motor for 1 electrical cycle.
- Wire Winding → By applying an electric signal to it, it'll generate a magnetic field, which will either attract or repel the rotor → so the motor turns

& The relation b/w phase & step size depends on motor construction

MOTOR SCHEMATICS. (PM stepper motor)

Use electromagnets and rotors turning in conjunction with opp poles of the energized electromagnets.

1. Let phase A be actuated first as a SOUTH pole. The rotor will align itself so the poles attract.
2. Aft, phase A is turned off phas B is energised, Rotor will turn by 90°. The magnetic force produced by the stator coil pulls the rotor to align with it
3. Now, phase A & B are off, phase C is on. The rotor moves another 90° and so on for D. (real world application will not be 100% 90°)

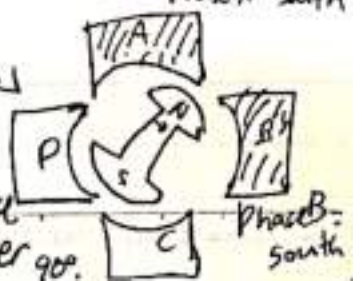
A microcontroller can control the current in the phase easily by sending binary valued voltages to it. By during so, it controls the position. By timing the signal, it will control the speed & acceleration.

Full step → entire step 90° → 1 phase

Half step → 2 phase at once → move b/w 2 phase.

If A & B is on, it will rest at the halfway position

Although twice the current flows in a half step, torque increase 1.414 times only ($\sqrt{2}$) Then phase B & C are on to move another 90°.



No.

Date

MOTOR DRIVING.

Control movement & torque produced both by activating the proper phases & by the motor electrical connections.

So far, connect the stator coil so coil flows one way through the transistor

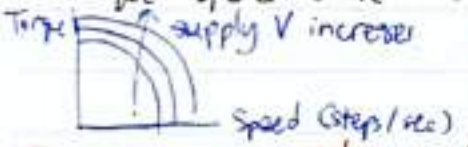
Bipolar Drive → by connecting up the coils & transistor so current can flow both ways (e.g. full step → A as South, C as North, the B & D are on, and soon). unipolar drive

Control issues:

- Micro-stepping - Not supplying full current → don't send full current, rotor stops in between the position of 2 phase
- Moving direction → ABCD or DCBA → Cannot drive a motor at an arbitrary speed & acceleration. If the rate at which coils are energized is too high, motor may not rotate in sync with the pulse rate, & may stop moving.
- Speed → The time delay b/w 2 phase energizing signal → at the speed, the motor may vibrate / even rotate in reverse by itself. ↳ shorter delay b/w phase ↓ faster speed
- High speed problem → loss of synchronisation & resonance.

TORQUE SPEED CURVE / TRASLATOR & INDEXERS.

- Torque speed curve → characteristic of stepper motor. → phase voltage ∝ torque it can produce



Motors must be controlled simulatneously additionally piece of hardware are used → translator & indexers.

Translators → motor sends steps. motor control speed. Direction, full or half is done by setting some pins.

Indexers → accepts a piece of commands from the processor specifying start/stop. acceleration / deceleration rates, slew rate & final pos, it generates the req. stepping rates to achieve the speed profile & informs the processor when final pos is reached; by using an int or by setting a bit in status regis. The processor doesn't send out timing signals.

CALCULATIONS EXAMPLE.

- A stepper motor rotates 1.8 deg per step, what step rate is needed to make it rotate at 10 rev. per min (rpm).

1. Assume the prog. execution time is negligible.
2. Access to a routine called Delay 1ms → provides a 1-ms delay.
3. Full step mode.

Solution.

① Convert rpm to steps per secs → $10\,rpm = \frac{360°}{min} \times 10 = \frac{3600}{60}°$ per sec $= 60°$

② Since one step is 1.8°, we need to issue → $\frac{60°}{1.8°} = 33.33$ steps per second

or $\frac{1}{33.33}$ secs / step $= 0.03$

or 30 ms / step

CINTER - 8.
Date 16 . 8 . 22.

REAL-WORLD DATA ACQUISITION.

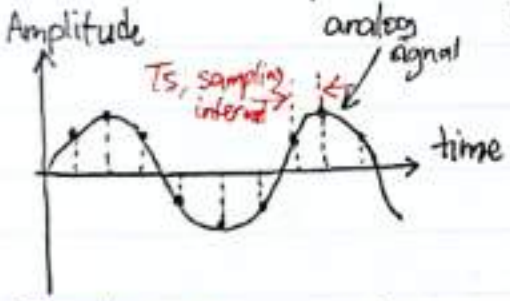
ANALOG VS. DIGITAL.

Real world → analog → continuos amplitude & continuous time signal
→ amp & time can be any value.

Digital → discrete amplitude & time → only logic "1" or "0".
only 2 discrete-amplitude signals.

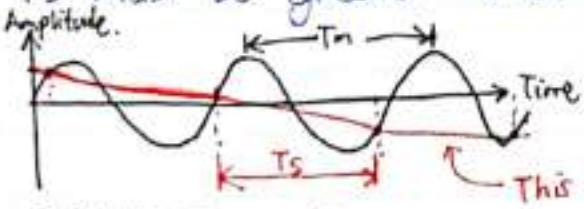
SAMPLING ANALOG SAMPLING

Since an analog signal is a signal whose amplitude varies with continuous time, it is not possible to obtain the amplitude values for all instance of time. This would result in an infinite num. of values to obtain even if it were possible, no computer can process an infinite amt of values.
Therefore, only samples of the analog signal are taken at certain time, that is, discrete time. In order to retain certain timing info abt the signal, samples are taken at regular time interval. This process is known as sampling process.



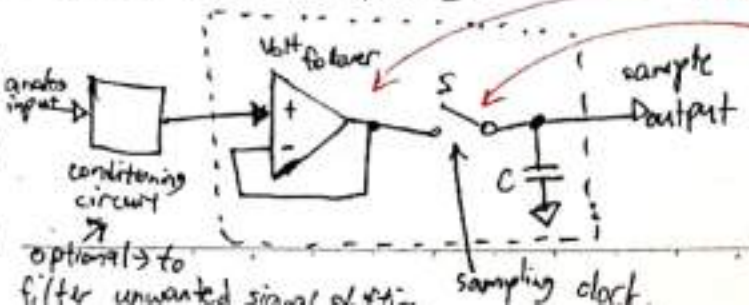
samples of the analog signal had been taken at a regular time interval of Ts.
It should be obvious that for a given duration of the signal. The number of samples taken is directly dependent on the sampling freq.

Sampling therem states:
In order to retain the complete info of the signal, the sampling freq, fs must be greater than 2x the highest signal freq. ie $f_s > 2f_m$.



Samples don't seem to represent the orginal sin signal but one whose freq appears to be much lower.
This is known as the alias signal

SAMPLE & HOLD

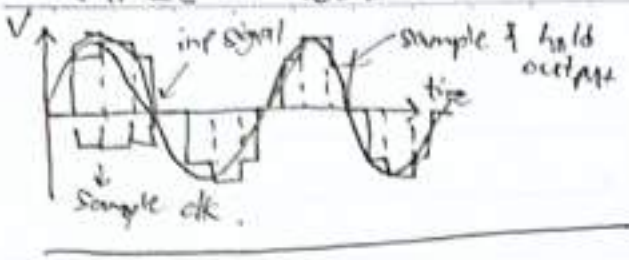


Give Vout same as Vin, S closes.
when clock is on, current flow charging C (C is v small) & charge it instantly, with the V same as the one going in, the S opens

No.

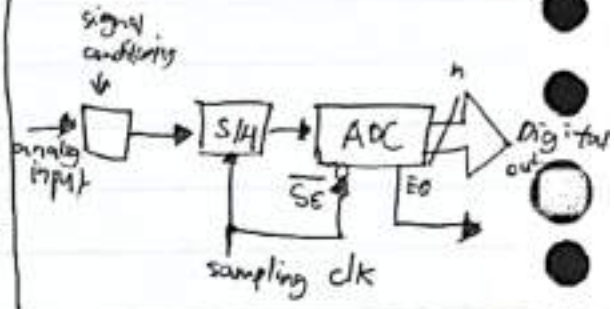
Date

SAMPLE & HOLD



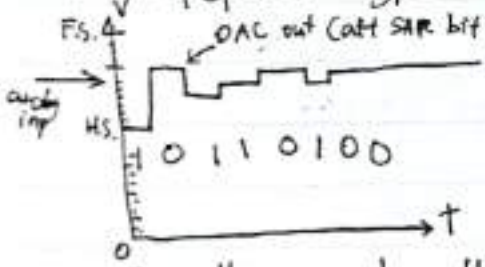
The reason for holding the sampled signal voltage constant until the nxt sampling is for the circuit aft an ADC. to convert to digital signal.

At the end of each sampling clk pulse, the ADC is to start the conversion via the SC (Start conversion inp). As soon as the conversion is done, the corresponding digital value of the analog amplitude is placed at the data output of the ADC & indicating signal EOC (end of conversion) to tell the consumer data is ready.



ANALOG-TO-DIGITAL CONVERTERS (ADC)

There are 3 category of ADC designs → Feedback ADC → Integrating ADC
Most popular type → Successive Approximation ADC → Flash or parallel ADC



Takes in 8 clk pulses to complete a conversion no matter wht the external analog input voltage is, since ADC has 8-bit SAR.

In other words, the conversion time is dependent on the num. of bits the SAR has, and it always constant for given successive approx. ADC.
↑ successive approximation register.

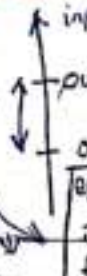
A/D CONVERTER CHARACTERISTICS.

There are a no. of ADC characteristics that deserve special attention in order to hv a better understanding abt using ADC. These are:
→ Resolution & Accuracy → Linearity → Scale or Gain Error → Monotocity

★ Resolution & Accuracy

resolution is the amnt of inp volt chg req to increment from one out code of the ADC to the next adjacent code. An n-bit ADC can resolve one part in 2^n. The smallest increment then 2^{-n}, or one LSB. Thus, resolution can be expressed in percentage of full scale, or in number of ADC bits

Accuracy is defined as the absolute error incurred in the measurement of a signal.



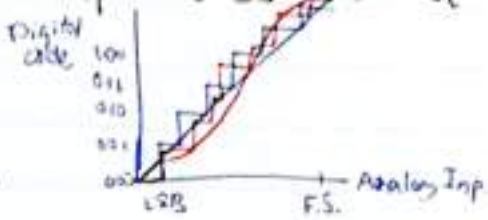
accuracy also be ±2%

eg. 12 bit → 4096
$\frac{1}{4096} = 0.0244\%$ if full scale is 10V then resolution is 2.44mV

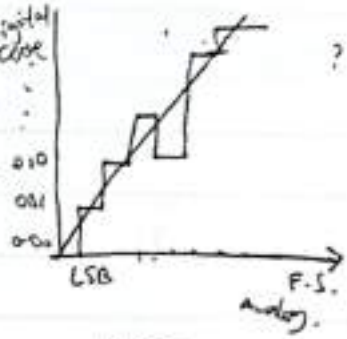
A/D CONVERTER CHARACTERISTICS

☆ Linearity → describe the departure of the output codes from a linear transfer function curve.

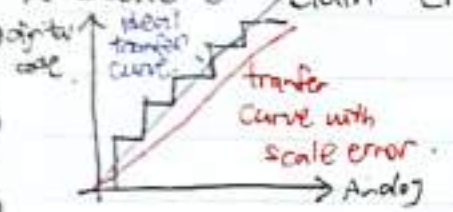
Linearity error is caused by Ideal Transfer Function Curve on an ADC. one or more unequal steps or inconsistent in the analog output voltage.



☆ Monotonicity → transfer characteristic curve of an ADC, where increase in digital inp results in increase in output digital code val throughout the entire transfer characteristic curve



☆ Scale or Gain Error. →



Departure of act inp volta from design inp volt for a full scale out code.

DIGITAL - TO - ANALOG CONVERTERS (DAC).

- construction somewhat simpler than ADC.
- Few catagories of DAC → Resistance Ladder Network → Current Steering → Charge Redistribution → Sigma - Delta.

☆ Resistance Ladder Network (DAC) → consist of an resistor network and a summing op-amp.

The current through each member of the resistor network depends on R val, and the digital inp code. When Din is at "1", the associated selector switch is switched to V_{REF}, when the Din is at "0" the selector switch is switched to Gnd.

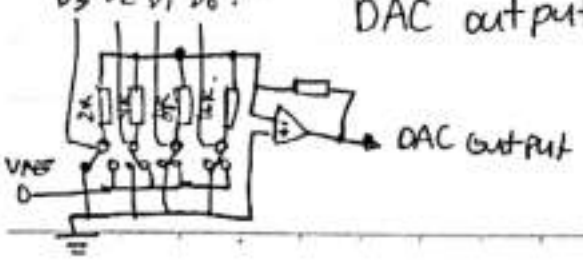
$\therefore I_T = I_0 + I_1 + I_2 + I_3$.

$= V_{REF}(D0/16R + D1/8R + D2/4R + D3/2R)$

Digital input. $= (V_{REF}/16R)(2^0 . D0 + 2^1 . D1 + 2^2 . D2 + 2^3 . D3)$

$D_3\ D_2\ D_1\ D_0$.

DAC output $= -I_T \times R = -(V_{REF}/16)(2^0 . D0 + 2^1 . D1 + 2^2 D2 + 2^3 . D3)$



No.

Date

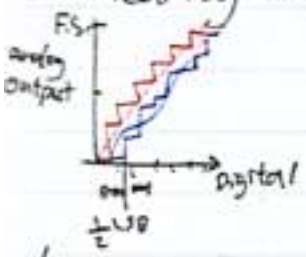
D/A CONVERTER CHARACTERISTICS.

Important DAC characteristics are similar to ADC.

☆ Resolution & Accuracy → smallest incremental change in out volt.
→ expressed in percentage of full scale / in num of DAC bits.

Resol dont tell us abt accuracy.
Accuracy is defined as the absolute error incurred in the measurement of its output. Diff b/w act out volt and full scale weighted equivalent of the digital inp code.
12-DAC to be ± LSB accurate. → ±0.244% of full scale out volt
accuracy → ±2.44mV.

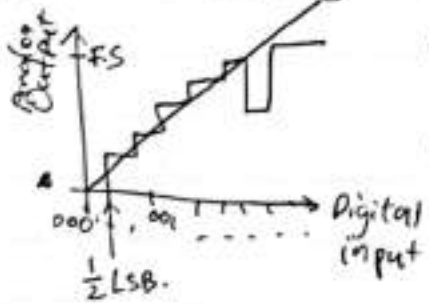
☆ Linearity → departure of the out volt from a linear transfer function code.



← same curve but digital & analog axis switched.

☆ Scale or Gain Error → departure of actual output volt from design out for a full scale input code.

☆ Monotonicity → transfer characteristic curve of DAC → when increase in digital inp code results in increase in the analog out volt throughout the entire transfer characteristic curve.



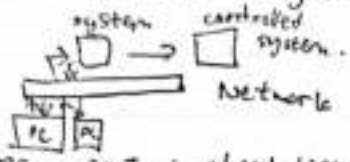
No. CINTER - 9
Date 17. 8. 22.

IoT.

& can be remotely accessed → PC is limited in nature need the user to be physically nearby a monitor & control them → better if connect to Internet and accessed from a distance.

In the Web of Things, any device can be accessed using standard web protocols. Connecting heterogeneous devices to the web makes the integration across system & applications simpler.

Traditonal →



PC → GUI developed here, local stored in PC GUI is OS-specified.

↓

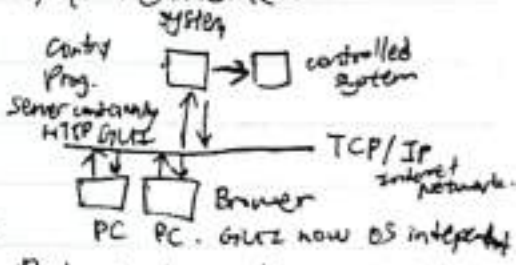
Problem → All PC must be in same OS, or else diff GUI is needed.
→ Robustness of GUI on remote station depends on the stability of underlying OS.
→ software maintenance (like upgrading & debugging) must be done in multiple locations.

↓

Increase time & effort needed to maintain the network.

↓

Use / implement an Internet System



→ control code remains the same.
→ Embedded HTTP server serves as the GUI
→ Use a TCP/IP stack (Transmission control Protocol / Internet Protocol).
→ Only maintanance is the embedded system. HTML → GUI → provided any client that host a browser independent of the OS. The TCP/IP stack is universal, hense resolves network compatibility issues

At remote station:
- Each remote station interface with TCP/IP network
- Req for GUI using a URL and display them using any browser
- Web page becomes universal GUI of the Internet System.

At the network:
- TCP/IP is an open standard that is commonly understood, inexpensive, and readily available.
- Is the universal protocol for internet connection
- Can use SMTP TCP to send alerts via email & SMS message if appropriate serve or present

Maintanance:

Software maintanance of both control code & HTML GUI are performed only at a single point at the embedded system. HTML code can now be updated from any browser, though it requires the appropriate security and/or permissions for this.

TYPES OF WEB PAGES.

* Static webpage.
  - Info don't chg.
  - For displaying help pg & fixed diagram.
  - May not be suitable as GUI

HTML — CSS, JS.

* Dynamic webpage. accepts info from operator and provide real-time feedback on status of embedded system. use to provide real time info to user.
  - Info can chg.
  - Used to provide real-time info abt embedded system.
  - Ability to be updated makes them suitable as GUI

HTML — CSS, SQL, PHP, JS

more useful for GUI / software control panel in the case of an Internet System.

HARDWARE → PC.

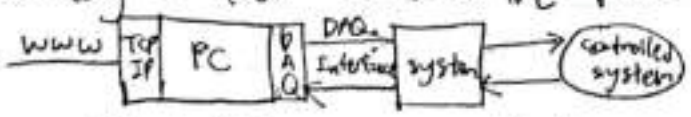
Many ways to implement the hardware of an internet system. Eg. PC.

Internet System → Personal Computer
→ Board Level Controllers
→ Internet Chips (iChips)

- One way to control system remotely is to connect it to a PC equipped with both a Data Acquisition (DAQ) Card and an Internet Connection

Data Acquisition Card (DAQ)

- consist of several Digital I/O and Analog I/O Ports that can be used to interface with the system electrically. A DAQ provide signal to start a system. Alt. it may issue multi-bit commands to the system to chg its behaviour or performance.
- Obtains info / status from the system tht is to be displayed in the software control panel. This info can be analog / digital, if analog → then ADC → the DAQ to convert for PC.



DAQ consist: - 16-48 channels of digital I/O.
- 8 channels of multiplexed 12-bit 100kHz ADC
- 2 or more internal timers.

A multiplexed ADC means tht although up to 8 analog devices can be connected to the DAQ card, only 1 of them can be read & converted to digital at a time. As there is only 1 ADC available for conversion

SET-UP.

PC must have - HTML GUI, - GUI, - Dynamic HTML Page, software eg. Labview

With the PC / DAQ card → control the system directly, the PC will act as the server & interface directly with client.

However, this is an expensive solution if sole purpose of PC is to control the system, It is more viable if PC is used for other secondary purpose as well.

No. CINTER - 09
Date 17 . 8 . 22.

HARDWARE → BOARD LEVEL CONTROLLERS

Another way to implement the hardware of an internet system is by using board level controllers
It consist of : - A microcontroller chip, - TCP/IP stack for internet interface. - Interface for embedded system
- firmware to handle 2 interface - RAM/ROM to store firmware

- It is itself an Internet system but this don't control a system. instead it handles the interface b/w the computer & the internet

3 version → Parallel I/O → Serial Comm → Internet Chip.
basically work the same way, but Interface to Inet sys is diff. support software avail to reduce development time.

Parallel I/O Board Controller.

→ Interfaces with the target embeded system via digital input and output ports.
advantage → simple, ez to setup & interface
→ easy understand.
disadvantage → No analog I/O
→ Limited no. of connection line (16 in, 16out)
→ Large no. of connection to potentially ES
→ Limited physical separation to ES.

Serial Communication Board Controller.

→ To overcome the disadvantage of the parallel I/O board controller, a serial comm interface between the internet board and the embedded system can be used.
advantage → No. "I/O lines" flexible.
→ accomodate analog & digital info.
→ Greater physical distance from system possible.
→ No. of wired connections minimized
disadvantage → slower transfer rate
→ More complicated firmware.

HARDWARE → INTERNET CHIPS (iChips). ← cheaper alt ESP8266 (WiFi Module Chip)

Another way to implement the hardware of internet sys is by using iChips.
- Like Board Level Controllers, but smaller.
- System-On-Chip IC for implementing internet system, can host web-pg.
- Like BLC, iChips are internet system themselves, they interface with the process system and the internet.
- Small footprint → iChips are sited on the system board itself.
→ minimized connections to other board.
hence,
→ must incorporate from the design stage.
- With iChips, we can now implement internet connectivity on almost any device, such as mobile phones, watch & PDAs.
- Expensive → so use Beck Hybrid chip → though as a miniaturised board lvl EIS.
~~has our~~ → has most functionality of a normal iChips.
22 ~~mm~~ x 44 x 9.5mm (w x l x h)

features: - 44 pin package.
- 12 x 12mm footprint.
- In built 50MHz 8051 microcontroller.
- 16kB SRAM
- 64kB internal Flash EEPROM.
- 4 multiplexed serial ports at 15Mbps/port
- 512 bytes of bi-directional buffer for each port.
- Support diff protocol (RS232, RS485).

- 10 or 100 base T ethernet
- Flexible TCP/IP protocol stack
- Software support
- security features.

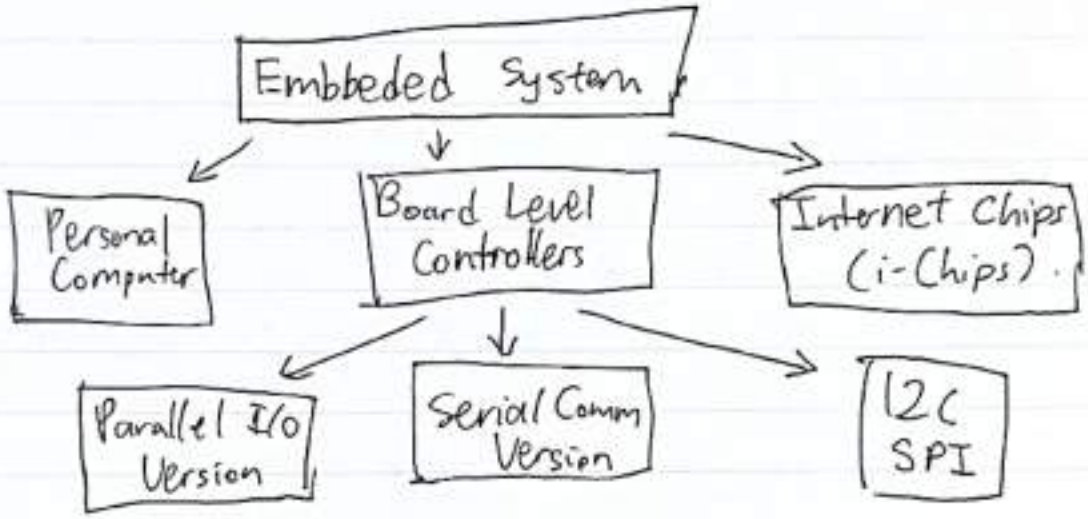
No.

Date

NETWORK TECHNOLOGIES FOR INTERNET SYSTEMS.

The most common type of network technology used for internet connection are wired ones → Wired dial up → slow (up to 56kbps) but cheaper.
→ wired broadband → higher bandwidth & faster.
→ wireless.

The type of technology to use depends on the type of information being exchanged, along with its availibility (or practicality) of fixed access point.



Control word → Out32 (Base + 3, 0x__) Control = Base + 3.
Data → Inp32 (Base) Port A = Base
Status → Out32 (Base + 1, 0x__) Port B = Base + 1
Control → Out32 (Base + 2, 0x__) Port C = Base + 2.

Permanent Magnet, Variable reluctance, Hybrid ← type of motors.

Data → 0 → 3 ~ 15V, 1 → -3 ~ -15V Control → ON → 3 ~ 15V, OFF → -3 ~ -15V

$\frac{\text{range}}{\text{bit out}} = \frac{20}{4096}$ (-10 to 10; 12 bit)

Resolution → smallest incremental changes in out volt. If DAC.
Accuracy → absolute error incurred in the measurement & its output → Diff b/w act. out volt and ~~full scale~~ inp code
Linearity → departure of the out volt from linear transfer funct (curve if AD)
Scale / Gain error → departure of act. out volt from design out for a full-scale inp code.
Monotonicity → transfer characteristic of AD / DAC → when increase in digital inp code → increase analog out volt throughout the entire transfer char curve

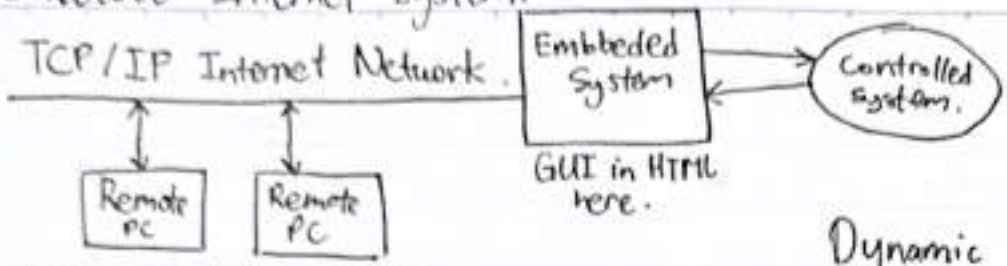
1.8 per step. / 10 rpm.
rpm to steps/sec → $10 \times \frac{360°}{60} = 60°$ ~~steps~~ sec
1 step is 1.8 → $\frac{60°}{1.8°} = 33.33$ steps / sec
how many step for 1 full rotation → $\frac{360}{1.8} = 200$ steps

status = Inp32 (LSR) ← line status register.
data = Inp32 (IB) ← input buffer.

Sampling freq → 2x original ← if hv 2 freq take higher one.

Motor at high speed → The back EMF generated by the moving rotor reduces the voltage available to drive current through the motor coils → loss of sync & resonance

Embedded Internet System.



GUI accesed as a webpage using a webbrowser.

Information webpage → present static data to browser.
→ for first diagram 2 help B

Dynamic webpage → can accept input from operator & provide real time feedback on current status.

Enhance security in HTTP:
- Restrict client by verifying valid IP address
- Use passwords to protect resource on the server.
- Encrypt the data in the HTTP session.

Reduce work in stepper motor → Indexer & Translator.
Half & Full step diff → precision of movement will be different.

USB descriptors → Device, Config, Interface, Endpoint, String
USB data transfer mode → Control, Isochronous, Bulk, Interupt.
Device driver → provides a software interface to hardware devices enabling OS and other computer programs to access hardware functions without needing to know precise details of the hardware used.
Enumeration → initiated both when the host is powered up and a device connected or removed from the network.
Why Hold in ADC → ADC needs a finite amount of time to measure the signal voltage.

Implement Embedded Internet System:
- Using PC with Internet Connection and DAQ
- Using a board level Internet Controller.
- Using a chip level Internet Controller.

Why use HTML GUI: → platform independent, easiler to manage.
→ Use internet protocol, which is stable & well estabilished, no need customized network