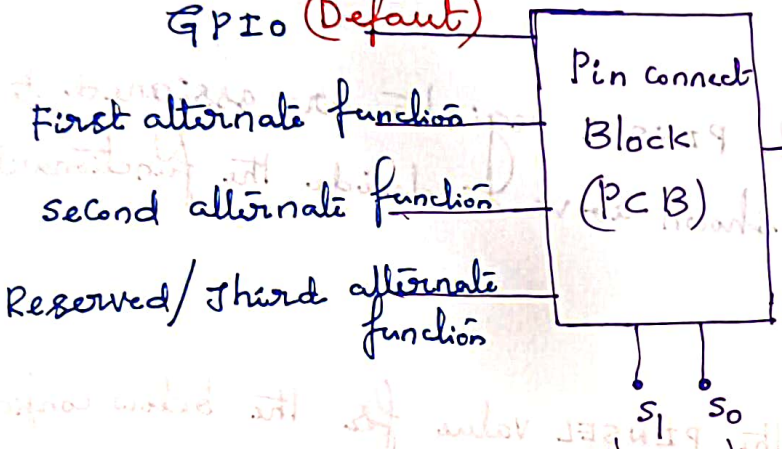
* PIN Connect Block/Module of LPC 2148:

→ LPC 2148 is a ARM-7 architecture based microcontroller.

→ Each pin of LPC 2148 can perform four different functions as shown below. Default function is GPIO.



where,
x = 0 & 1 → port number
y = 0 to 31 → pin number

select lines decides the function of the pin.

→ The values of the select lines and the corresponding function of the pin is as follows.

S_1	S_0	Function of the pin
0	0	GPIO
0	1	First alternate function
1	0	Second alternate —"—
1	0	Reserved/Third alternate function

→ The three Registers are associated with pin connect block.

1) PINSELECT0
2) PINSELECT1
3) PINSELECT2

PINSEL0, PINSEL1, PINSEL2 Registers. All are 32 bit registers

1) PINSEL0 = 0000 0000 00 00 0000 0000 0000 0000 0000
P0.15 P0.14 P0.1 P0.0

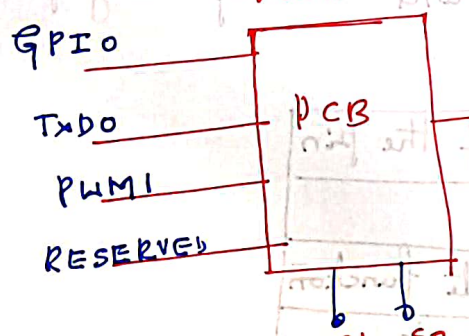
2) PINSEL1 = 0000 0000 0000 0000 0000 0000 0000 0000
P0.31 P0.16

3) PINSEL2 = 0000 0000 0000 0000 0000 0000 0000 0000
P1.31 P1.16

→ Each two bits of PINSEL register are assigned to one pin of port as shown above to decide the functionality of the pin.

* Example: What is the PINSEL value for the below configuration

P0.0 → TxD
P0.1 → PWM



PINSEL0 = 0000 0000 0000 0000 0000 0000 0000 0001

∴ PINSEL0 = 0x00000001 ; P0.0 → TxD

PINSEL0 = 0x00000002 ; P0.1 → PWM 0010
2

NOTE : The direction control bit in the IODIR0 register is effective only when the GPIO function is selected for a pin.

* Introduction to the ARM-7 TDMI (LPC 2148) Microcontroller:

TDMI expansion

T → Thumb mode (16-bit Instructions)

D → Debugging feature

M → Fast Multiplier

I → Enhanced Interrupts

* Main basic features:

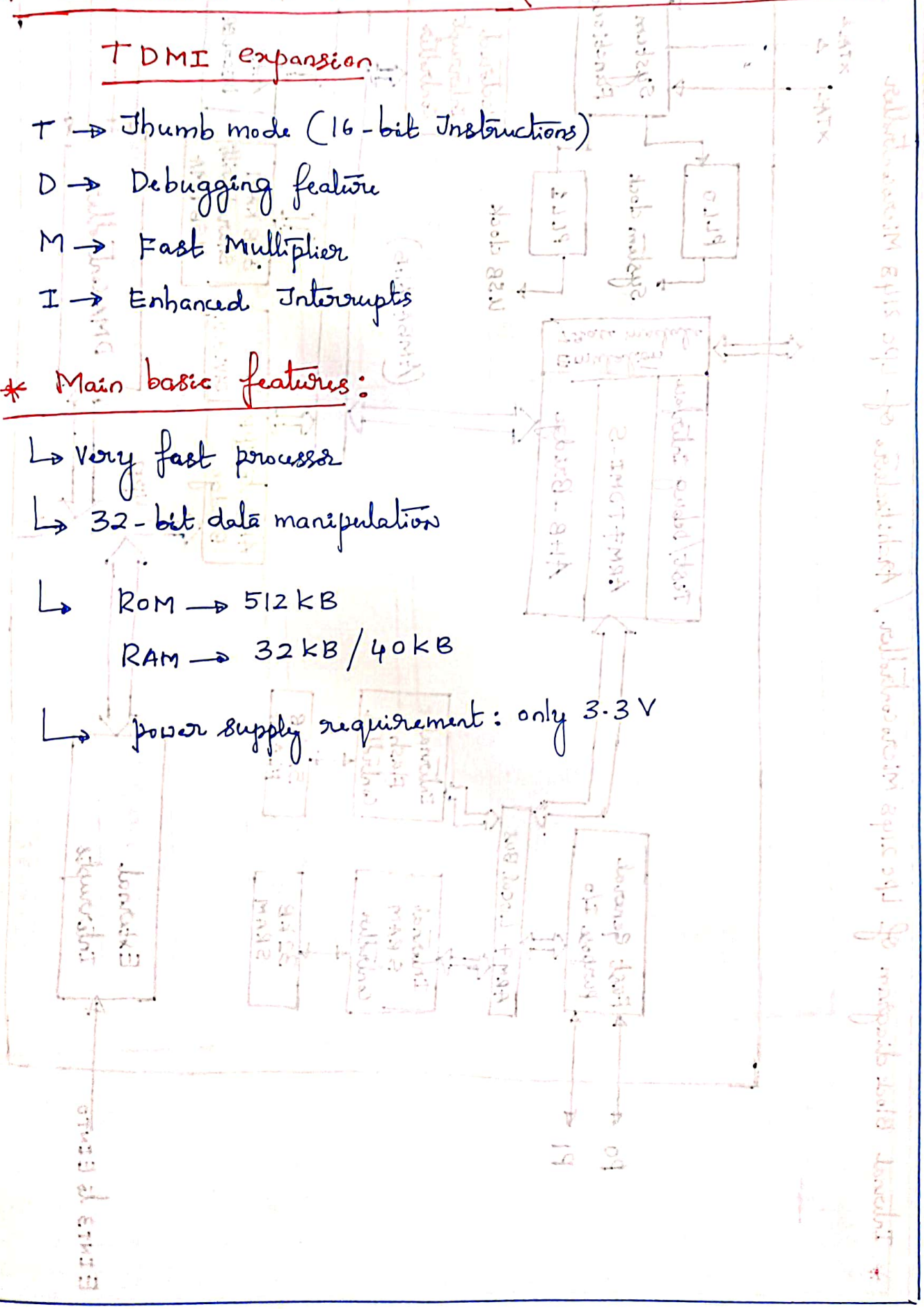
↳ Very fast processor

↳ 32-bit data manipulation

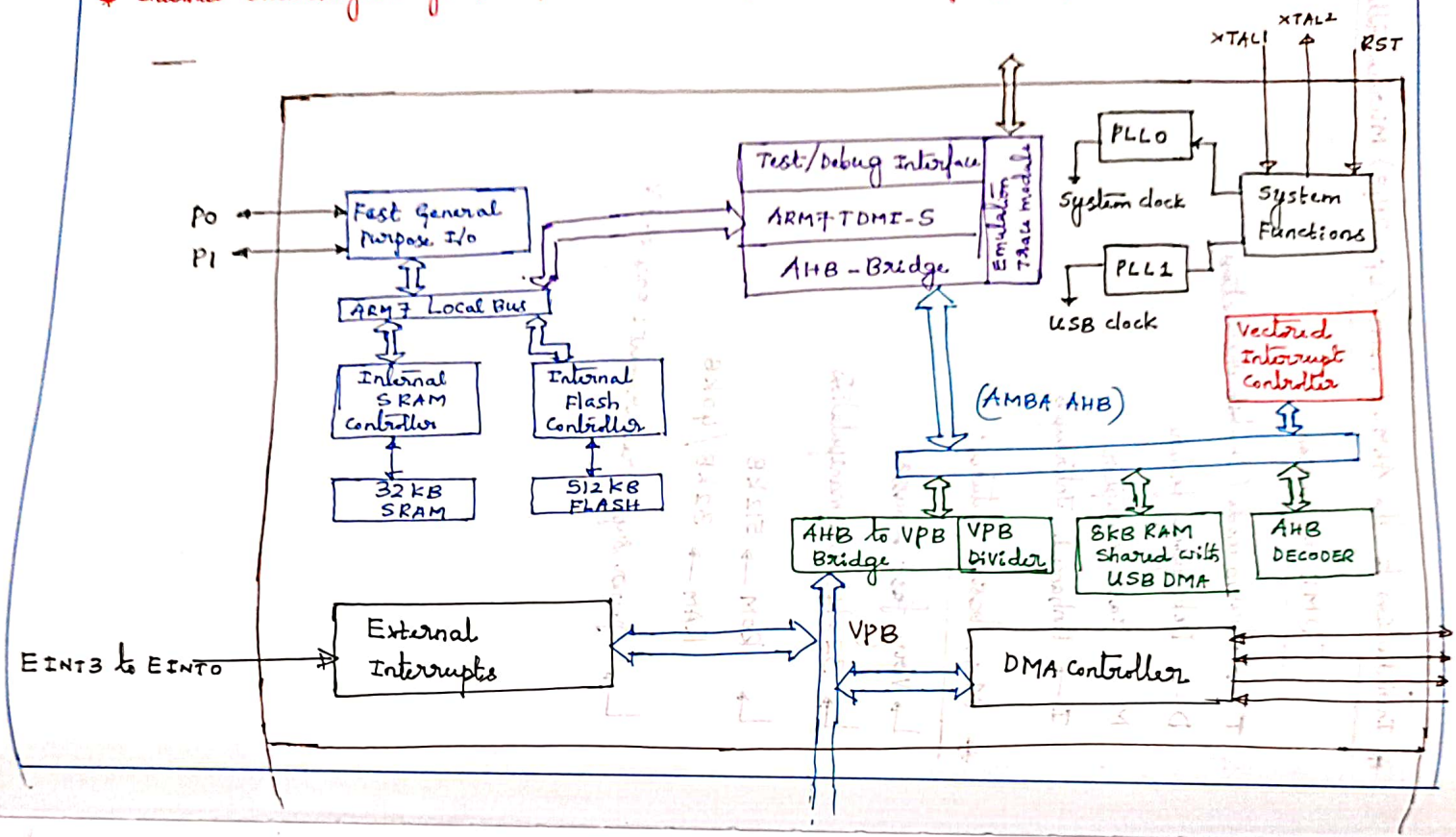
↳ ROM → 512 kB

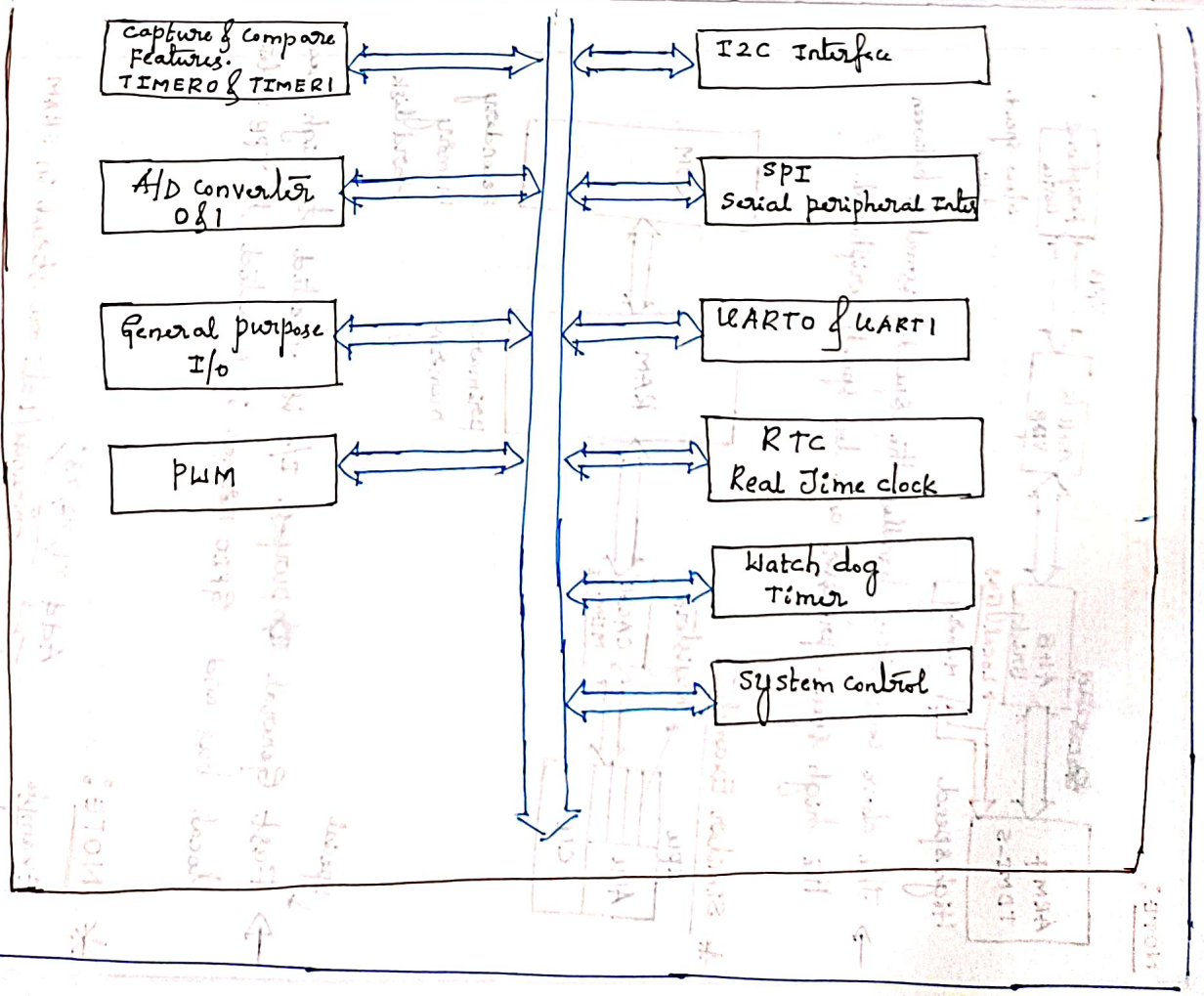
RAM → 32 kB / 40 kB

↳ power supply requirement: only 3.3 V

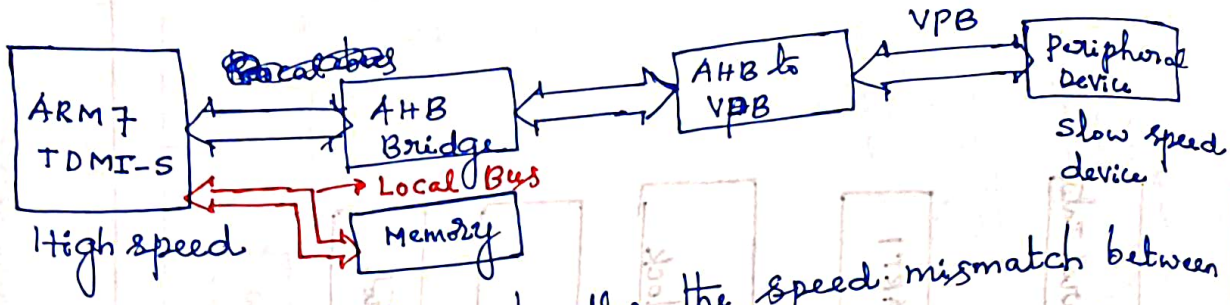


* Internal Block diagram of Lpc 2148 Microcontroller / Architecture of Lpc 2148 Microcontroller



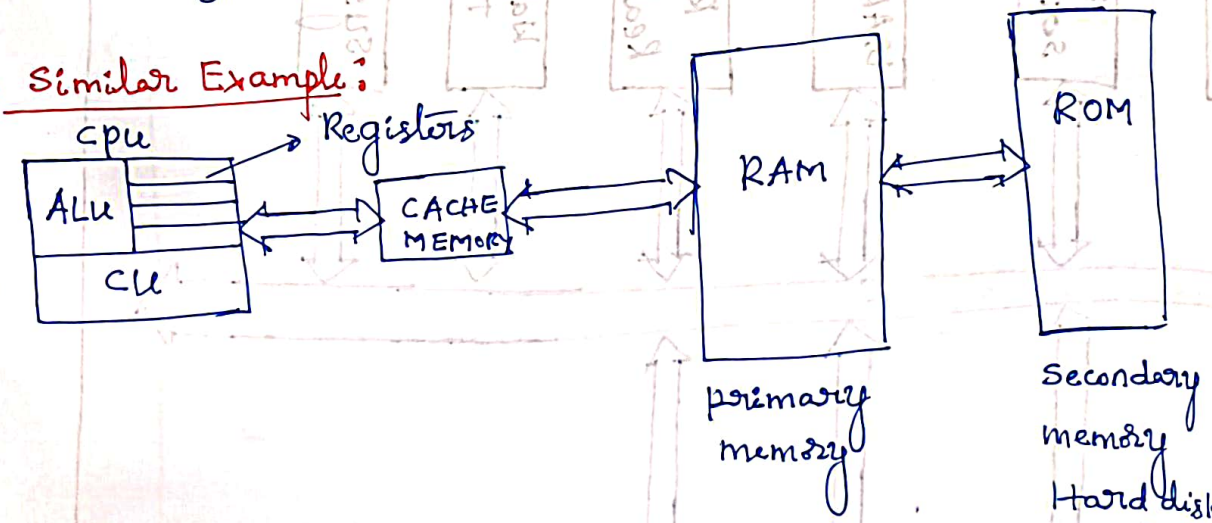


NOTE:



→ The above connectivity handles the speed mismatch between the high speed processor and low speed peripheral.

* Similar Example:



→ Fast General purpose (special) I/O is connected to high speed local bus and GPIO normal is connected to VPB system

* NOTE:

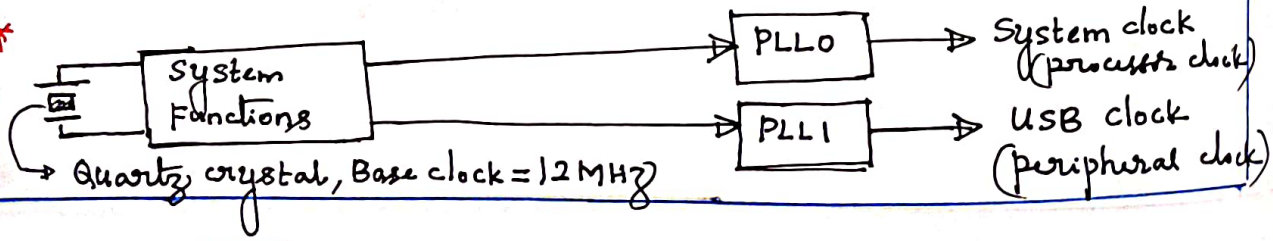
Example

Add r1, r2, r3;

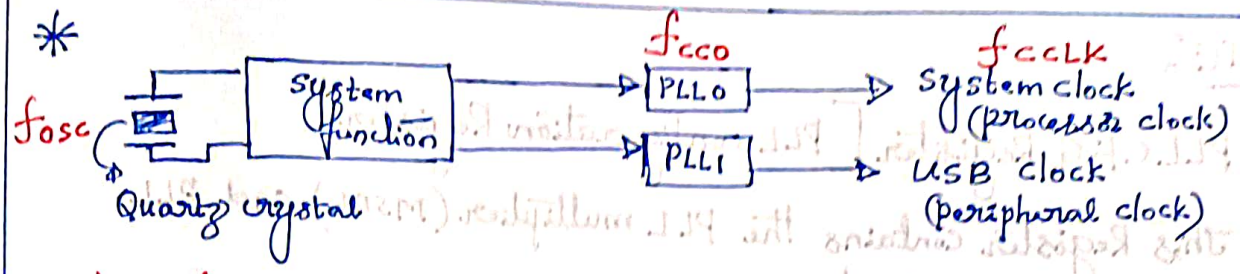
operands/data are stored in SRAM

→ Instruction is stored in FLASH memory.

*



*



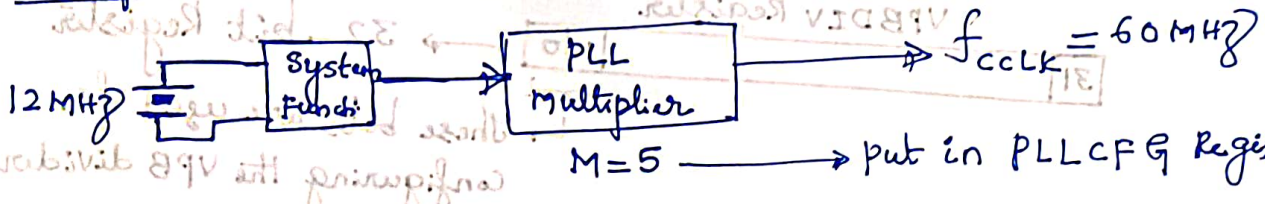
1) $f_{CCLK} = M \times f_{OSC}$

- f_{OSC} → Frequency of the Quartz crystal oscillator
- M → PLL Multiplier Value in PLLCFG Register
- f_{CCLK} → System/processor clock frequency.

2) $f_{CCO} = f_{CCLK} \times 2 \times P$

- P → PLL divider Value in PLLCFG reg.
- f_{CCLK} → System/processor clock frequency
- f_{CCO} → frequency of the PLL oscillator

Example:



* Finding Multiplier Value (M) and Divider (P) Value:

Given specifications: $f_{CCLK} = 60\,MHz$; $f_{OSC} = 12\,MHz$.

1) We have Equation, $M = \frac{f_{CCLK}}{f_{OSC}} = \frac{60\,MHz}{12\,MHz} = 5$

2) Range of f_{CCO} is 156 MHz to 320 MHz.

For 156 MHz, $P = \frac{f_{CCO}}{2 \times f_{CCLK}} = \frac{156\,MHz}{2 \times 60\,MHz} = 1.3$

For 320 MHz, $P = \frac{320\,MHz}{2 \times 60\,MHz} = 2.67$

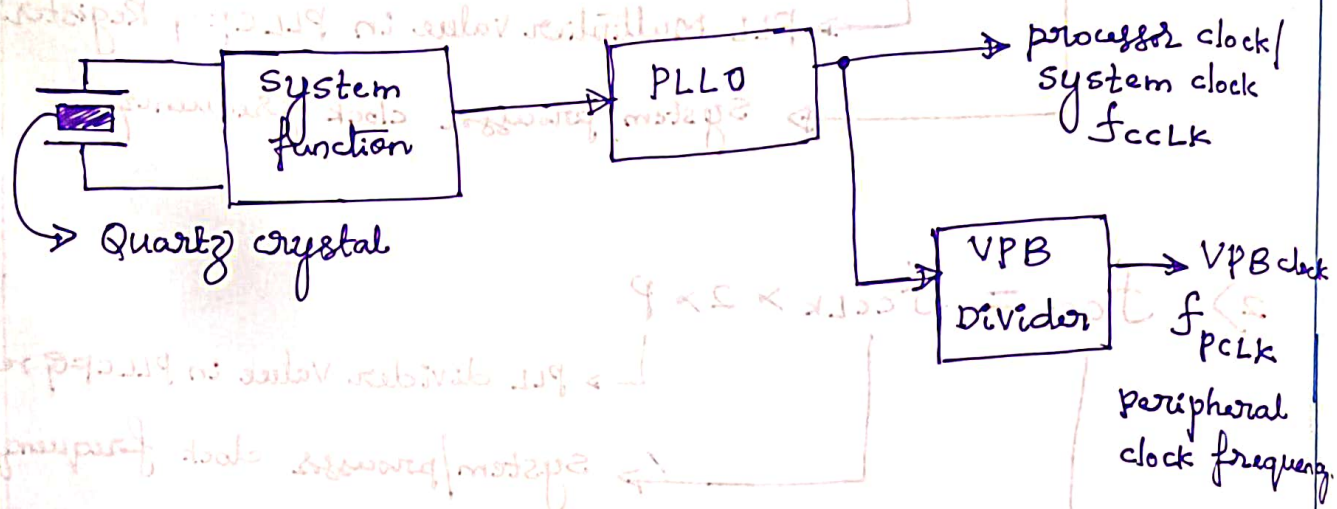
The value of P must be integer. Hence, the integer value between 1.3 and 2.67 is 2.

NOTE:

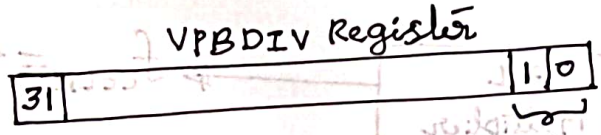
↳ PLLCFG Register [PLL Configuration Register]

↳ This Register contains the PLL multiplier (MSEL) and PLL divider (PSEL) values.

* VPB Divider [VLSI peripheral Bus] divider:



↳ VPBDIV Register is used for configuring the VPB Divider.



↳ These bits are used for configuring the VPB divider

↳ VPBDIV Value = 00 → f_{PCLK} = VPB bus CLK = $\frac{1}{4} \times f_{CCLK}$

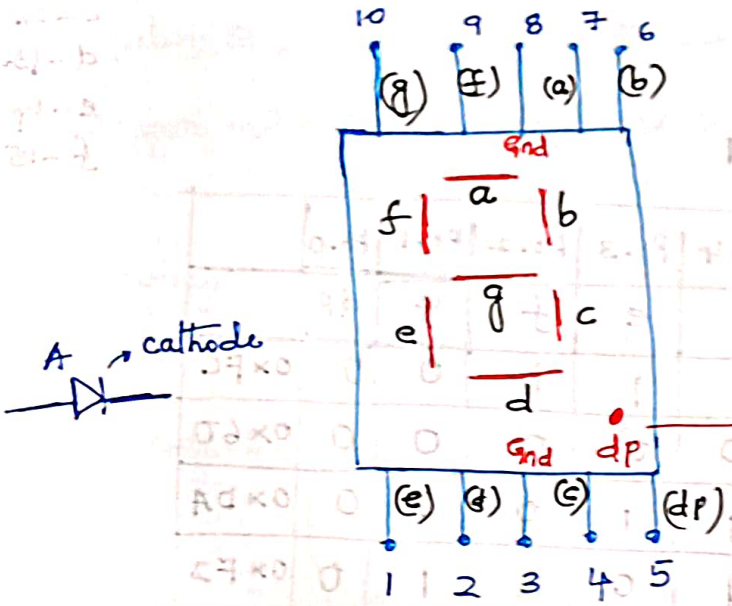
= 01 → No division

= 10 → $f_{PCLK} = \frac{1}{2} \times f_{CCLK}$

= 11 → Reserved.

* NOTE: By default value is 00 ∴ $f_{PCLK} = \frac{1}{4} \times f_{CCLK}$

* Seven Segment display Interfacing :



8 → Gnd
3 → Gnd
} Two ground pins are provided to reduce the circuit complexity.

7 → a
6 → b
4 → c
2 → d
1 → e
9 → f
10 → g

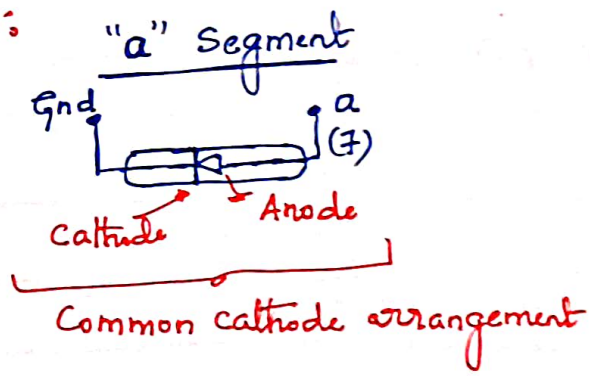
→ Totally eight LED's including dp (decimal point)

* Types of Seven segment display
- Common cathode
- Common anode

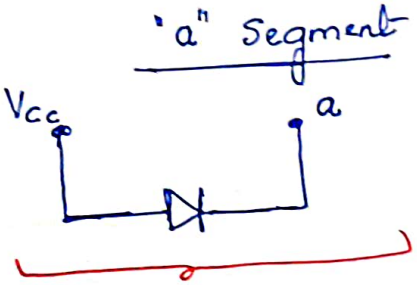
1) Common cathode: All cathodes of eight LED's are connected to ground point. we should apply '1' (High) to make particular segment "ON".

2) Common Anode: All anodes of eight LED's are connected to V_{cc} point. we should apply '0' (Low) to make particular segment "ON".

Example:

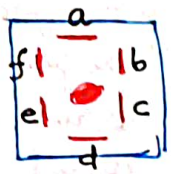


Common cathode arrangement

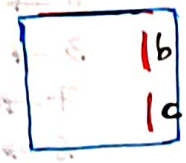


common anode arrangement

Example:



Display: 0



Display: 1

a - 10
b - 11
c - 12
d - 13
e - 14
f - 15

	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	
	a	b	c	d	e	f	g	dp	
0	1	1	1	1	1	1	0	0	0xFC
1	0	1	1	0	0	0	0	0	0x60
2	1	1	0	1	1	0	1	0	0xDA
3	1	1	1	1	0	0	1	0	0xF2
4	0	1	1	0	0	1	1	0	0x66
5	1	0	1	1	0	1	1	0	0xB6
6	1	0	1	1	1	1	1	0	0xBE
7	1	1	1	0	0	0	0	0	0xE0
8	1	1	1	1	1	1	1	0	0xFE
9	1	1	1	1	0	1	1	0	0xF6

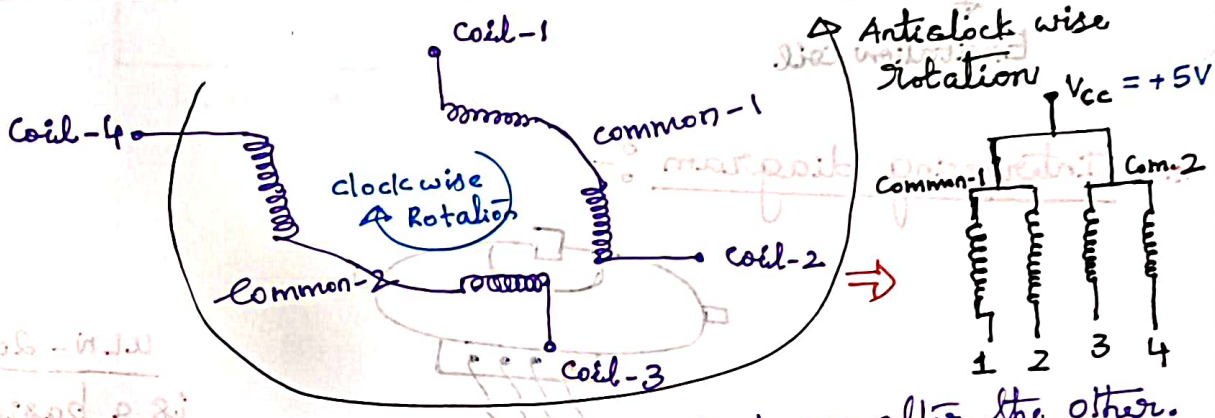
0xBE

* Stepper motor Interfacing :

→ Stepper motor is an electric motor which rotates in a series of small angular steps instead of continuously

→ Stepper motor is also known as step motor or stepping motor.

* principle of Operation : Electrical model of stepper motor



→ All the four coils should get energized one after the other.

* Excitation Sequence :

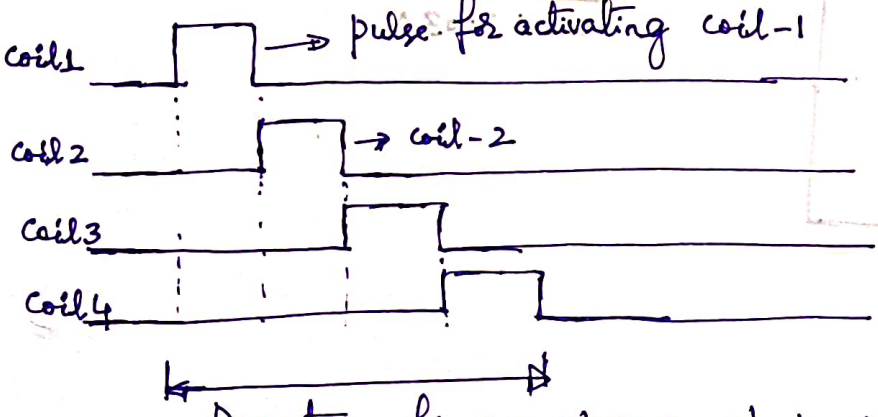
Step	coil 1	coil 2	coil 3	coil 4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

NOTE :

Types of stepper motor

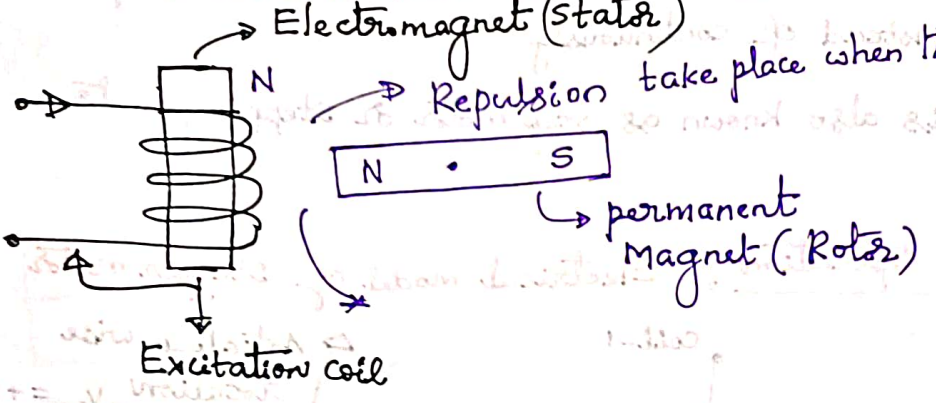
1) Unipolar stepper motor

2) Bipolar stepper motor : No common connection in the circuit.

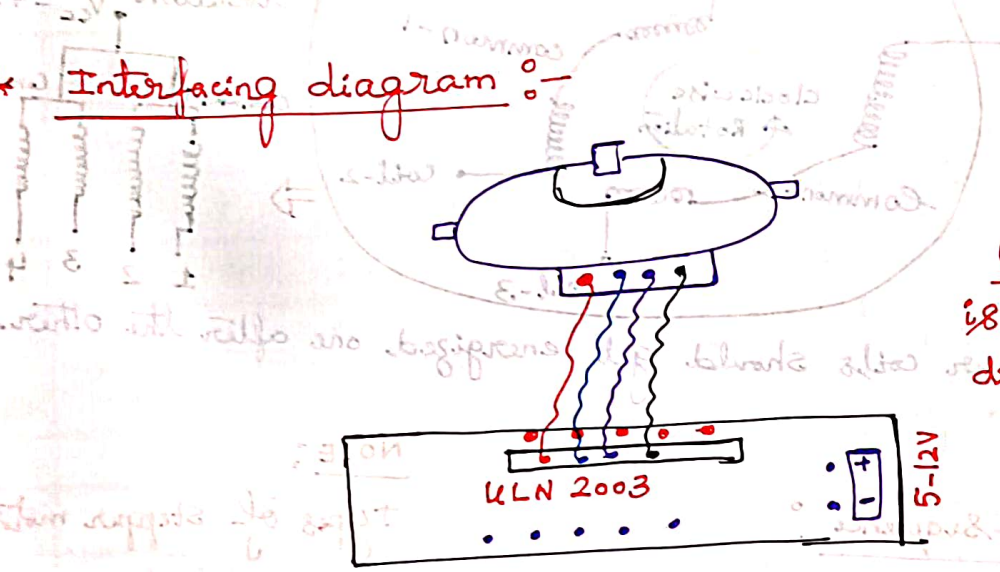


Duration for one step in clock wise direction.

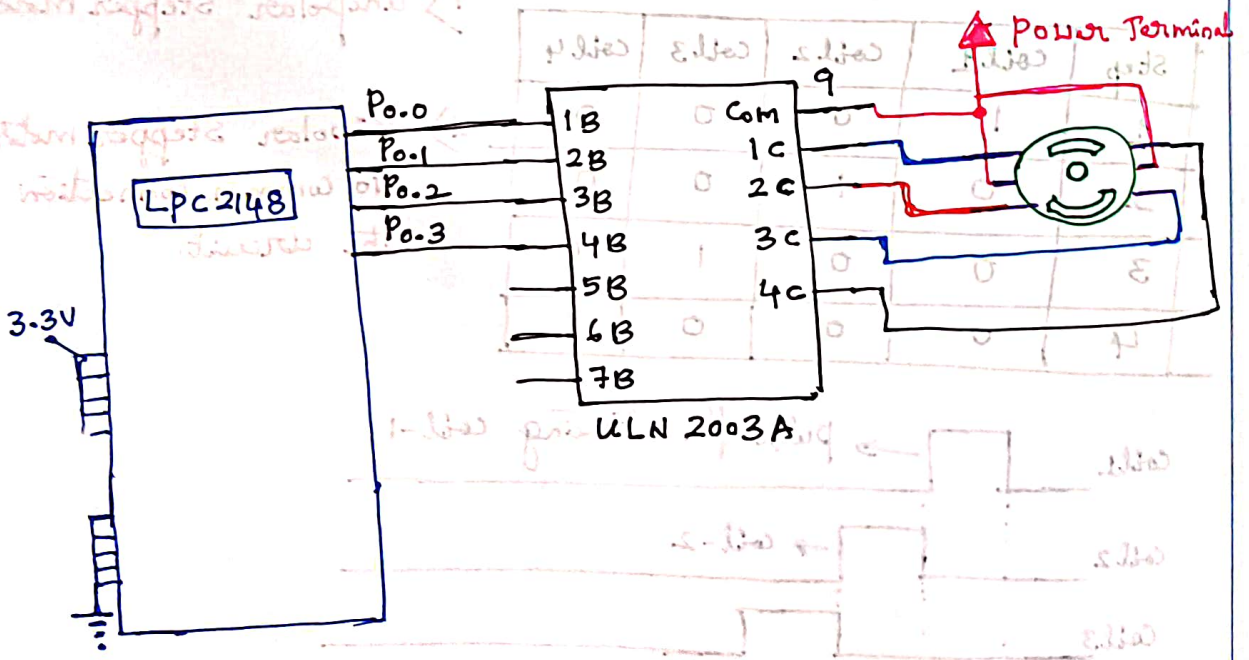
* Basic principle



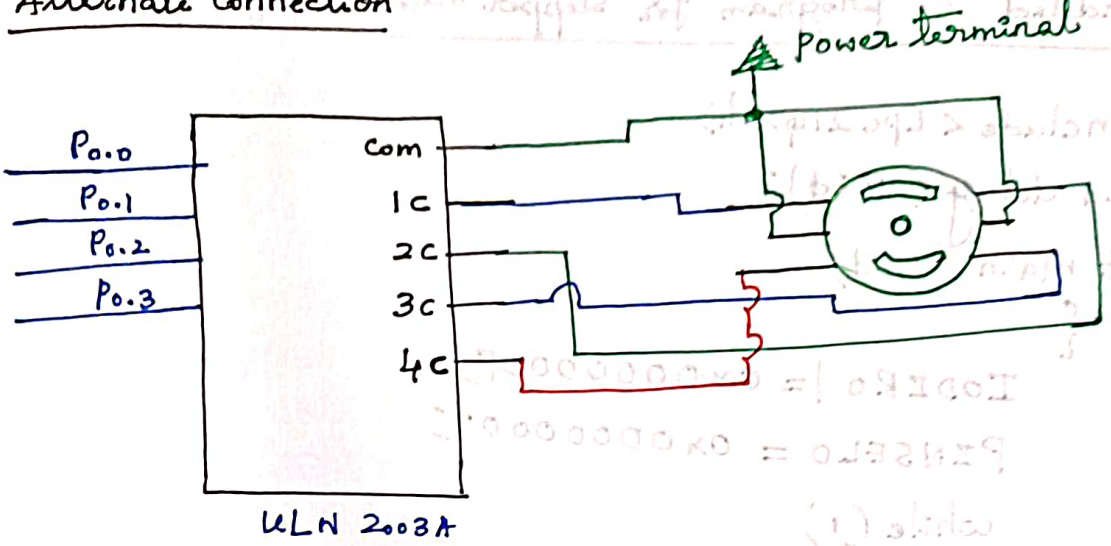
* Interfacing diagram :-



ULN-2003 is a basic motor driver circuit



Alternate connection



* Pipeline:

→ The ARM7 TDMI processor uses a three-stage pipelining technique to increase the flow of instructions to the processor.

→ Pipelining is a technique where multiple instructions are overlapped during execution.

→ Pipelining is a technique which defines the overlapping of stages for Instruction execution.

Example: Execution of four (4) Instructions program.

	①	②	③	④	⑤	⑥	⑦	⑧	⑨	⑩	⑪	⑫	⑬	⑭
Normal Execution without Pipelining	Inst1 F1	D1	E1											
	2			F2	D2	E2								
	Inst3						F3	D3	E3					
	Inst4									F4	D4	E4		
ARM-7 with Pipelining	1 F1	D1	E1											
	2	F2	D2	E2										
	Inst3		F3	D3	E3									
	Inst4			F4	D4	E4								

Conclusion: For the execution of 4-Instructions program, Normal execution without pipelining technique takes 12 clock cycles whereas in ARM-7 with pipelining technique takes 6 clock cycles. Therefore, pipelining technique improves the speed of execution.

∴ Total clock cycles in pipelining $= k + n - 1$

$= 3 + (4 - 1) = 6$

Where, k → No. of Stages

n → No. of Instructions.

* Differences between the General-purpose computing Systems and Embedded Systems:

→ General-purpose computing systems are versatile, designed for a wide range of tasks.

→ Embedded Systems are specialized, optimized for specific functions within a larger system.

General-purpose Computing Systems	Embedded Systems
1) Are versatile, designed for a wide range of tasks.	1) Are specialized, optimized for specific functions, often with real-time constraints.
2) Flexibility: can be easily adapted to new tasks.	2) No flexibility.
3) Functionality: Wide range of tasks	3) Specific, often real-time
4) Resources: Ample processing power, memory & storage	4) Limited processing power, memory & storage
5) User Interface: Has a GUI	5) Minimal or no user Interface
6) Examples: Desktops, laptops, servers	6) Automotive control systems, Industrial automation.
7) purpose: Multipurpose	7) Single functioned
8) user Interface: Keyboard, display, mouse, Touch screen	8) Integrated into the real world: Button, sensors.

* Embedded System:

→ Embedded System is defined as a computing system which is designated to serve a dedicated purpose.

→ An Embedded System is a computer system with a dedicated function within a larger system.

→ An Embedded System consists of both Hardware and Software

* classification Of Embedded Systems :—

→ The classification can be done based on the following.

1) Based on complexity
2) Based on generation
3) Based on performance and functional Requirements
4) Based on Deterministic Behavior
5) Based on Triggering

1) Based on complexity:

- Small scale embedded systems
- Medium scale embedded systems
- Large scale Embedded systems

2) Based on Generation

- First generation: During 1960s
- Second generation
- Third generation
- Fourth generation

3) Based on performance and Functional requirements:

- Network Embedded Systems
- Real-time Embedded Systems
- Mobile Embedded Systems
- Standalone Embedded Systems

4) Based on Deterministic Behavior:

- Soft Real-time Systems
- Hard Real-time Systems.

5) Based on Triggering :

- Event - Triggered
- Time - Triggered

* Characteristics of Embedded systems :

- Task specific
- Time specific
- High efficiency
- High reliability
- Highly stable
- Low cost
- Requires less power
- →

→ These characteristics are to be considered while designing the embedded systems.

* The system design of an embedded system has constraints

- Performance
- Power
- Size
- Design
- price

* LCD Interfacing: [Liquid-crystal display]

→ Both Seven-segment and LCD displays are used for displaying informations.

→ Seven segment displays are simpler and used for displaying numbers and limited characters.

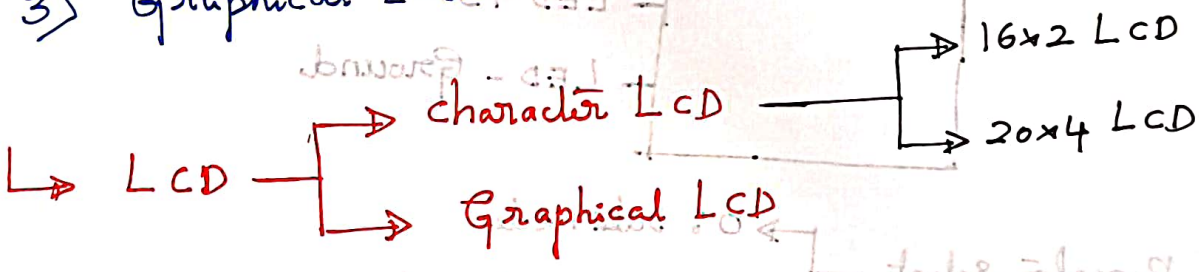
→ LCD displays are used to display Text information. It can display a wide range of characters.

* Basic types of LCD's available:

1) 16×2 LCD: There are two lines and in each line 16 characters can be displayed.

2) 20×4 LCD: There are four lines and in each line 20 characters can be displayed.

3) Graphical LCD:

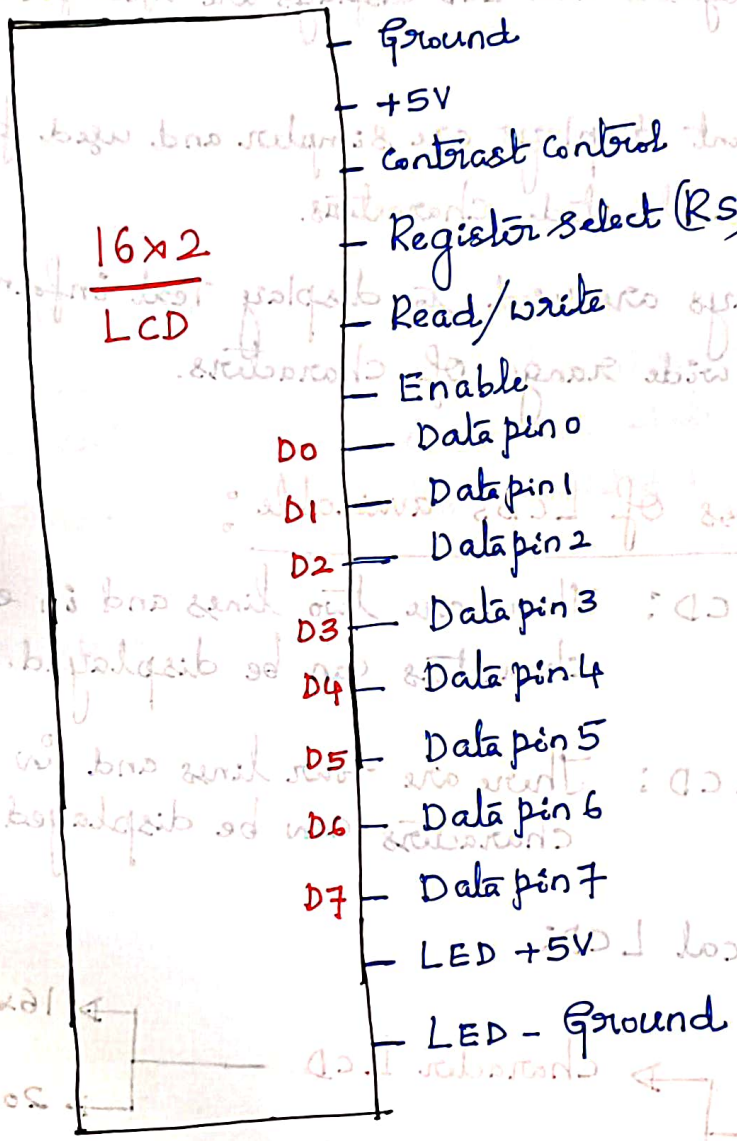


NOTE: 16×1, 16×4 } also available.

→ LCD controller (HD44780) which is present inside the LCD display which controls the operations.

* Pin diagram of 16×2 LCD :-

→ There are 16 pins available



Pin 4: Register select → 0: Data mode; 1: Command mode

Pin 5: Read/write control pin → 0: Write operation; 1: Read operation

Pin 6: Enable pin. This pin should be high for Read/write operation. ← Initiate operation

Pins 7 – 14: Data pins. These pins are used to send data to the display.

* Bitwise operations in Embedded programming:

→ Bitwise Vs Boolean operations

→ &: Bitwise And operator

|: Bitwise OR operator

^: Bitwise Ex-OR operator

!: NOT

→ Basics of Bitwise operations:

A	B	A|B	A&B	A^B
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	0

→ Boolean operators:

&& → Logical And operator

|| → Logical OR operator

! → Logical NOT operator

NOTE:

(x && y) → Returns True only if both x and y are true.

(x || y) → Returns True if either x or y are true.

Example: A = 0x3B → 00111011

B = 0x96 → 10010110

A&B = 0x12 → 00010010

A|B = 0xBF → 10111111

A^B = 0xAD → 10101101

A - 10
B - 11
C - 12
D - 13

A D

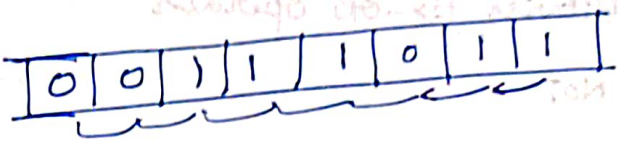
* Bitwise shift operators :

→ << → Left shift operator

>> → Right shift operator.

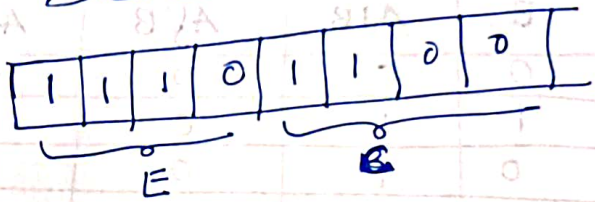
Example : C = A << 2 ; // shift left A by 2

A = 0x3B



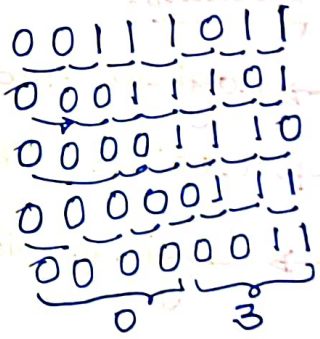
A =

A = 0xEC



Example : D = B >> 4 ;

B = 0x3B ;



B = 0x03 ;

* setting bits, Inverting bits :

Example : Setting bit

1) `bits = bits | (1 << 7) ; /* sets bit 7 */`

$(b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0) \mid (1000\,0000) \rightarrow (1 b_6 b_5 b_4 b_3 b_2 b_1 b_0)$

Equivalent Statement

2) `bits |= (1 << 7) ; // Sets bit 7`

Example : Testing Bits

`if ((bits & 64) != 0) // check to see if bit 6 is set or not.`

64 decimal
16|64
4 – 0
64 = 40H

40H

$(b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0) \& (01000000) = 0b_6 0000000$

`if ( bits & (1<<6)) // check to see if bit 6 is set or not`

Example :

`bits = bits & (1 << 7);`

$(b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0) \& (10000000) \rightarrow (b_7 0000000)$

`bits &= (1<<7);`

`bits &= ~(1<<7);` clear bits → (01111111)