LECTURE 26 - NXP LPC2148 PLL

* STEPS FOR PLL CONFIGURATION :

(FOSC = 12MHz, CCLK = 60MHz, PCLK = 15MHz)

① Enable PLL and Disconnect PLL from CPU and other peripherals
PLL0CON = 0x01;

② Configure M value and P value
PLL0CFG = 0x24;

③ Feed sequence for locking to desired frequency
PLL0FEED = 0xAA; PLL0FEED = 0x55;

④ Check whether the PLL0 has locked on to the desired frequency
while (!(PLL0STAT & 0x00000400)); PLOCK = 0

⑤ Enable (again) PLL and connect the PLL to CPU
PLL0CON = 0x03;

⑥ Feed sequence for connecting the PLL0 as system clock
PLL0FEED = 0xAA; PLL0FEED = 0x55;

⑦ Configure PCLK at 1/4 frequency of System clock
VPBDIV = 0x00; CCLK = 60MHz and PCLK = 15MHz

* STEP 1 - PLL0CON = 0x01;

Table - PLL control register (PLL0CON - address 0xE01F C080, PLL1CON - address 0xE01F C0A0) bit description

Bit	Symbol	Description	Reset value
0	PLLE	PLL Enable. When one, and after a valid PLL feed, this bit will activate the PLL and allow it to lock to the requested frequency.	0
1	PLLC	PLL Connect. When PLLC and PLLE are both set to one, and after a valid PLL feed, connects the PLL as the clock source for the microcontroller. Otherwise, the oscillator clock is used directly by the microcontroller.	0

7:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

* STEP 2 - PLLOCFG = 0x24;

Table - PLL configuration register (PLLOCFG - address 0xE01F C084, PLLCFG - address 0xE01F C0A4) bit description

Bit	Symbol	Description	Reset Value
4:0	MSEL	PLL Multiplier value. Supplies the value "m" in the PLL frequency calculations.	0
6:5	PSEL	PLL Divider value. Supplies the value "P" in the PLL frequency calculations.	0
7	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

$$CCLK = M \times F_{OSC}$$

$F_{OSC} = 12\,MHz$, $CCLK = 60\,MHz$, $M = ? \rightarrow 5$

$$F_{CCO} = CCLK \times 2 \times P$$

$F_{CCO} = 156\,MHz$ to $320\,MHz$, $CCLK = 60\,MHz$, $P = ? \rightarrow 2$

Table - PLL Multiplier values

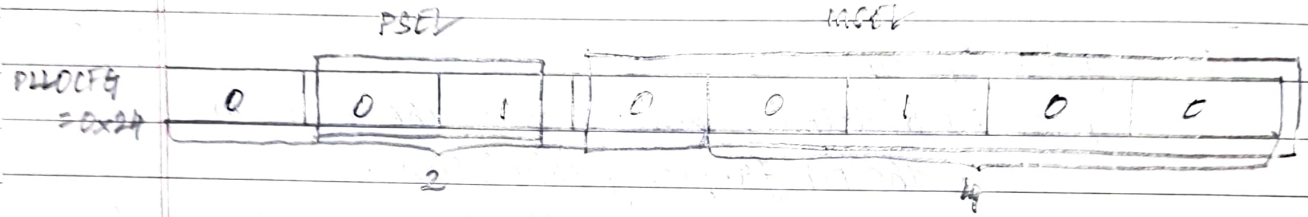
MSEL Bits (PLLCFG bits [4:0])	Value of M
00000	1
00001	2
00010	3
...	...
11110	31
11111	32

For 214x
$F_{OSC} = 10 - 25\,MHz$
Max. CCLK = 60 MHz

So Max. MSEL = 6 for 214x

Table - PLL Divider values

PSEL Bits (PLLCFG bits [6:5])	Value of P
00	1
01	2
10	4
11	8



* STEP 3 - PLL0FEED = 0xAA ; PLL0FEED = 0x55;

Table - PLL Feed register (PLL0FEED - address 0xE01F C08C, PLL1FEED - address 0xE01F C0AC) bit description

Bit	Symbol	Description	Reset value
7:0	PLLFEED	The PLL feed sequence must be written to this register in order for PLL configuration and control register changes to take effect.	0x00

* STEP 4 - while ((PLL0STAT & 0x00000400) == 0) ; PLOCK = 0 ;

It can also be written as while (!(PLL0STAT & 0x00000400)).

→ PLL status Register (PLL0STAT) -

Table - PLL status register (PLL0STAT - address 0xE01F C088, PLL1STAT - address 0xE01F C0A8) bit description

PTO

Bit	Symbol	Description	Reset value
4:0	MSEL	Read-back for the PLL multiplier value. This is the value currently used by the PLL.	0
6:5	PSEL	Read-back for the PLL Divider value. This is the value currently used by the PLL.	0
7	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
8	PLLE	Read-back for the PLL Enable bit. When one, the PLL is currently activated. When zero, the PLL is turned off. This bit is automatically cleared when Power-down mode is activated.	0
9	PLLC	Read-back for the PLL Connect bit. When PLLC and PLLE are both one, the PLL is connected as the clock source for the microcontroller. When either PLLC or PLLE is zero, the PLL is bypassed and the oscillator clock is used directly by the microcontroller. This bit is automatically cleared when Power-down mode is activated.	0
10	PLOCK	Reflects the PLL Lock status. When zero, the PLL is not locked. When one, the PLL is locked onto the requested frequency.	0

* STEP 5 - PLLOCON = 0x03;

* STEP 6 - PLLOFEED = 0xAA; PLLOFEED = 0x55;

PTO

* STEP 7 - VPBDIV = 0x00; CCLK = 60MHz and PCLK = 15MHz;

Table - VPB divider register map

Name	Description	Access	Reset Value	Address
VPBDIV	Controls the rate of the VPB clock in relation to the processor clock.	R/W	0x00	0xE01F C100

* REGISTER SUMMARY - PLL:

	7	6	5	4	3	2	1	0
PLL0FEED	PLLFEED							
PLL0CON							PLLC	PLLE
PLL0CFG		PSEL		MSEL				
VPBDIV							VPBDIV	

	10	9	8	7	6	5	4	0
PLL0STAT	PLOCK	PLLC	PLLE		PSEL		MSEL	

LECTURE 27 - TIMER

Q. What is timer ?

A. ① Timer is fully depend upon the oscillator that attached externally to the microcontroller because it uses the frequency of oscillator to operate.

② When we trigger timer it start from initial value and run up to decided value stored by user in special function registers.

③ When it reach its maximum value, it overflows and a certain bit is decided to show overflow in SFR (special function register) also called flag bit.

④ Some timers also used prescalar technique to enhance its limits. For example,

8-bit timer : 256 (2^8)

16-bit timer : 65536 (2^16)

32-bit timer : 2^32

+ TIMER IN LPC2148 :

① The LPC2148 has two functionally identical general purpose timers, Timer0 and Timer1. Both timer/counter with a programmable 32-bit prescaler; counter/timer operation.

② Up to four 32-bit capture channels per timer, that can take a snapshot of the timer value when an input signal transitions. A capture event may also optionally generate an interrupt.

③ Four 32-bit match registers that allow -

(i) Continuous operation with optional interrupt generation on match.

(ii) Stop timer on match with optional interrupt generation.

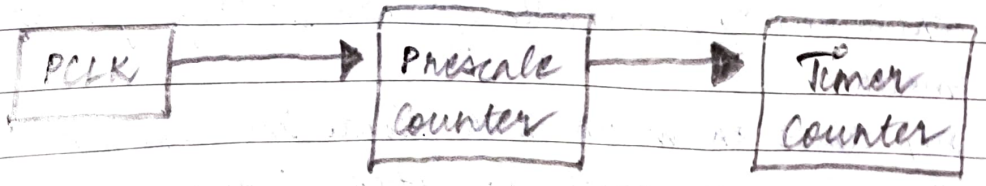
(iii) Reset timer on match with optional interrupt generation.

(4) Up to four external outputs corresponding to match registers, with the following capabilities -
(i) Set low on match.
(ii) Set high on match.
(iii) Toggle on match.
(iv) Do nothing on match.

* CAPTURE REGISTER:
As the name suggests it is used to capture input signal. When a transition event occurs on a capture pin, it can be used to copy the value of TC into any of the 4 capture Register or to generate an interrupt. Hence, these can be also used to demodulated PWM signals.

* MATCH REGISTER:
A match register is a register which contains a specific value set by the user. When the timer starts, every time after TC is incremented, the value in TC is compared with match register. If it matches then it can reset the timer or can generate an interrupt as defined by the user. We are only concerned with match registers in this lecture.

Q. How timers in LPC2148 ARM7 microcontroller works?
A. The heart of timers of the LPC2148 microcontroller is a 32-bit free running counter, which is designed to count cycles of the peripheral clock (PCLK) or an external clock, this counter is programmable with 32-bit prescaler.



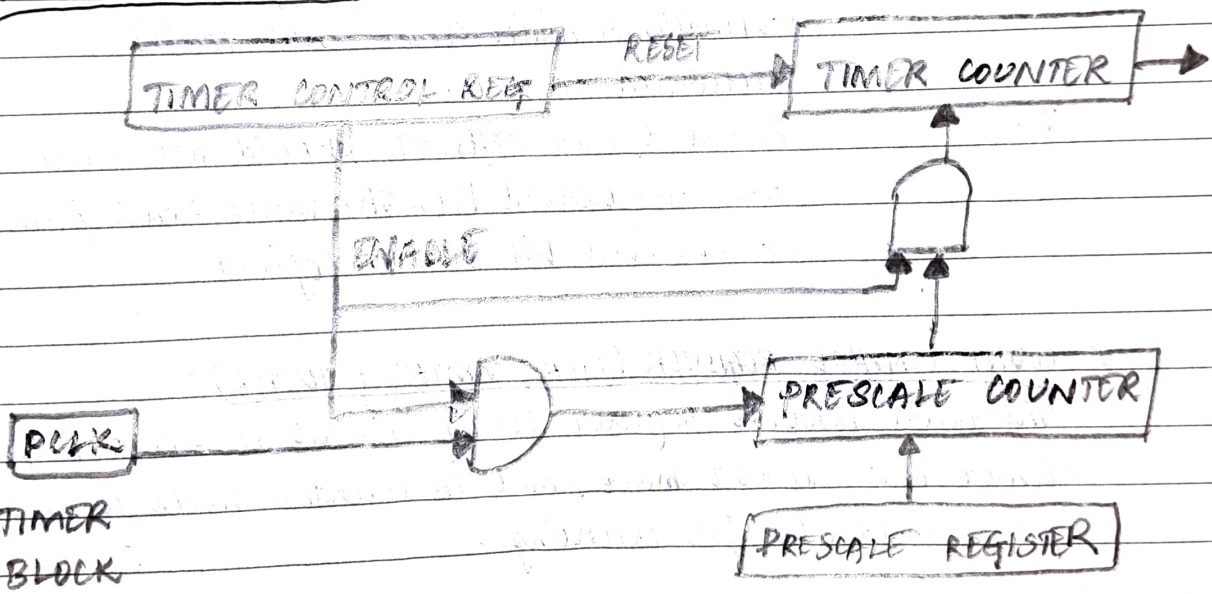
The tick rate of the timer counter (TC) is controlled by the 32-bit number written in the prescale register (PR) in the following way-

(i) There is a prescale counter (PC) which increments on each tick of the PCLK.

(ii) When it reaches the value in the prescaler register, the timer count is incremented and the prescaler counter (PC) is reset, on the next PCLK.

(iii) This cause the timer counter to increment on every PCLK when PR=0, every 2 PCLKs when PR=1, etc.

* LPC2148 TIMER BLOCK:



The prescale counter is incremented on every PCLK. When prescale counter reaches the value stored in the prescale register, the timer counter is incremented and the prescale counter is reset on the next PCLK.

The match register values are continuously compared to the timer counter value, when the two values are equal, actions (match pin / interrupt) can be triggered automatically.

* TIMER CONTROL REGISTER (TCR, TIMER0 : T0TCR) :

Timer control register used to control the timer control functions. We'll enable, disable and reset timer counter (TC) through this register.

Bit	Symbol	Description	Reset value
0	Counter Enable	When one, the timer counter and prescale counter are enabled for counting. When zero, the counters are disabled.	0
1	Counter Reset	When one, the timer counter and the prescale counter are synchronously reset on the next positive edge of PCLK. The counters remain reset until TCR[1] is returned to zero.	0
7:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

* COUNT CONTROL REGISTER (CTCR, TIMER0 : T0CTCR) :

The count control register (CTCR) is used to select between timer and counter mode, and in counter mode to select the pin and edge(s) for counting.

Bit	Symbol	Value	Description
1:0	Counter/ Timer Mode		This field selects which rising PCLK edges can increment timer's prescale counter (PC), or clear PC and increment timer counter (TC).
		00	Timer Mode: every rising PCLK edge
		01	Counter Mode: TC is incremented on rising edges on the CAP input selected by bits 3:2
		10	Counter Mode: TC is incremented on falling edges on the CAP input selected by bits 3:2.

		11	Counter Mode: TC is incremented on both edges on the CAP input selected by bits 3:2.

* **TIMER COUNTER (TC, TIMER0 : T0CR) :**
This is the main counting register. Timer counter increments when PC reaches its maximum value as specified by PR. If timer is not reset explicitly (directly) or by using an interrupt then it will act as a free running counter which resets back to zero when it reaches its maximum value which is 0xFFFF FFFF.

* **PRESCALE REGISTER (PR, TIMER0 : T0PR):**
The 32-bit prescale register specifies the maximum value for the prescale counter.

* **PRESCALE COUNTER REGISTER (PC, TIMER0 : T0PC):**
- This register increments on every PCLK (peripheral clock). This register controls the resolution of the timer. When PC reaches the value in PR, PC is reset back to 0 and timer counter is incremented by 1.
- Hence if PR=0 then timer counter increments on every 1 PCLK. If PR=9 then timer counter increments on every 10th cycle of PCLK. Hence, by selecting an appropriate prescale value we can control the resolution of the timer.

* **MATCH REGISTERS (MR0 - MR3):**
The match register values are continuously compared to the timer counter value. When the two values are equal, actions are triggered automatically. The action possibilities are to generate an interrupt, reset the timer counter, or stop the timer. Actions are controlled by the settings in the MCR register.

* **MATCH CONTROL REGISTER: (MCR, TIMER0: T0MCR)**

This register is used to control which all operations can be done when the value in MR matches the value in TC. Bits 0, 1, 2 are for MR0, Bits 3, 4, 5 for MR1 and so on. For MR0 (Match Register 0):-

① Bit 0 - Interrupt on MR0 i.e. trigger an interrupt when MR0 matches TC. Interrupts are enabled when set to 1 and disabled when set to 0.

② Bit 1 - Reset on MR0. When set to 1, TC will be reset when it matched MR0. Disabled when set to 0.

③ Bit 2 - Stop on MR0. When set to 1, TC and PC will stop when MR0 matches TC.

④ Similarly bits 3-5, 6-8, 9-11 are for MR1, MR2, MR3 respectively.

* **INTERRUPT REGISTER (IR, TIMER0: T0IR):**

The interrupt register consists of four bits for the match interrupts and four bits for the capture interrupts.

Bit	Symbol	Description
0	MR0 Interrupt	Interrupt flag for match channel 0.
1	MR1 Interrupt	Interrupt flag for match channel 1.
2	MR2 Interrupt	Interrupt flag for match channel 2.
3	MR3 Interrupt	Interrupt flag for match channel 3.
4	CR0 Interrupt	Interrupt flag for capture channel 0 event.
5	CR1 Interrupt	Interrupt flag for capture channel 1 event.
6	CR2 Interrupt	Interrupt flag for capture channel 2 event.
7	CR3 Interrupt	Interrupt flag for capture channel 3 event.

* **PRESCALE (TxPR) RELATED CALCULATIONS:**

The delay or time required for 1 clock cycle at 'x' MHz is given by :- $\frac{1}{x \times 10^6}$ seconds.

Hence, in our case when PR=0 i.e. TC increments at every PCLK the delay required for TC to increment by 1 is-

$$\frac{0+1}{60 \times 10^6} \text{ seconds}$$

Similarly when we set PR = 59999, the delay in this case will be-

$$\frac{59999+1}{60 \times 10^6} = \frac{60000}{60 \times 10^6} = \frac{1000}{10^6} = \frac{1}{10^3} \text{ seconds}$$

which boils down to 0.001 seconds which is nothing but 1 milli-second i.e. ms. Hence, the delay required for TC to increment by 1 will be 1 ms.

* **BASIC STEP FOR TIMER :**

1. Set appropriate value in TxCTCR.
2. Define the prescale value in TxPR.
3. Set values in match register(s), if required.
4. Set appropriate value in TxMCR if using match register / interrupts.
5. Reset timer, which resets PR and TC.
6. Set TxTCR to 0x01 to enable the timer when required.
7. Reset TxTCR to 0x00 to disable the timer when required.

Note - x denotes which timer we want to use.

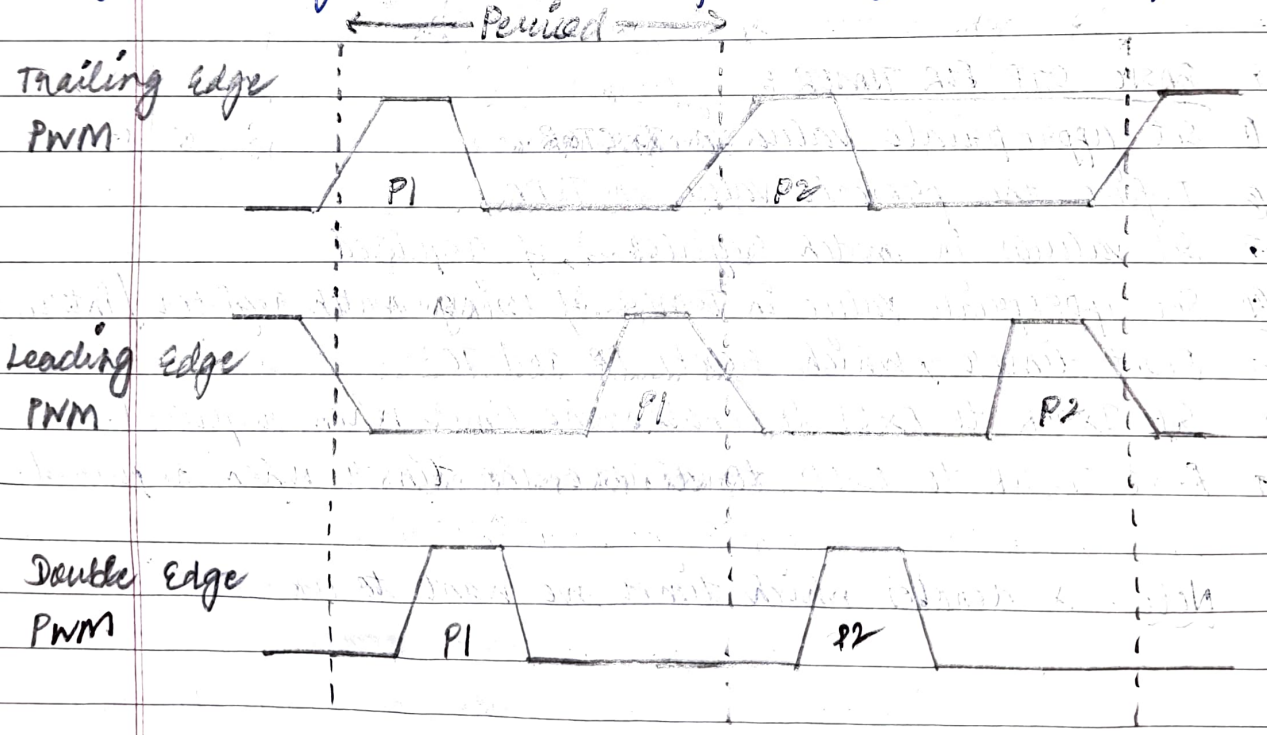
LECTURE 28 - NXP's LPC2148 PWM

PWM is used for generating pulses, which in turn is used for motor control. There are 6 PWM channels available in LPC2148 namely, P0.0 - PWM1, P0.1 - PWM3, P0.7 - PWM2, P0.8 - PWM4, P0.9 - PWM6 and P0.21 - PWM5.

★ **PWM TYPES:**

① Single Edged PWM - Pulse at the beginning or the ending of the period, trailing edge PWM or leading edge PWM respectively.

② Double Edge PWM - Pulse can be placed anywhere in the period.

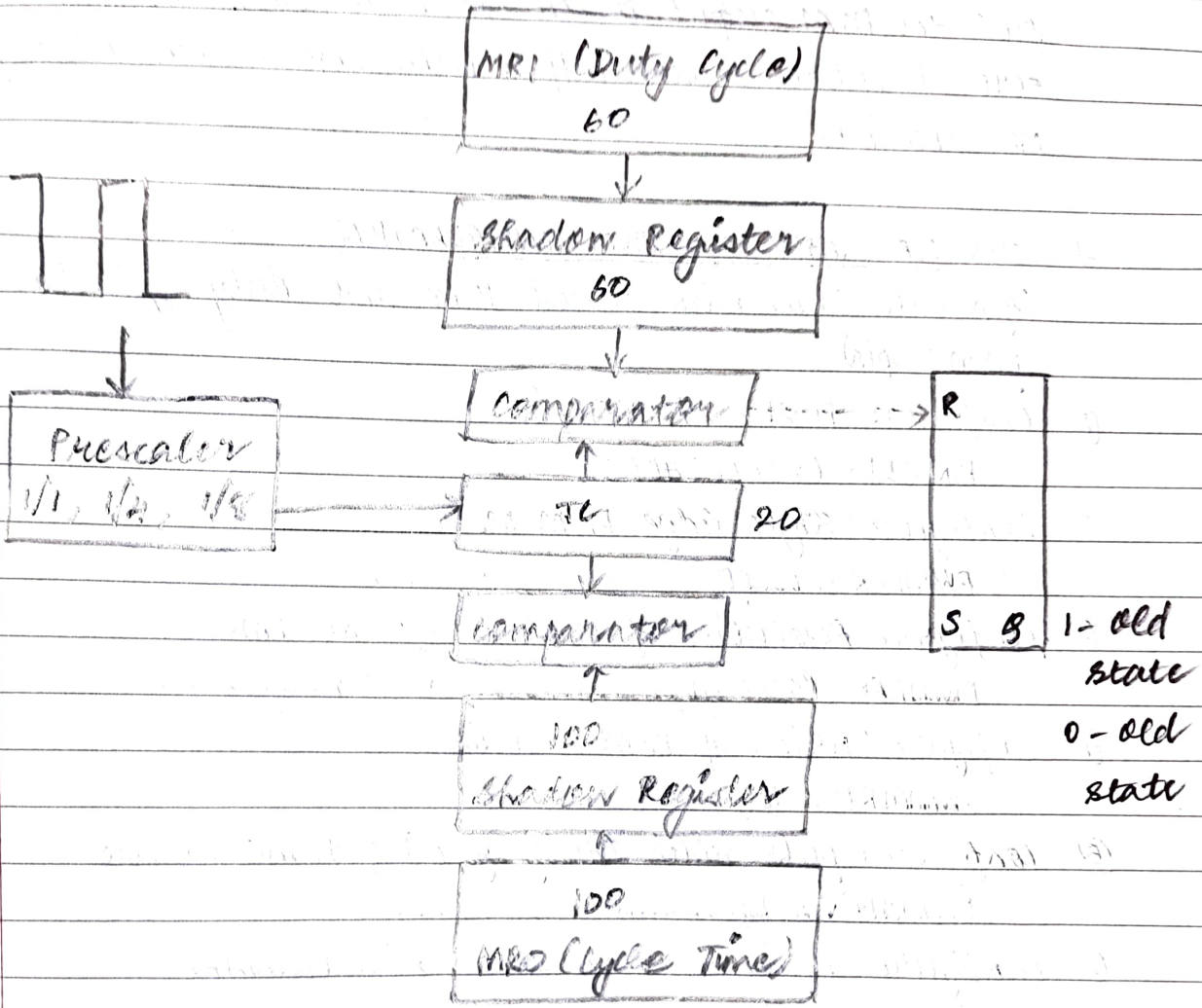


★ **PWM OPERATION:**

→ TC matches MR1-6 (Duty Cycle)
- PWM pin pulled Low.

→ TC matches MR0 (PWM Time Period)
- Reset the TC value.
- Pull the PWM pin High.

- Latches new match register values (Both Time Period - MR0 and Duty Cycle - MR1-6).



* PWM OPERATION - SUMMARY :

① The TC is being incremented as per the pre-scalar configuration. The PWM output is high as the TC is still less that duty cycle.

② TC is incremented to 40 and still the PWM pin as HIGH.

③ TC is incremented to 60 and it matches the Duty Cycle (MR1=60).

④ Now the comparator1 (green) will trigger the R of SR latch and pulls the PWM output to zero (Q = 0). TC still continues to increment.

⑤ TC is incremented to 80 and PWM pin is low as TC > Duty cycle.

⑥ Now TC is 100 and it matches the cycle time (MR0 == 100).

⑦ Now, the comparator 2 (Red) will trigger the S of SR latch and pulls the PWM output to ONE (Q == 1). It also resets the TC to zero. It updates shadow buffers with new match values from MR0 and MR1.

* **STEPS FOR SINGLE EDGE PWM CONFIGURATION:**

(generate PWM with period 10 ms and duty cycle 1 ms in PWM5 pin)

① Configure P0.21 as PWM5 pin
```
PINSEL1 = 0x00000400;
```
② Configure Single Edge PWM Mode
```
PWMPCR = 0x00;
```
③ Configure Resolution of PWM is set at 1 ms
```
PWMPR = 60000 - 1;
```
④ Configure period of PWM is 10 ms
```
PWMMR0 = 10;
```
⑤ Configure pulse width (duty cycle) of PWM5 is 1 ms
```
PWMMR5 = 1;
```
⑥ Configure TC to reset on match with PWMMR0
```
PWMMCR = (1 << 1);
```
⑦ Update match registers PWMMR0 and PWMMR5
```
PWMLER = (1<<5) | (1<<0);
```
⑧ Enable PWM5 output
```
PWMPCR = (1<<13);
```
⑨ Reset PWM TC and PWM PR
```
PWMTCR = (1<<1);
```
⑩ Enable PWM timer counters and PWM mode (Also Counter Reset = 0)
```
PWMTCR = (1<<0) | (1<<3);
```

* STEP 1 - PINSEL1 = 0x00000400;

Bit	Symbol	Value	Function
11:10	P0.21	00	GPIO Port 0.21
		01	PWM5
		10	Reserved or AD1.6
		11	Capture 1.3 (Timer 1)

* STEP 2 - PWMPCR = 0x00;

Table - PWM control Register (PWMPCR - address 0xE001 404C) bit description

Bit	Symbol	Value	Description	Reset Value
1:0	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
2	PWMSEL2	1	selects double edge controlled mode for the PWM2 output.	0
		0	selects single edge controlled mode for PWM2.	
3	PWMSEL3	1	Selects double edge controlled mode for the PWM3 output.	0
		0	Selects single edge controlled mode for PWM3.	
4	PWMSEL4	1	Selects double edge controlled mode for the PWM4 output.	0
		0	Selects single edge controlled mode for PWM4.	
5	PWMSEL5	1	Selects double edge controlled mode for the PWM5 output.	0

		0	selects single edge controlled mode for PWM6.	
6	PWMSEL6	1	selects double edge controlled mode for the PWM6 output.	0
		0	selects single edge controlled mode for PWM6.	

* STEP 3 - PWMPR = 60000 - 1;

PWM block derives its clock from the peripheral clock (PCLK) - 60 MHz. Time taken for one PCLK cycle is $\frac{1}{60 \times 10^6}$.

So, time taken for 'x' PCLK cycles at 60MHz is $\frac{x}{60 \times 10^6}$.

If the prescale is considered, then x = PC = PR + 1.

If prescale register value is 59,

Delay = $\frac{(59+1)}{60 \times 10^6}$ = 1 µs.

If prescale register value is 59999,

Delay = $\frac{(59999+1)}{60 \times 10^6}$ = 1 ms.

* STEP 4 - PWMMRD = 10;

PWMMRD - PWM time period for all PWM channels

* STEP 5 - PWMMR5 = 1;

PWMMR1 - 6 - PWM duty cycle for each PWM channels respectively.

PTD

* STEP 6 - PWMMCR = (1<<1) ;

Table - Match Control Register (MCR, TIMER 0 : TOMCR - address 0xE000 4014 and TIMER 1 : TIMCR - address 0xE000 8014) bit description

Bit	Symbol	Value	Description	Reset Value
0	PWMMR0I	1	Interrupt on PWMMR0: an interrupt is generated when PWMMR0 matches the value in the PWMTC.	0
		0	This interrupt is disabled.	
1	PWMMR0R	1	Reset on PWMMR0: the PWMTC will be reset if PWMMR0 matches it.	0
		0	This feature is disabled.	
2	PWMMR0S	1	Stop on PWMMR0: the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR0 matches the PWMTC.	0
		0	This feature is disabled.	
3	PWMMR1I	1	Interrupt on PWMMR1: an interrupt is generated when PWMMR1 matches the value in the PWMTC.	0
		0	This interrupt is disabled.	
4	PWMMR1R	1	Reset on PWMMR1: the PWMTC will be reset if PWMMR1 matches it.	0
		0	This feature is disabled.	
5	PWMMR1S	1	Stop on PWMMR1: the PWMTC and PWMPC will be stopped and PWMTCR[0] will be set to 0 if PWMMR1 matches the PWMTC.	0
		0	This feature is disabled.	

PTO

* STEP 7 - PWMLER = (1<<5) | (1<<0);

Table - PWM Latch Enable Register (PWMLER - address 0x E001 4050) bit description

Bit	Symbol	Description	Reset Value
0	Enable PWM Match 0 Latch	Writing a one to this bit allows the last value written to the PWM Match 0 register to be become effective when the timer is next reset by a PWM Match event.	0
1	Enable PWM Match 1 Latch	Writing a one to this bit allows the last value written to the PWM match 1 register to be become effective when the timer is next reset by a PWM match event.	0
2	Enable PWM Match 2 Latch	Writing a one to this bit allows the last value written to the PWM match 2 register to be become effective when the timer is next reset by a PWM match event.	0
5	Enable PWM match 5 Latch	Writing a one to this bit allows the last value written to the PWM Match 5 register to be become effective when the timer is next reset by a PWM match event.	0
6	Enable PWM Match 6 Latch	Writing a one to this bit allows the last value written to the PWM match 6 register to be become effective when the timer is next reset by a PWM match event.	0

* STEP 8 - PWMPCR = (1<<13);

Table - PWM Control Register (PWMPCR - address 0x E001 404C) bit registration

PTO

Bit	Symbol	Value	Description	Reset value
9	PWMENA1	1	The PWM1 output enabled.	0
		0	The PWM1 output disabled.	
10	PWMENA2	1	The PWM2 output enabled.	0
		0	The PWM2 output disabled.	
11	PWMENA3	1	The PWM3 output enabled.	0
		0	The PW3 output disabled.	
12	PWMENA4	1	The PWM4 output enabled.	0
		0	The PWM4 output disabled.	
13	PWMENA5	1	The PWM5 output enabled.	0
		0	The PWM5 output disabled.	
14	PWMENA6	1	The PWM6 output enabled.	0
		0	The PWM6 output disabled.	

* STEP 9 - PWMTCR = (1 <<1);

Table - PWM Timer Control Register (PWMTCR - address 0xE001 4004) bit description

Bit	Symbol	Description	Reset Value
0	Counter Enable	When one, the PWM Timer Counter and PWM Prescale Counter are enabled for counting. When zero, the counters are disabled.	0
1	Counter Reset	When one, the PWM Timer Counter and the PWM Prescale Counter are synchronously reset on the next positive edge of PCLK. The counters remain reset until TCR[1] is returned to zero.	0
2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

3	PWM Enable	When one, PWM mode is enabled. PWM mode causes shadow registers to operate in connection with the match registers. A program write to a Match register will not have an effect on the match result until the corresponding bit in PWMLER has been set, followed by the occurrence of a PWM Match 0 event. Note that the PWM Match register that determines the PWM rate (PWM match 0) must be set up prior to the PWM being enabled. otherwise a match event will not occur to cause shadow register contents to become effective.	0

* STEP 10 - PWMTCR = (1<<0) | (K<3);

LECTURE 29 - LCD INTERFACING

* LCD PIN:

Pin Number	Name	Description
1	VSS	Power supply (GND)
2	VCC	Power supply (+5V)
3	VEE	Contrast adjust
4	RS	0 = Instruction input 1 = Data input
5	R/W	0 = Write to LCD module 1 = Read from LCD module
6	EN	Enable signal
7	D0	Data bus line 0 (LSB)
8	D1	Data bus line 1
9	D2	Data bus line 2
10	D3	Data bus line 3
11	D4	Data bus line 4
12	D5	Data bus line 5
13	D6	Data bus line 6
14	D7	Data bus line 7 (MSB)

* LCD COMMANDS:

Table - Frequently used commands and instructions for LCD.

No.	Instruction	Hex	Decimal
1	Function set: 8-bit, 1 Line, 5x7 Dots	0x30	48
2	Function set: 8-bit, 2 Line, 5x7 Dots	0x38	56
3	Function set: 4-bit, 1 Line, 5x7 Dots	0x20	32
4	Function set: 4-bit, 2 Line, 5x7 Dots	0x28	40
5	Entry mode	0x06	6

6	Display off Cursor off (clearing display without clearing DDRAM content)	0x08	8
7	Display on Cursor on	0x0E	14
8	Display on Cursor off	0x0C	12
9	Display on Cursor blinking	0x0F	15
10	Shift entire display left	0x18	24
11	Shift entire display right	0x1C	30
13	Move cursor left by one character	0x10	16
14	Move cursor right by one character	0x14	20
15	Clear Display (also clear DDRAM content)	0x01	1
16	Set DDRAM address or cursor position on display	0x80 + add*	128 + add*
M	Set CGRAM address or set pointer to CGRAM location	0x40 + add**	64 + add**

* STEPS TO PRINT A CHARACTER - LCD:

① Set P0.12 to P0.15 LCD Data (D4 to D7). P0.4, 5, 6 as RS, RW and EN as O/P

```
IO0DIR = 0x0000FFF0;
```

LCD Initialize

② Set the Function Set (4-bit interface, two line, 5x8 dots)

```
LCD_CMD (0x28);
```

③ Clear LCD Display

```
LCD_CMD (0x01);
```

④ Initialize Cursor to Home Position

```
LCD_CMD (0x02);
```

⑤ Set the Entry Mode (Cursor Increment, Display shift off)

```
LCD_CMD (0x06);
```

⑥ Display on, Cursor off and Blink off

```
LCD_CMD (0x0C);
```

⑦ Set the Cursor to First Line or Second Line

```
LCD_CMD (0x80 + P00); Or LCD_CMD (0xC0 + P00);
```

⑧ Display the character

```
LCD_CHAR (c);
```

* STEP 1 - IODIR = 0x0000FFF0;

* STEP 2 - LCD_CMD (0x28);

Function set Command

Instruction	Instruction Code										Description
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
Function set	0	0	0	0	1	DL	N	F	-	-	Set interface data length (DL: 8-bit/4-bit), numbers of display line (N: 2-line/1-line) and, display font type (F: 5×11 dots/5×8 dots)

0	0	1	0	1	0	0	0
2				8			

* STEP 3 - LCD_CMD (0x01);

Clear Display Command

Instruction	Instruction code										Description
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
Clear Display	0	0	0	0	0	0	0	0	0	1	Write "00H" to DDRAM and set DDRAM address to "00H" from AC.

0	0	0	0	0	0	0	1
0				1			

* STEP 4 - LCD_CMD (0x02);

Cursor Home Command

P70

Instruction	Instruction Code										Description
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
Return Home	0	0	0	0	0	0	0	0	1	–	Set DDRAM address to "00H" from AC and return cursor to its original position if shifted. The contents of DDRAM are not changed.

0	0	0	0	0	0	1	0
0				2			

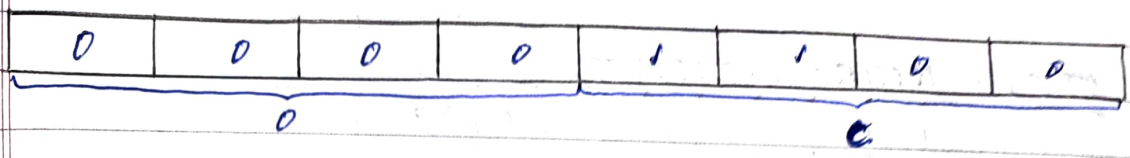
* STEP 5 - LCD_CMD (0x06);
Entry mode set command

Instruction	Instruction Code										Description
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	SH	Assign cursor moving direction and enable the shift of entire display.

0	0	0	0	0	1	1	0
0				6			

* STEP 6 - LCD_CMD (0x0C);
Display control command

Instruction	Instruction Code										Description
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	Set display (D), cursor (C) and blinking of cursor (B) on/off control bit.



* STEP 7 - LCD_CMD (0x80 + Pos); (or) LCD_CMD (0xC0 + Pos);

* STEP 8 - LCD_CHAR (c);

* LCD COMMAND WRITE (4-BIT):

① Set LCD_RS = 0 & LCD_RW = 0
```
IO0CLR = 0x00000030;
```
② Split MSB 4-Bits of LCD Command
```
cmd_MSB = cmd & 0xF0;
cmd_MSB = cmd_MSB >> 4;
```
③ Move MSB nibble to PORT0 - P0.15 to P0.12
```
IO0PIN = cmd_MSB;
```
④ Give LCD Enable signal
```
IO0SET = 0x00000040; // EN = 1
delay_ms(5);
IO0CLR = 0x00000040; // EN = 0
```
⑤ Split LSB 4-Bits of LCD Command
```
cmd_LSB = cmd & 0x0F;
cmd_LSB = cmd_LSB << 12;
```
⑥ Move LSB nibble to PORT0 - P0.15 to P0.12
```
IO0PIN = cmd_LSB
```
⑦ Give LCD Enable signal
```
IO0SET = 0x00000040; // EN = 1
delay_ms(5);
IO0CLR = 0x00000040; // EN = 0
```

* STEP 1 - IOCLR = 0x00000030;

* STEP 2 - cmd_MSB = cmd & 0xF0;
cmd_MSB = cmd_MSB << 8;

cmd 0100 1001
&0xF0 1111 0000
cmd-MSB 0100 0000

cmd_MSB 0000 0000 0100 0000
<<8
cmd_MSB 0100 0000 0000 0000

* STEP 3 - IOPIN = cmd_MSB;

PORT 0
15 14 13 12
IOPIN = 0100 0000 0000 0000

* STEP 4 - IOSET = 0x00000040; // EN=1
delay_ms(5);
IOCLR = 0x00000040; // EN=0

* STEP 5 - cmd_LSB = cmd & 0x0F;
cmd_LSB = cmd_LSB << 12;

cmd 0100 1001
&0x0F 0000 1111
cmd-LSB 0000 1001

cmd_LSB 0000 0000 0000 1001
<< 12
cmd_LSB 1001 0000 0000 0000

* STEP 6 - IOPIN = cmd_LSB;

PORT 0
15 14 13 12
IOPIN = 1001 0000 0000 0000

* STEP 7 - IOOSET = 0x00000040; // EN=1
  delay_ms(5);
  IOOCLR = 0x00000040; // EN=0

* LCD DATA WRITE (4-BIT):

① Set LCD_RS = 1 & LCD_RW = 0
  IOOSET = 0x00000010; // RS=1
  IOOCLR = 0x00000020; // RW=0

② Split MSB 4-Bits of LCD command
  data_MSB = data & 0xF0;
  data_MSB = data_MSB << 8;

③ Move MSB nibble to PORT0 - P0.15 to P0.12
  IOOPIN = data_MSB;

④ Give LCD Enable signal
  IOOSET = 0x00000040; // EN=1
  delay_ms(5);
  IOOCLR = 0x00000040; // EN=0

⑤ Split LSB 4-Bits of LCD command
  data_LSB = data & 0x0F;
  data_LSB = data_LSB << 12;

⑥ Move LSB nibble to PORT0 - P0.15 to P0.12
  IOOPIN = data_LSB;

⑦ Give LCD Enable signal
  IOOSET = 0x00000040; // EN=1
  delay_ms(5);
  IOOCLR = 0x00000040; // EN=0

* STEP 1 - IOOSET = 0x00000010; // RS=1
  IOOCLR = 0x00000020; // RW=0

* STEP 2 - data_MSB = data & 0xF0;
  data_MSB = data_MSB << 8;

data 0100 1001 data_MSB 0000 0000 0100 0000
0xF0 1111 0000 << 8
data_MSB 0100 0000 data_MSB 0100 0000 0000 0000

* STEP 3 - IODPIN = data_MSB;

PORTD
15 14 13 12
IODPIN = 0100 0000 0000 0000

* STEP 4 - IOSET = 0x00000040; //EN=1
  delay_ms(5);
  IOCLR = 0x00000040; //EN=0

* STEP 5 - data_LSB = data & 0x0F;
  data_LSB = data_LSB << 12;

data 0100 1001 data_LSB 0000 0000 0000 1001
&0x0F 0000 1111 << 12
data_LSB 0000 1001 data_LSB 1001 0000 0000 0000

* STEP 6 - IODPIN = data_LSB;

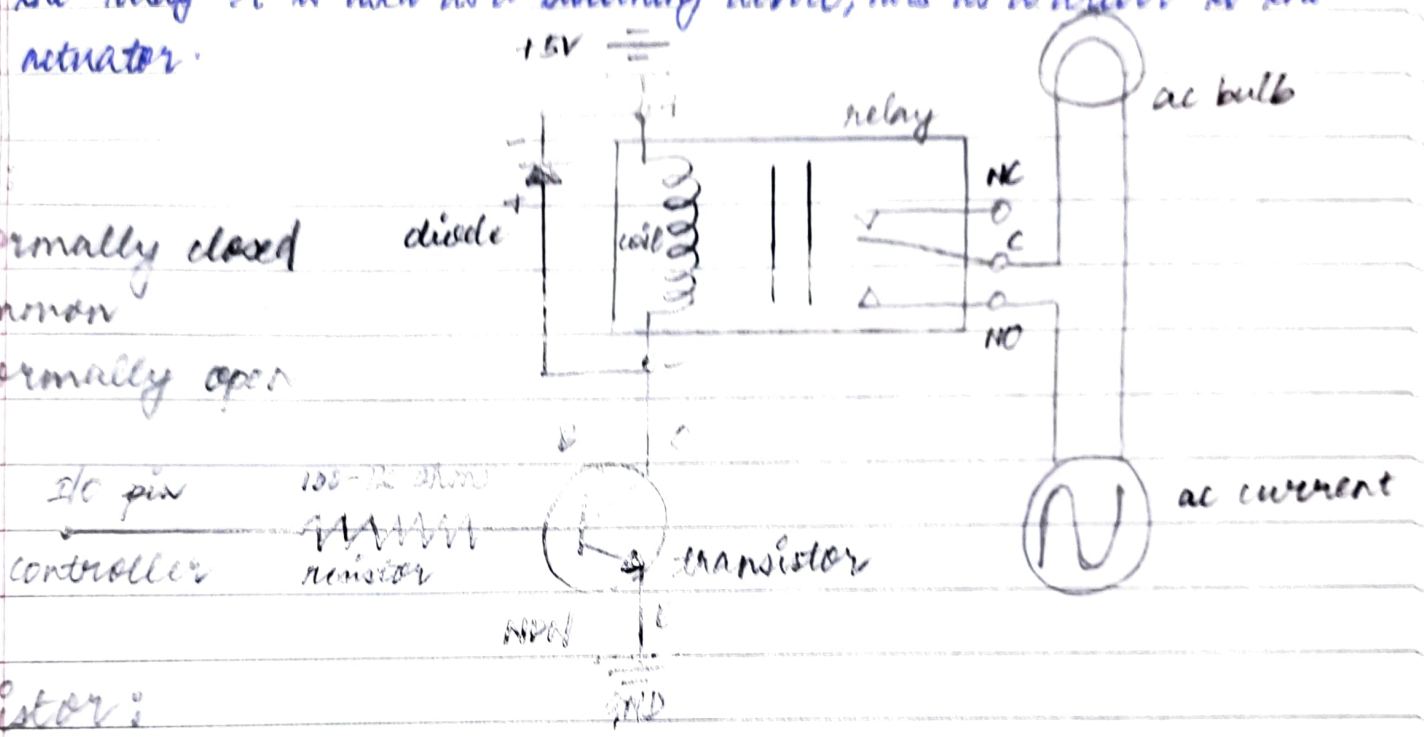
PORTD
15 14 13 12
IODPIN = 1001 0000 0000 0000

* STEP 7 - IOSET = 0x00000040; //EN=1
  delay_ms(5);
  IOCLR = 0x00000040; // EN=0

LECTURE 30 - RELAY INTERFACING

Relay is an electromechanical device, having an electromagnet inside the relay. It is used as a switching device, acts as a driver to the actuator.

relay:
NC = normally closed
C = common
NO = normally open



transistor:
B = base
C = collector
E = transmitter

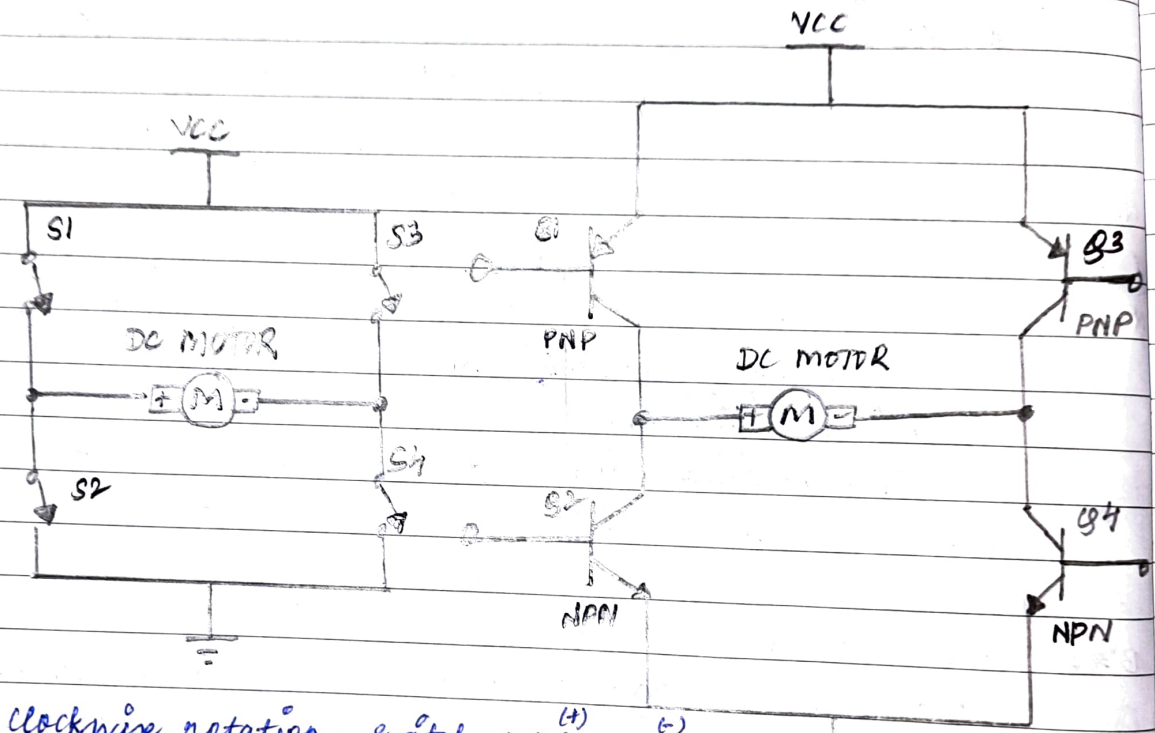
Figure - circuit for controlling an AC or other high-current device from a microcontroller by using a relay

When the coil is energized, the pole will come in contact with normally open and the bulb will glow. Hence, the common acts as a switch. The coil is switch ON using a transistor.

The DC source is set as the same voltage of the relay, i.e. 5V. The diode is used to remove the back EMF.

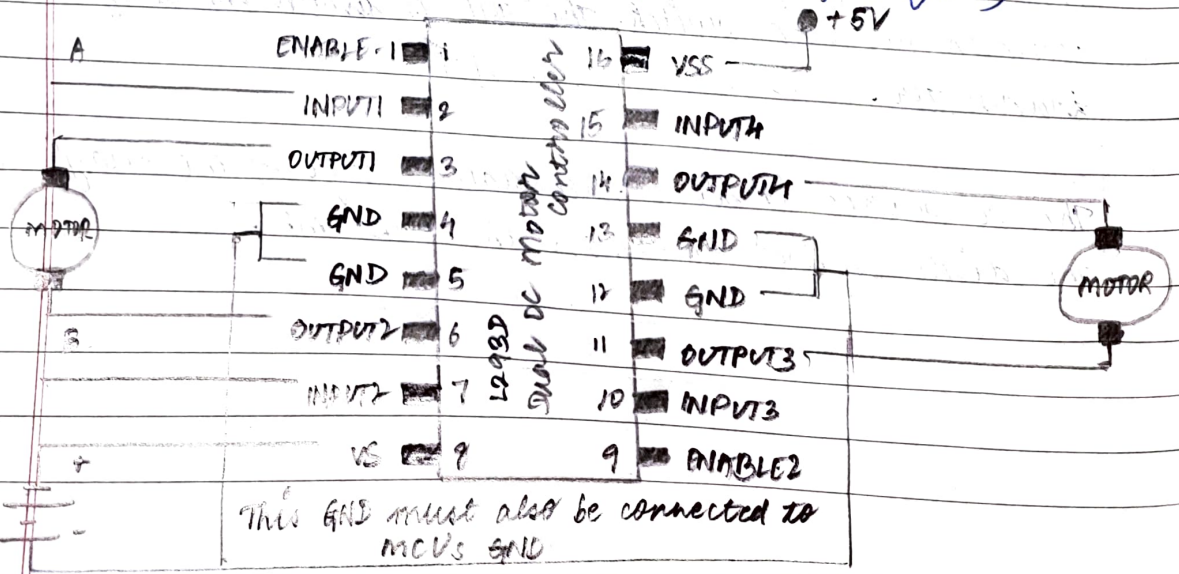
LECTURE 31 - DC MOTOR CONTROL

In the earlier lecture, we had controlled the motor using a relay. Here, we'll control the direction of the DC motor, using H Bridge



Clockwise rotation - Switch ON $S1_{(+)}$ & $S4_{(-)}$.
Anti-clockwise rotation - Switch ON $S3_{(-)}$ & $S2_{(+)}$.

Clockwise rotation - Q1 (Logic 0) & Q4 (Logic 1).
Anti-clockwise rotation - Q3 (Logic 0) & Q2 (Logic 1).



There are two H bridges which can be used to control two DC motors. All the transistors mentioned earlier are embedded in this IC. This can control the direction of two DC motors.

Pin Number 16 (VSS) acts as Vcc to the IC. This is programmed as 5V since the IC works on 5V voltage supply, not 3.3V as in the earlier lectures. Four grounds are present in this IC, all of which are shorted.

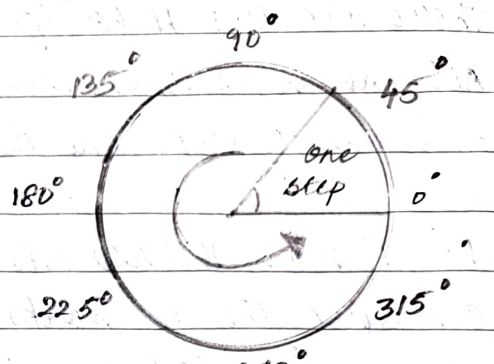
Pin Number 8 (VS) is the source voltage that has to be applied to the motor. Since we are using 5V motor, we are equating VS to 5V. Input 1 and 2 are used to give input to motor 1. Similarly, input 3 and 4 are used to give input to motor 2.

The first motor is connected across output 1 and 2. Similarly, the second motor is connected across output 3 and 4. To rotate the first motor in clockwise direction, we need to give 1 and 0 as input to input 1 and 2 respectively. To rotate it in the anticlockwise direction, we need to give 0 and 1 as input to input 1 and 2 respectively.

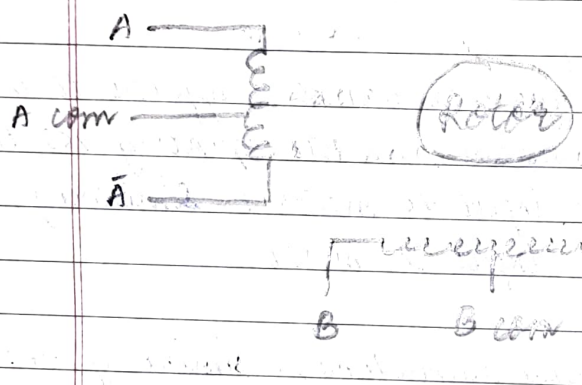
To rotate the second motor in clockwise direction, we need to give 1 and 0 as input 4 and 3 respectively. To rotate it in the anticlockwise direction, we need to give 0 and 1 as input to input 4 and 3 respectively.

LECTURE 32 - STEPPER MOTOR CONTROL

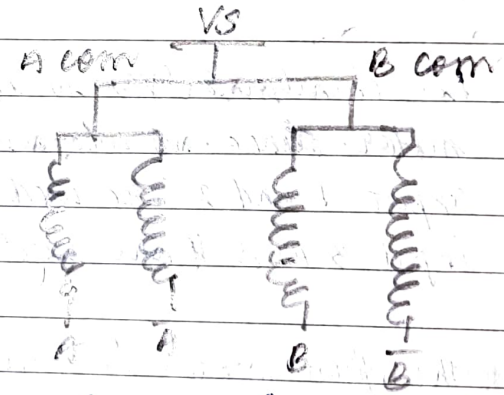
* **STEP ANGLE:**
Step Angle = 45°
Number of steps required to complete a rotation = 8



* **UNIPOLAR STEPPER:**

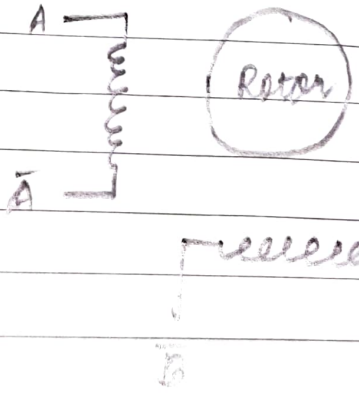


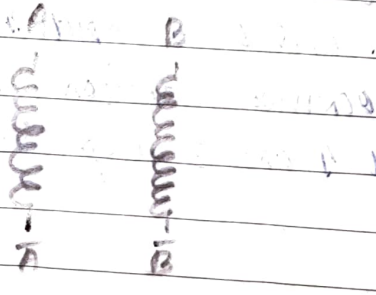
Conceptual Unipolar stepper motor Diagram



This popular configuration is used where, motor supply is connected to centres of both winding and ground to each winding end is controlled through motor driver circuit and microcontroller.

* **BIPOLAR STEPPER:**



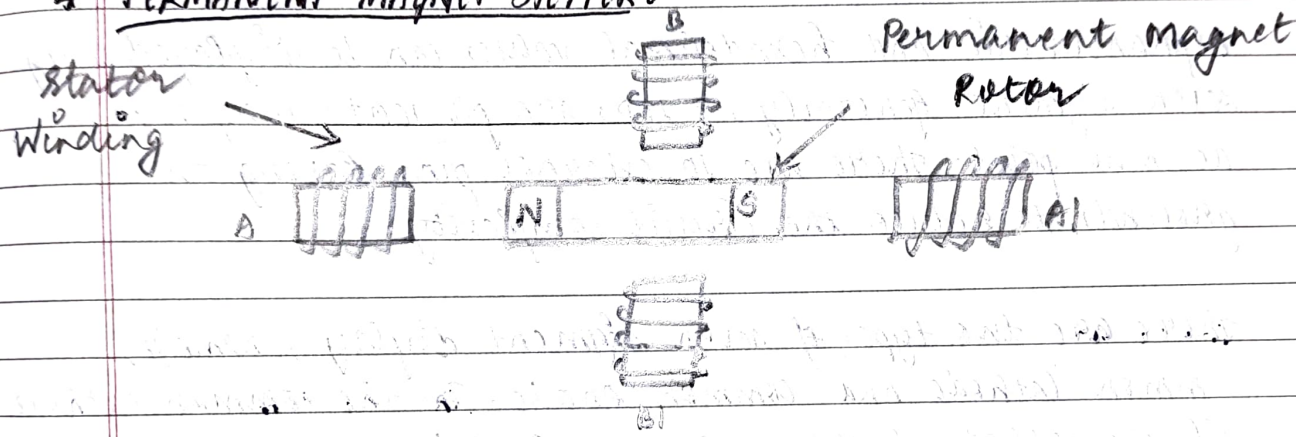


PTO

Conceptual Bipolar stepper Motor Diagram

This configuration requires control of current direction flow through each winding to control alteration of magnetic poles on winding required to attract and ripple respective rotor poles.

* PERMANENT MAGNET STEPPER:



Stepper motor is a brushless DC motor that divides the full rotation angle of 360° into number of equal steps. The motor is rotated by applying a certain sequence of control signals. The speed of rotation can be changed by changing the rate at which the control signals are applied.

Various stepper motors with different step angles and torque ratings are available in the market. Microcontroller can be used to apply different control signals to the motor to make it rotate according to the need of the application.

To find winding coils and their centre tap leads, measure resistance in between the leads. From centre leads we will get half the resistance value as compared to the resistance between winding ends.

LECTURE 33 - SEVEN SEGMENT DISPLAY

Since seven segment can withstand harsh conditions, it is still continued to be used in many places. Here, the contrast is more compared to LCD. Voltage / current displays use seven segment and they are available at different sizes in the market.

Alphanumeric and hexadecimal values can be displayed using seven segment. Generally, 8 LEDs are present along with a decimal point. There are 10 external pins having two grounds to reduce the circuit complexity.

There are two types of seven segment display, namely, common cathode and common anode. In the common cathode, all the cathode of the 8 LEDs are shorted together and taken out as a ground pin. Whereas in the common anode, all the anode of the 8 LEDs are shorted together and taken out as VCC pin.

To make a particular segment ON, write 1 to the corresponding pin in case of common cathode. Whereas, this is opposite in the case of common anode. Since VCC (+5V) is connected, to make a particular segment ON, write 0 to the corresponding pin.

In Proteaus, decimal point is not available.

	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	10PINO
	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