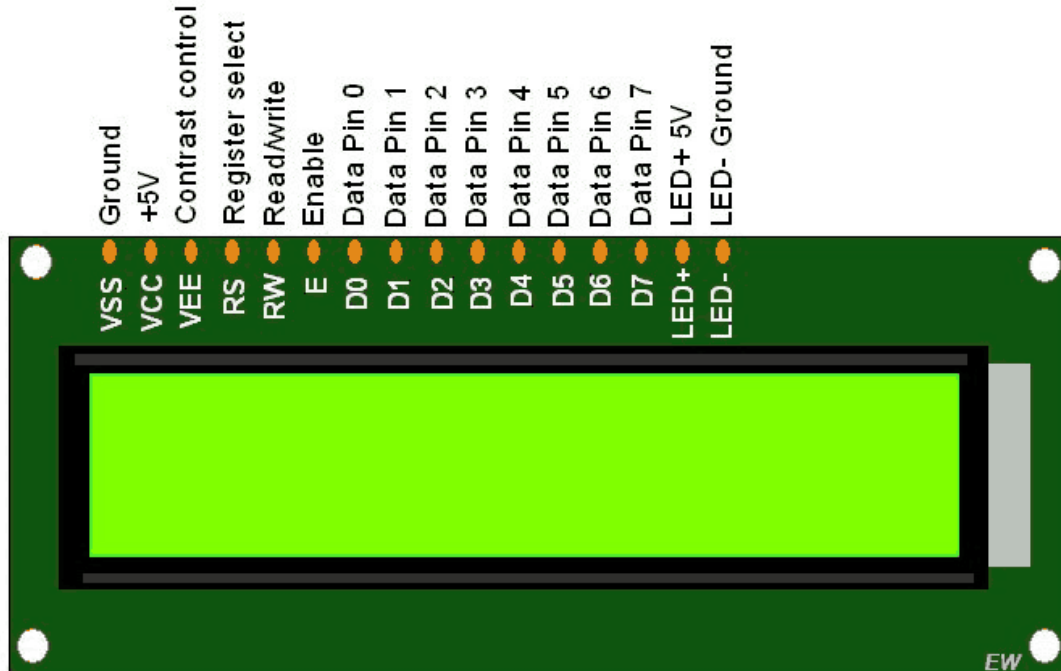


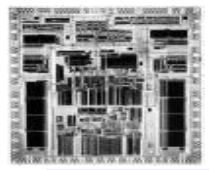
Liquid Crystal Display (LCD) Interfacing



16x2 Liquid Crystal Display which will display the 32 characters at a time in two rows (16 characters in one row).

- **Pin1 (Ground):** This is a GND pin of display.
- **Pin2 (VCC):** This is the voltage supply pin of the display.
- **Pin3 (VEE):** This pin is used to adjust the contrast by connecting to a potentiometer.
- **Pin4 (Register Select):** This pin used to select command or data register. When RS is 0, Command Register is selected and when RS=1, Data Register is selected
- **Pin5 (Read/Write):** This pin is used to select read or write operation (0 = Write Operation, and 1 = Read Operation).
- **Pin 6 (Enable):** LOW to HIGH transition at this pin to perform Read/Write operation
- **Pins 7-14 (Data Pins):** These pins are used to send data/command to the display. In 8-bit LCD mode all the pins D7 to D0 are used. In 4-bit LCD mode only D7-D4 pins are used.
- **Pin15 (+ve pin of the LED):** This pin is connected to +5V
- **Pin 16 (-ve pin of the LED):** This pin is connected to GND.





Liquid Crystal Display (LCD) Interfacing

LCD Command Codes

Instruction	Code										Description	Execution Time (max) (when f_{op} or f_{osc} is 270 kHz)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DDRAM address 0 in address counter.	
Return home	0	0	0	0	0	0	0	0	1	—	Sets DDRAM address 0 in address counter. Also returns display from being shifted to original position. DDRAM contents remain unchanged.	1.52 ms
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies display shift. These operations are performed during data write and read.	37 s
Display on/off control	0	0	0	0	0	0	1	D	C	B	Sets entire display (D) on/off, cursor on/off (C), and blinking of cursor position character (B).	37 s
Cursor or display shift	0	0	0	0	0	1	S/C	R/L	—	—	Moves cursor and shifts display without changing DDRAM contents.	37 s
Function set	0	0	0	0	1	DL	N	F	—	—	Sets interface data length (DL), number of display lines (N), and character font (F).	37 s

I/D=1: Increment Mode; I/D=0: Decrement Mode

S=1: Shift

S/C=1: Display Shift; S/C=0: Cursor Shift

R/L=1: Right Shift; R/L=0: Left Shift

DL=1: 8D DL=0: 4D

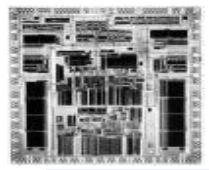
N=1: 2R N=0: 1R

F=1: 5x10 Style; F=0: 5x7 Style

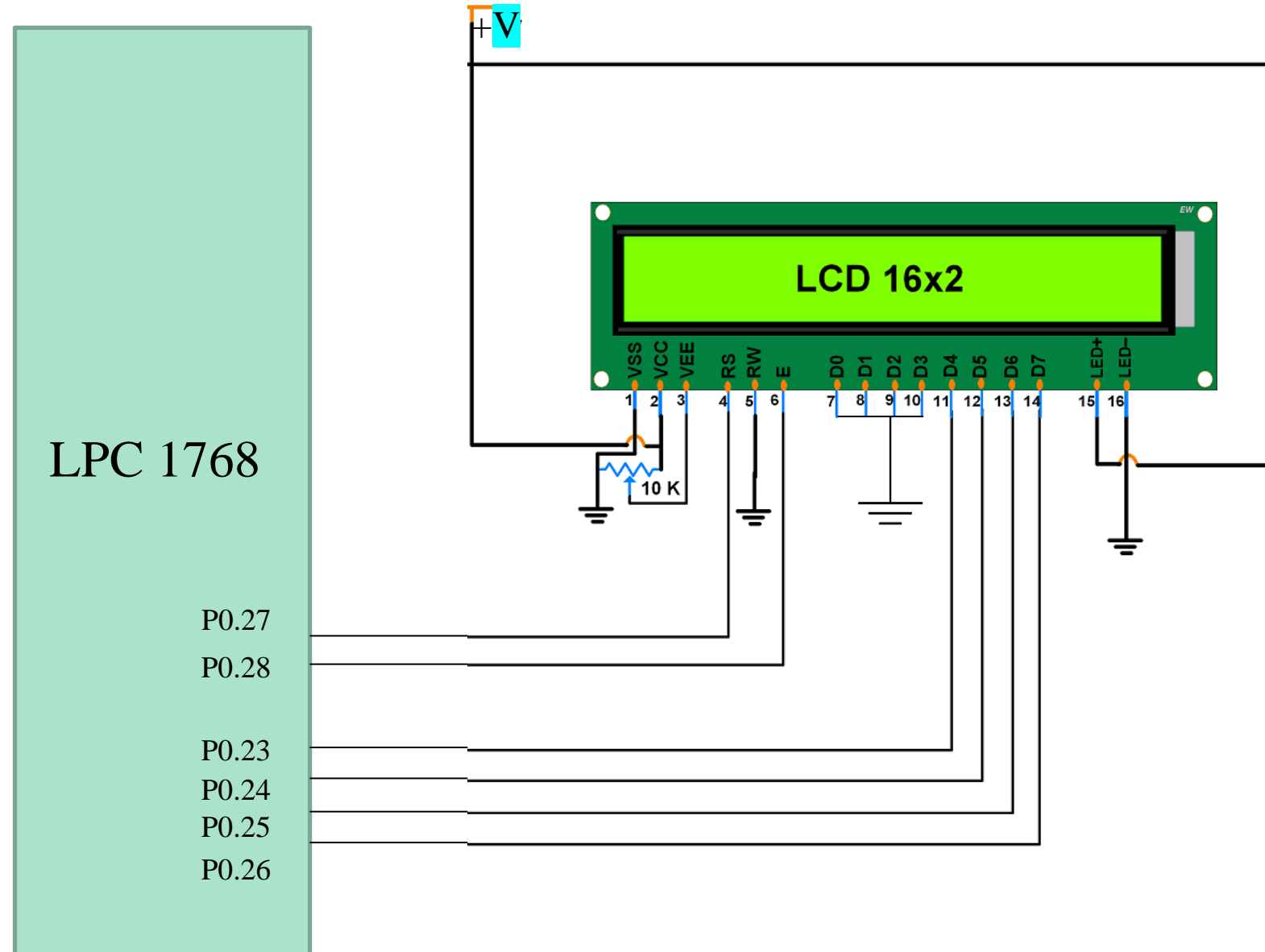
BF=1: Execute Internal Function;

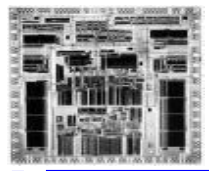
BF=0: Command Received

Code (Hex)	Command to LCD Instruction Register
1	Clear display screen
2	Return home
4	Decrement cursor (shift cursor to left)
6	Increment cursor (shift cursor to right)
5	Shift display right
7	Shift display left
8	Display off, cursor off
A	Display off, cursor on
C	Display on, cursor off
E	Display on, cursor blinking OFF
F	Display on, cursor blinking ON
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to the left
1C	Shift the entire display to the right
80	Force cursor to beginning to 1st line
C0	Force cursor to beginning to 2nd line
38	2 lines and 5x7 matrix



Liquid Crystal Display (LCD) Interfacing






Liquid Crystal Display (LCD) Interfacing

```
#define RS_CTRL 0x08000000 //P0.27
#define EN_CTRL 0x10000000 //P0.28
#define DT_CTRL 0x07800000 //P0.23 to P0.26 data lines

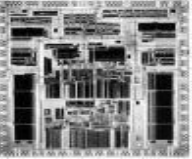
unsigned long int temp1=0, temp2=0,i,j ;
unsigned char flag1 =0, flag2 =0;
unsigned char msg[] = {"WELCOME "};

void lcd_write(void);
void port_write(void);
void delay_lcd(unsigned int);
unsigned long int init_command[] = {0x30,0x30,0x30,0x20,0x28,0x0c,0x06,0x01,0x80};
int main(void)
{
    SystemInit();
    SystemCoreClockUpdate();
    LPC_GPIO0->FIODIR = DT_CTRL | RS_CTRL | EN_CTRL; //Config output
    flag1 =0;//Command
    for (i=0; i<9;i++)
    {
        temp1 = init_command[i];
        lcd_write(); //send Init commands to LCD
    }
    flag1 =1;//Data
    i =0;
    while (msg[i++] != '\0')
    {
        temp1 = msg[i];
        lcd_write();//Send data bytes
    }
    while(1);
}
```

3 times send 8-bit mode command, followed by 4-bit mode



Liquid Crystal Display (LCD) Interfacing



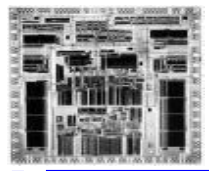
```
void lcd_write(void)
{
    flag2 = (flag1 == 1) ? 0 : ((temp1 == 0x30) || (temp1 == 0x20)) ? 1 : 0; // If command is 0x30 (Working in 8-bit mode initially), send '3' on D7-D4
    // (D3-D0 already grounded)
    temp2 = temp1 & 0xf0; //
    temp2 = temp2 << 19; // data lines from 23 to 26. Shift left (26-8+1) times so that higher digit is sent on P0.26 to P0.23
    port_write(); // Output the higher digit on P0.26-P0.23
    if (!flag2) // Other than command 0x30, send the lower 4-bit also
    {
        temp2 = temp1 & 0x0f; // 26-4+1
        temp2 = temp2 << 23;
        port_write(); // Output the lower digit on P0.26-P0.23
    }
}

void port_write(void)
{
    LPC_GPIO0->FIOPIN = temp2;
    if (flag1 == 0)
        LPC_GPIO0->FIOCLR = RS_CTRL; // Select command register
    else
        LPC_GPIO0->FIOSET = RS_CTRL; // Select data register

    LPC_GPIO0->FIOSET = EN_CTRL; // Apply -ve edge on Enable
    delay_lcd(25);
    LPC_GPIO0->FIOCLR = EN_CTRL;
    delay_lcd(5000);
}

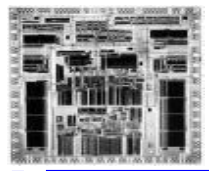
void delay_lcd(unsigned int r1)
{
    unsigned int r;
    for(r=0; r<r1; r++);

    return;
}
```



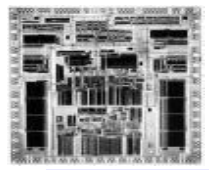
Liquid Crystal Display (LCD) Interfacing

1. Interface a matrix keyboard and display the keycode on the LCD.
2. 4- Digit BCD upcounter on LCD
3. Digital Clock on LCD
4. Simulate a Die Tossing on LCD. Use key connected to P2.12 for Die tossing
5. Input an expression of type **a operator b** from keyboard and display the result on LCD.



Analog To Digital Converter (ADC)

- Analog to Digital Conversion is used when we want to interface an external analog signal or when interfacing analog sensors, like for example a temperature sensor.
- The ADC block in LPC1768 Microcontroller is based on Successive Approximation Register(SAR) conversion method.
- LPC1768 ADC Module uses 12-bit SAR.
- The Measurement range is from VREFN to VREFP, or commonly from 0V to 3.3 V.
- Maximum of 8 multiplexed inputs can be used for ADC.



Analog To Digital Converter (ADC)

Analog voltage and Digital value of the ADC are related as follows:

$$V_A = (V_{REFP} / 2^N) \text{ (Decimal Equivalent of Digital value)}$$

V_A is the Input analog voltage

V_{REFP} Is the Reference voltage of ADC

N is the number of bits

$(V_{REFP} / 2^N)$ is Resolution; It is a constant for given value of N and V_{REFP}

Digital output is directly proportional to Analog input voltage

For 3.3 V, N=12, Resolution is 0.805 mV. i.e 0.805 mV change at the input creates ± 1 change at the digital output.

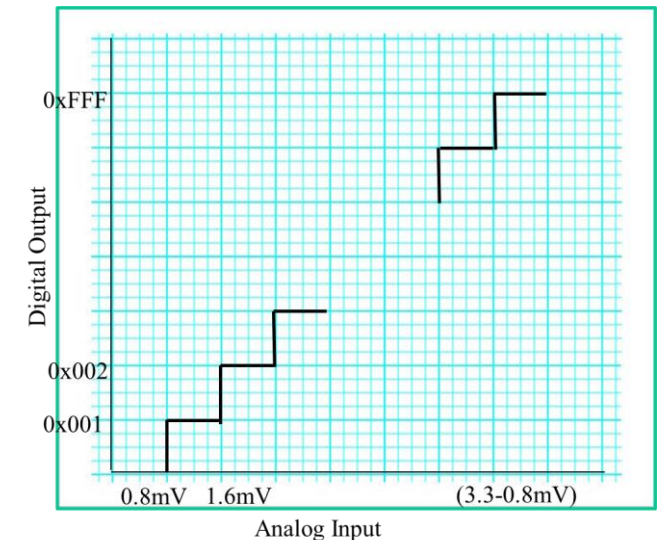
When analog voltage is 0 mV output decimal value is 0

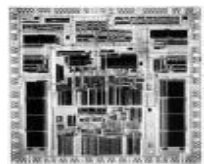
When analog voltage is 0.805 mV output decimal value is 1

When analog voltage is $(0.805 \times 2 = 1.6)$ mV output decimal value is 2

When analog voltage is $(0.805 \times 3 = 2.4)$ mV output decimal value is 3

When analog voltage is $(0.805 \times 4095 = 3.3 - 0.805)$ mV output decimal value is 4095





Analog To Digital Converter (ADC)

Pin	Type	Description
AD0.7 to AD0.0	Input	Analog Inputs. The ADC cell can measure the voltage on any of these input signals. Digital signals are disconnected from the ADC input pins when the ADC function is selected on that pin in the Pin Select register.

ADCR – A/D Control Register

Bit	Symbol	Value	Description
7:0	SEL		Selects which of the AD0.7:0 pins is (are) to be sampled and converted. For AD0, bit 0 selects Pin AD0.0, and bit 7 selects pin AD0.7. In software-controlled mode, only one of these bits should be 1. In hardware scan mode, any value containing 1 to 8 ones is allowed. All zeroes is equivalent to 0x01.
15:8	CLKDIV		The APB clock (PCLK_ADC0) is divided by (this value plus one) to produce the clock for the A/D converter, which should be less than or equal to 13 MHz. Typically, software should program the smallest value in this field that yields a clock of 13 MHz or slightly less, but in certain cases (such as a high-impedance analog source) a slower clock may be desirable.
16	BURST	1	The AD converter does repeated conversions at up to 200 kHz, scanning (if necessary) through the pins selected by bits set to ones in the SEL field. The first conversion after the start corresponds to the least-significant 1 in the SEL field, then higher numbered 1-bits (pins) if applicable. Repeated conversions can be terminated by clearing this bit, but the conversion that's in progress when this bit is cleared will be completed. Remark: START bits must be 000 when BURST = 1 or conversions will not start. If BURST is set to 1, the ADGINTEN bit in the AD0INTEN register (Table 534) must be set to 0.
		0	Conversions are software controlled and require 65 clocks.
20:17	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

AD0.0 –P0.23 FN 01

AD0.1-P0.24 FN 01

AD0.2-P0.25 FN 01

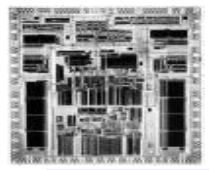
AD0.3-P0.26 FN 01

AD0.4-P0.30 FN 03

AD0.5-P0.31 FN 03

AD0.6-P0.3 FN 02

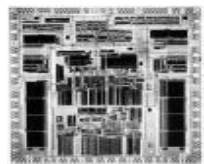
AD0.7-P0.2 FN 02



Analog To Digital Converter (ADC)

ADCR – A/D Control Register

21	PDN	1	The A/D converter is operational.
		0	The A/D converter is in power-down mode.
23:22	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
26:24	START		When the BURST bit is 0, these bits control whether and when an A/D conversion is started:
		000	No start (this value should be used when clearing PDN to 0).
		001	Start conversion now.
		010	Start conversion when the edge selected by bit 27 occurs on the P2.10 / EINT0 / NMI pin.
		011	Start conversion when the edge selected by bit 27 occurs on the P1.27 / CLKOUT / USB_OVRCRn / CAP0.1 pin.
		100	Start conversion when the edge selected by bit 27 occurs on MAT0.1. Note that this does not require that the MAT0.1 function appear on a device pin.
		101	Start conversion when the edge selected by bit 27 occurs on MAT0.3. Note that it is not possible to cause the MAT0.3 function to appear on a device pin.
		110	Start conversion when the edge selected by bit 27 occurs on MAT1.0. Note that this does not require that the MAT1.0 function appear on a device pin.
		111	Start conversion when the edge selected by bit 27 occurs on MAT1.1. Note that this does not require that the MAT1.1 function appear on a device pin.
27	EDGE		This bit is significant only when the START field contains 010-111. In these cases:
		1	Start conversion on a falling edge on the selected CAP/MAT signal.
		0	Start conversion on a rising edge on the selected CAP/MAT signal.
31:28	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

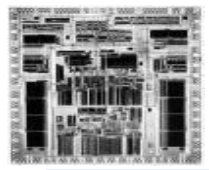


Analog To Digital Converter (ADC)

A/D Global Data Register (ADGDR) :

The A/D Global Data Register holds the result of the most recent A/D conversion that has completed, and also includes copies of the status flags that go with that conversion. Results of ADC conversion can be read in one of two ways. One is to use the A/D Global Data Register to read all data from the ADC. Another is to use the A/D Channel Data Registers.

Bit	Symbol	Description
3:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
15:4	RESULT	When DONE is 1, this field contains a binary fraction representing the voltage on the AD0[n] pin selected by the SEL field, as it falls within the range of V_{REFP} to V_{REFN} . Zero in the field indicates that the voltage on the input pin was less than, equal to, or close to that on V_{REFN} , while 0xFFF indicates that the voltage on the input was close to, equal to, or greater than that on V_{REFP} .
23:16	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
26:24	CHN	These bits contain the channel from which the RESULT bits were converted (e.g. 000 identifies channel 0, 001 channel 1...).
29:27	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
30	OVERRUN	This bit is 1 in burst mode if the results of one or more conversions was (were) lost and overwritten before the conversion that produced the result in the RESULT bits. This bit is cleared by reading this register.
31	DONE	This bit is set to 1 when an A/D conversion completes. It is cleared when this register is read.

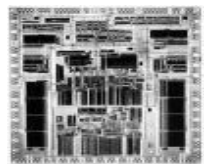


Analog To Digital Converter (ADC)

A/D Data Registers (ADDR0 to ADDR7)

The A/D Data Registers hold the result of the last conversion for each A/D channel, when an A/D conversion is complete. They also include the flags that indicate when a conversion has been completed and when a conversion overrun has occurred.

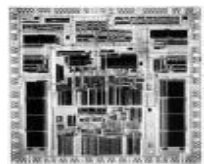
Bit	Symbol	Description
3:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
15:4	RESULT	When DONE is 1, this field contains a binary fraction representing the voltage on the AD0[n] pin, as it falls within the range of V_{REFP} to V_{REFN} . Zero in the field indicates that the voltage on the input pin was less than, equal to, or close to that on V_{REFN} , while 0xFFF indicates that the voltage on the input was close to, equal to, or greater than that on V_{REFP} .
29:16	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
30	OVERRUN	This bit is 1 in burst mode if the results of one or more conversions was (were) lost and overwritten before the conversion that produced the result in the RESULT bits. This bit is cleared by reading this register.
31	DONE	This bit is set to 1 when an A/D conversion completes. It is cleared when this register is read.



Analog To Digital Converter (ADC)

A/D Interrupt Enable register (ADINTEN) : This register allows control over which A/D channels generate an interrupt when a conversion is complete.

Bit	Symbol	Value	Description
0	ADINTEN0	0	Completion of a conversion on ADC channel 0 will not generate an interrupt.
		1	Completion of a conversion on ADC channel 0 will generate an interrupt.
1	ADINTEN1	0	Completion of a conversion on ADC channel 1 will not generate an interrupt.
		1	Completion of a conversion on ADC channel 1 will generate an interrupt.
2	ADINTEN2	0	Completion of a conversion on ADC channel 2 will not generate an interrupt.
		1	Completion of a conversion on ADC channel 2 will generate an interrupt.
3	ADINTEN3	0	Completion of a conversion on ADC channel 3 will not generate an interrupt.
		1	Completion of a conversion on ADC channel 3 will generate an interrupt.
4	ADINTEN4	0	Completion of a conversion on ADC channel 4 will not generate an interrupt.
		1	Completion of a conversion on ADC channel 4 will generate an interrupt.
5	ADINTEN5	0	Completion of a conversion on ADC channel 5 will not generate an interrupt.
		1	Completion of a conversion on ADC channel 5 will generate an interrupt.
6	ADINTEN6	0	Completion of a conversion on ADC channel 6 will not generate an interrupt.
		1	Completion of a conversion on ADC channel 6 will generate an interrupt.
7	ADINTEN7	0	Completion of a conversion on ADC channel 7 will not generate an interrupt.
		1	Completion of a conversion on ADC channel 7 will generate an interrupt.
8	ADGINTEN	0	Only the individual ADC channels enabled by ADINTEN7:0 will generate interrupts. Remark: This bit must be set to 0 in burst mode (BURST = 1 in the AD0CR register).
		1	Only the global DONE flag in ADDR is enabled to generate an interrupt.
31:17	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

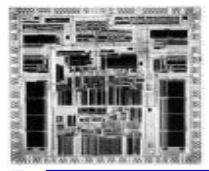


Analog To Digital Converter (ADC)

A/D Status register (ADSTAT) :

The A/D Status register allows checking the status of all A/D channels simultaneously. The DONE and OVERRUN flags appearing in the ADDRn register for each A/D channel are mirrored in ADSTAT. The interrupt flag (the logical OR of all DONE flags) is also found in ADSTAT.

Bit	Symbol	Description
0	DONE0	This bit mirrors the DONE status flag from the result register for A/D channel 0.
1	DONE1	This bit mirrors the DONE status flag from the result register for A/D channel 1.
2	DONE2	This bit mirrors the DONE status flag from the result register for A/D channel 2.
3	DONE3	This bit mirrors the DONE status flag from the result register for A/D channel 3.
4	DONE4	This bit mirrors the DONE status flag from the result register for A/D channel 4.
5	DONE5	This bit mirrors the DONE status flag from the result register for A/D channel 5.
6	DONE6	This bit mirrors the DONE status flag from the result register for A/D channel 6.
7	DONE7	This bit mirrors the DONE status flag from the result register for A/D channel 7.
8	OVERRUN0	This bit mirrors the OVERRRUN status flag from the result register for A/D channel 0.
9	OVERRUN1	This bit mirrors the OVERRRUN status flag from the result register for A/D channel 1.
10	OVERRUN2	This bit mirrors the OVERRRUN status flag from the result register for A/D channel 2.
11	OVERRUN3	This bit mirrors the OVERRRUN status flag from the result register for A/D channel 3.
12	OVERRUN4	This bit mirrors the OVERRRUN status flag from the result register for A/D channel 4.
13	OVERRUN5	This bit mirrors the OVERRRUN status flag from the result register for A/D channel 5.
14	OVERRUN6	This bit mirrors the OVERRRUN status flag from the result register for A/D channel 6.
15	OVERRUN7	This bit mirrors the OVERRRUN status flag from the result register for A/D channel 7.
16	ADINT	This bit is the A/D interrupt flag. It is one when any of the individual A/D channel Done flags is asserted and enabled to contribute to the A/D interrupt via the ADINTEN register.



Analog To Digital Converter (ADC)

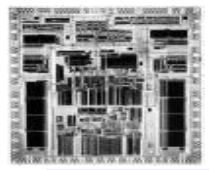
ADC software mode for 2-channel concurrent conversion

```
#include<LPC17xx.h>
#include<stdio.h>
int main(void)
{
    unsigned long temp4, temp5;
    unsigned int i;

    SystemInit();
    SystemCoreClockUpdate();
    LPC_PINCON->PINSEL3 = (3<<28) | (3<<30);          //P1.30 as AD0.4 and P1.31 as AD0.5
    LPC_ADC->ADINTEN = 0;
    while(1)
    {
        LPC_ADC->ADCR = (1<<4)|(1<<21)|(1<<24);        //ADC0.4, start conversion and operational
        while(((temp4=LPC_ADC->ADDR4) & (1<<31)) == 0); //wait till 'done' bit is 1, indicates conversion complete
        temp4 = LPC_ADC->ADDR4;
        temp4 >>= 4;
        temp4 &= 0x00000FFF;                            //12 bit ADC

        LPC_ADC->ADCR = (1<<5)|(1<<21)|(1<<24);        //ADC0.5, start conversion and operational
        for(i=0;i<2000;i++);                             //delay for conversion
        while(((temp5=LPC_ADC->ADDR5) & (1<<31)) == 0); //wait till 'done' bit is 1, indicates conversion complete
        temp5 = LPC_ADC->ADDR5;
        temp5 >>= 4;
        temp5 &= 0x00000FFF;                            //12 bit ADC

        //Now you can use temp4 and temp5 for further processing based on your requirement
    }
}
```



Analog To Digital Converter (ADC)

ADC burst mode for 2-channel concurrent conversion

```
#include<LPC17xx.h>
#include<stdio.h>
int main(void)

{
    SystemInit();
    SystemCoreClockUpdate();
    LPC_PINCON->PINSEL3 =(3<<28)|(3<<30);    //P1.30 as AD0.4 and P1.31 as AD0.5
    LPC_ADC->ADCR = (1<<4) | (1<<5)|(1<<16) | (1<<21); //Enable CH 4 and 5 for BURST mode with ADC power ON
    LPC_ADC->ADINTEN =(1<<4)|(1<<5); // Enable DONE for INTR
    NVIC_EnableIRQ(ADC_IRQn);
    while(1);
}

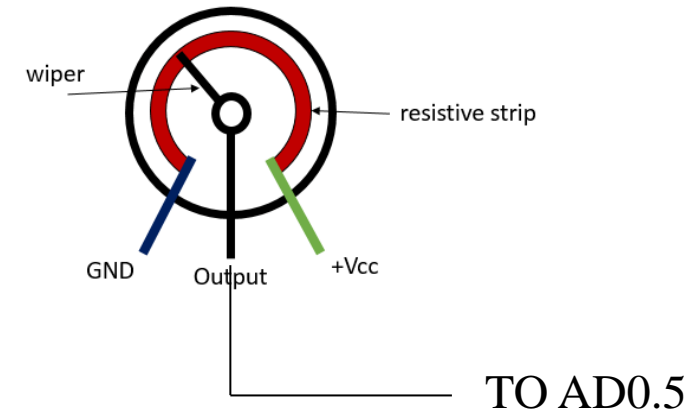
void ADC_IRQHandler(void)
{
    int channel,temp,result;
    channel=(LPC_ADC->ADGDR >>24) & 0x07;
    result= ( LPC_ADC->ADGDR >>4) & 0xFFF;
    if(channel == 4)
    {
        temp4 = ( LPC_ADC->ADDR4 >>4) & 0xFFF ; //Read to Clear Done flag
    }
    else if(channel == 5)
    {
        temp5 = ( LPC_ADC->ADDR5 >>4) & 0xFFF ; //Read to Clear Done flag
    }
    //Now you can use temp4 and temp5 for further processing based on your requirement
}
```

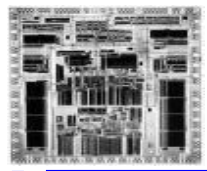
Analog To Digital Converter (ADC)

Input Analog voltage and display its digital equivalent on LCD

```
include<LPC17xx.h>
#include<stdio.h>
#define Ref_Vtg 3.300
#define Full_Scale 0xFFF//12 bit ADC
int main(void)
{
    unsigned long adc_temp;
    unsigned int i;
    float in_vtg;
    unsigned char vtg[7], dval[7];
    unsigned char Msg3[] = {"ANALOG IP:"};
    unsigned char Msg4[] = {"ADC OUTPUT:"};
    SystemInit();
    SystemCoreClockUpdate();
    lcd_init();//Initialize LCD
    LPC_PINCON->PINSEL3 |= 3<<30;//P1.31 as AD0.5
    LPC_SC->PCONP |= (1<<12);//enable the peripheral ADC
    flag1=0;//Command
    temp1 = 0x80;//Cursor at beginning of first line
    lcd_write();
    flag1=1;//Data
    i =0;
    while (Msg3[i++] != '\0')
    {
        temp1 = Msg3[i];
        lcd_write();//Send data bytes
    }
}
```

ANALOG INPUT 1.1V
ADC OUTPUT 555

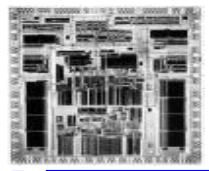




Analog To Digital Converter (ADC)

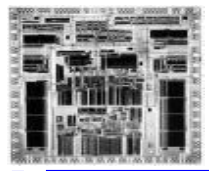
```
flag1=0; //Command
temp1 = 0xC0; //Cursor at beginning of second line
lcd_write();
flag1=1;
i =0;
while (Msg4[i++] != '\0')
{
    temp1 = Msg4[i];
    lcd_write(); //Send data bytes
}

while(1)
{
    LPC_ADC->ADCR = (1<<5)|(1<<21)|(1<<24); //ADC0.5, start conversion and operational
    while(((adc_temp=LPC_ADC->ADGDR) & (1<<31)) == 0);
    adc_temp = LPC_ADC->ADGDR;
    adc_temp >>= 4;
    adc_temp &= 0x00000FFF; //12 bit ADC
    in_vtg = (((float)adc_temp * (float)Ref_Vtg))/((float)Full_Scale); //calculating input analog voltage
    sprintf(vtg,"%3.2fV",in_vtg); //convert the readings into string to display on LCD
    sprintf(dval,"%x",adc_temp);
    flag1=0;;
    temp1 = 0x8A;
    lcd_write();
    flag1=1;
    i =0;
    while (vtg[i++] != '\0')
    {
        temp1 = vtg[i];
        lcd_write(); //Send data bytes
    }
}
```



Analog To Digital Converter (ADC)

```
flag1=0;
temp1 = 0xCB;
lcd_write();
flag1=1;
i =0;
while (dval[i++] != '\0')
{
    temp1 = dval[i];
    lcd_write();//Send data bytes
}
for(i=0;i<7;i++)
    vtg[i] = dval[i] = 0;
}
}
```



Digital to Analog Converter (DAC)

DACs are used in audio equipment like Music Players to convert the digital data into analog audio signals.

Similarly, there are video DACs, for converting digital video data into analog video signals to be displayed on a screen.

LPC1768 contains a 10-bit DAC peripheral based on Resistor String Architecture. It can produce buffered output and the maximum update rate is 1 MHz.

The output analog voltage of this 10-bit DAC can be calculated using the following formula.

$$V_{AOUT} = \frac{DACVALUE * (V_{REFP} - V_{REFN})}{1024} + V_{REFN}$$

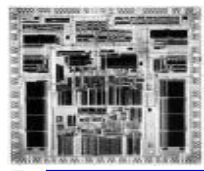
Where,

V_{AOUT} = Output Analog Voltage of DAC [Pin P0.26]

V_{REFP} = Positive Reference Voltage of DAC

V_{REFN} = Negative Reference Voltage of DAC

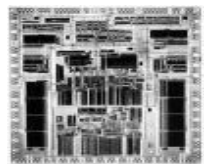
DACVALUE = 10-bit digital value, which must be converted to analog voltage.



Digital to Analog Converter (DAC)

DACR – D/A Converter Register: It contains the digital value that must be converted to analog value.

Bits [15:6]	VALUE	These bits contain the digital value (DACVALUE) that is to be converted to analog value (V_{AOUT}) based on the previously mentioned formula.
Bit [16]	BIAS	When 0, settling time of DAC is 1 μ S (update rate is 1 MHz), max current is 700 μ A. When 1, settling time of DAC is 2.5 μ S (update rate is 400 kHz), max current is 350 μ A.



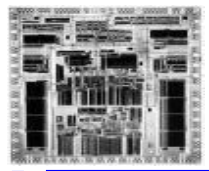
Digital to Analog Converter (DAC)

D/A Converter Control register (DACCTRL)

Bit	Symbol	Value	Description
0	INT_DMA_REQ	0	This bit is cleared on any write to the DACR register.
		1	This bit is set by hardware when the timer times out.
1	DBLBUF_ENA	0	DACR double-buffering is disabled.
		1	When this bit and the CNT_ENA bit are both set, the double-buffering feature in the DACR register will be enabled. Writes to the DACR register are written to a pre-buffer and then transferred to the DACR on the next time-out of the counter.
2	CNT_ENA	0	Time-out counter operation is disabled.
		1	Time-out counter operation is enabled.
3	DMA_ENA	0	DMA access is disabled.
		1	DMA Burst Request Input 7 is enabled for the DAC (see Table 544).

D/A Converter Counter Value register (DACCNTVAL)

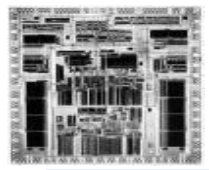
Bit	Symbol	Description
15:0	VALUE	16-bit reload value for the DAC interrupt/DMA timer.



Digital to Analog Converter (DAC)

Double buffering

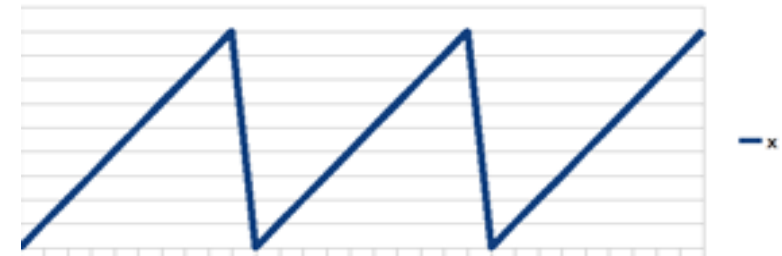
- Double-buffering is enabled only if both, the CNT_ENA and the DBLBUF_ENA bits are set in DACCTRL.
- In this case, any write to the DACR register will only load the pre-buffer
- The DACR will be loaded from the pre-buffer whenever the counter reaches zero
- At the same time the counter is reloaded with the COUNTVAL register value.
- Reading the DACR register will only return the contents of the DACR register itself, not the contents of the pre-buffer register.

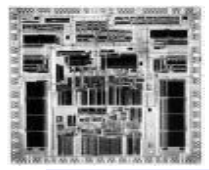


Digital to Analog Converter (DAC)

Generate a sawtooth waveform with peak to peak amplitude 3.3v at P0.26.

```
#include <lpc17xx.h>
void DAC_Init(void);
int main (void)
{
    unsigned int m,i;
    SystemInit();
    SystemCoreClockUpdate();
    LPC_PINCON->PINSEL1 = 2<<20;// Analog Input P0.26
    LPC_DAC->DACCNTVAL = 0x0050; // DAC Counter for Double Buffering
    LPC_DAC->DACCTRL = (0x1<<1)|(0x1<<2); //Double buffering
    while ( 1 )
    {
        LPC_DAC->DACR = (i << 6) ;
        i++;
        if ( i == 0x400 )                //Maximum value is 0x3FF in 10 bit DAC
        {
            i = 0;
        }
    }
}
```

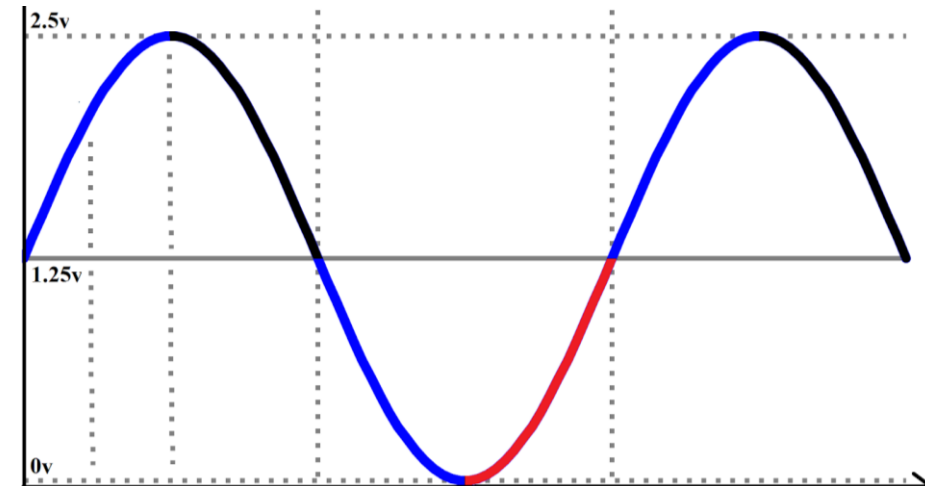


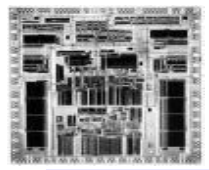


Digital to Analog Converter (DAC)

Generate a sinewave with peak to peak amplitude 2.5v at P0.26. (i.e. $V_{out} = 1.25 + 1.25 \sin \theta$)

```
#include <lpc17xx.h>
void DAC_Init(void);
sinetable[] = { 388, 582, 723, 776, 723, 582, 388, 194, 52, 0, 52, 194 };
int main (void)
{
    unsigned int m,i;
    SystemInit();
    SystemCoreClockUpdate();
    LPC_PINCON->PINSEL1 = 2<<20; // Analog Input P0.26
    while ( 1 )
    {
        for(i=0; i < 12; i++) // Assuming samples separated by 30 degrees
        {
            LPC_DAC->DACR = (sinetable[i % 12]<< 6) ;
            delay(); // Call timer delay based on period. If period is 10 ms. Delay is (10ms/12)
        }
    }
}
```





Digital to Analog Converter (DAC)

Generate a sinewave with peak to peak amplitude 2.5v at P0.26. (i.e. $V_{out} = 1.25 + 1.25 \sin \theta$)

$$V_0 = 1.25 + 1.25 \sin 0 = 1.25; \text{ Dig value} = 1024 \times V / 3.3 = 388$$

$$V_{30} = 1.25 + 1.25 \sin 30 = 1.875; \text{ Dig value} = 582$$

$$V_{60} = 1.25 + 1.25 \sin 60 = 2.33; \text{ Dig value} = 723$$

$$V_{90} = 1.25 + 1.25 \sin 90 = 2.5; \text{ Dig value} = 776$$

$$V_{120} = 1.25 + 1.25 \sin 120 = 2.33; \text{ Dig value} = 723$$

$$V_{150} = 1.25 + 1.25 \sin 150 = 1.875; \text{ Dig value} = 582$$

$$V_{180} = 1.25 + 1.25 \sin 180 = 1.25; \text{ Dig value} = 388$$

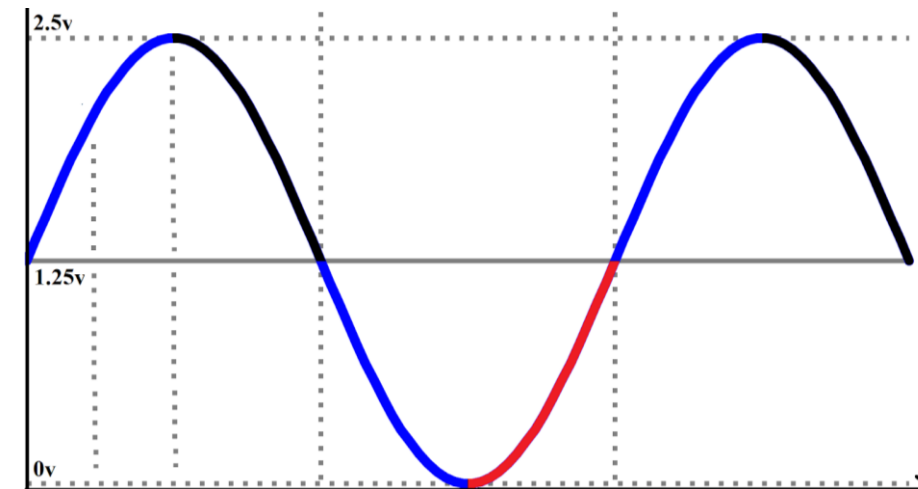
$$V_{210} = 1.25 + 1.25 \sin 210 = 0.626; \text{ Dig value} = 194$$

$$V_{240} = 1.25 + 1.25 \sin 240 = 0.167; \text{ Dig value} = 52$$

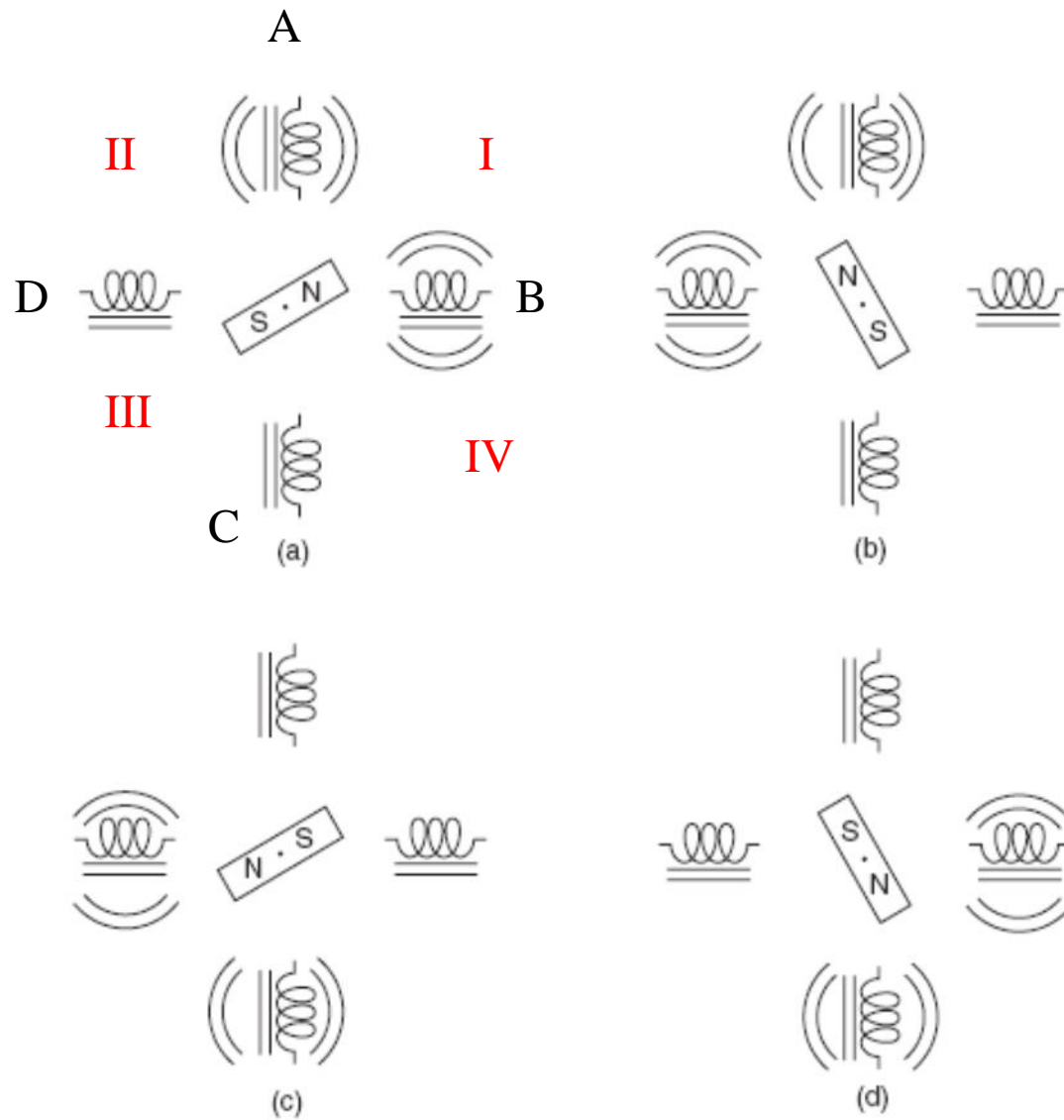
$$V_{270} = 1.25 + 1.25 \sin 270 = 0; \text{ Dig value} = 0$$

$$V_{310} = 1.25 + 1.25 \sin 310 = 0.167; \text{ Dig value} = 52$$

$$V_{330} = 1.25 + 1.25 \sin 330 = 0.626; \text{ Dig value} = 194$$



Stepper Motor Interfacing

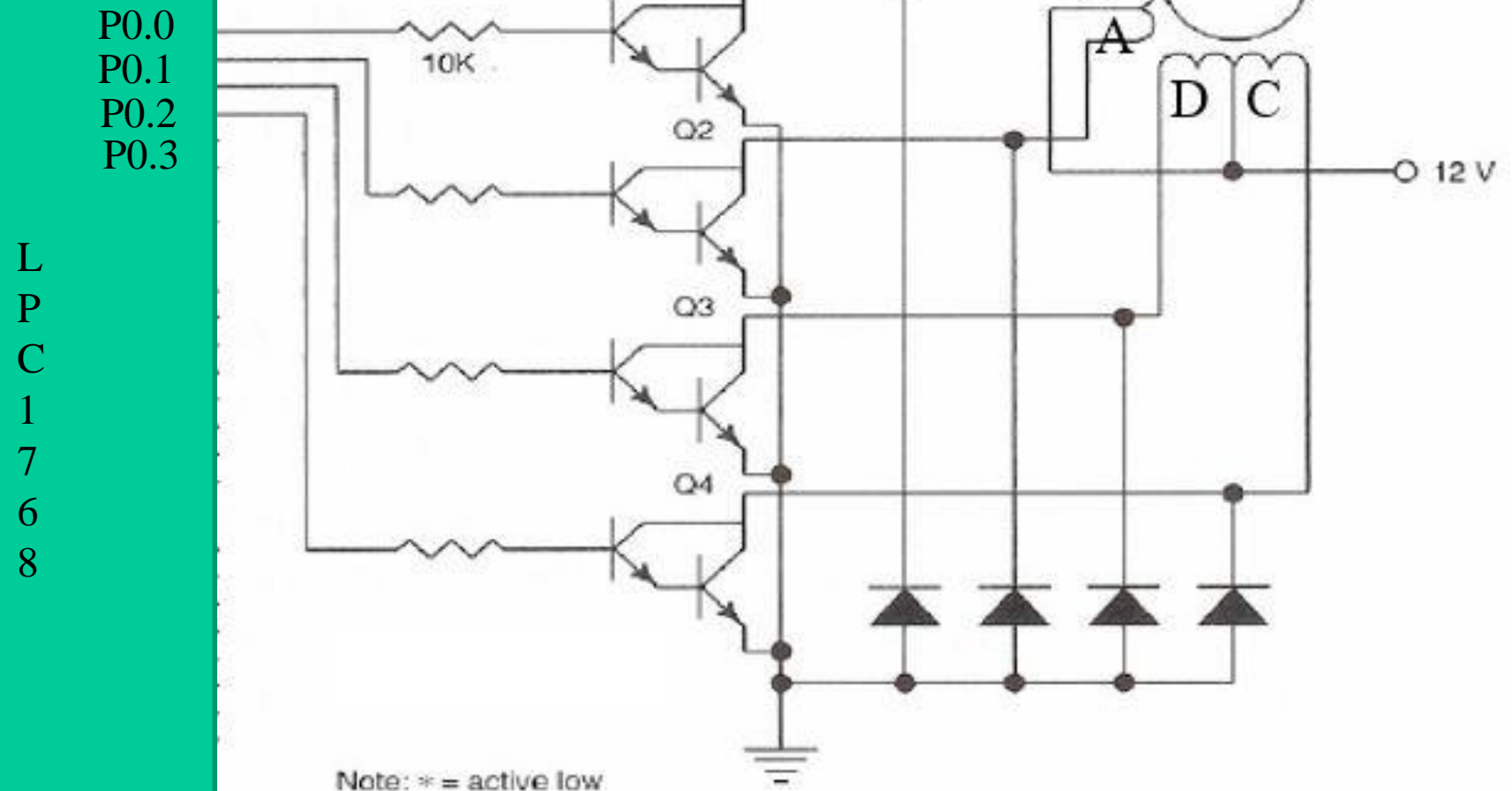


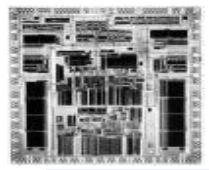
CDAB				
0	0	1	1	3
1	0	0	1	9
1	1	0	0	C
0	1	1	0	6

For Clockwise (Forward) : 3,9,C,6

For Anticlockwise (Reverse): 3,6,C,9

Stepper Motor Interfacing





Stepper Motor Interfacing

Rotate the stepper motor 2.5 revolutions in the clockwise direction at 60 Revolutions Per Minute.

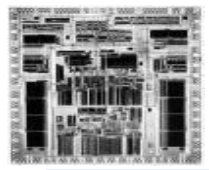
Assume step angle = 6 degrees

```
#include <lpc17xx.h>
step_pos[] = { 3, 9, C, 6 };
int main (void)
{
    unsigned int m, i;
    SystemInit();
    SystemCoreClockUpdate();
    LPC_GPIO0->FIODIR = 0x0F; // Output
    while ( 1 )
    {
        for(i=0; i < 150; i++)
        {
            LPC_GPIO0->FIOPIN = step_pos[i % 4];
            delay(); // Use Timer delay for 16.67 ms
        }
    }
}
```

Time for 1 revolution @ 60 rpm = 1 second

Time for rotating one step = 1 second / 60 steps = 16.67 ms
(i.e. $360/6 = 60$ steps in 1 revolution)

Number of steps needed = $2.5 \times 60 = 150$



Pulse Width Modulation (PWM)

PWM is one of the commonly used techniques to control the amount of power delivered through a pin. The PWM technique allows you to control the brightness of an LED, speed of a Motor, position of a Servo etc.

The width of the Pulse i.e. the duration for which the pulse stays ON in a period is varied. Hence, it is called Pulse Width Modulation.

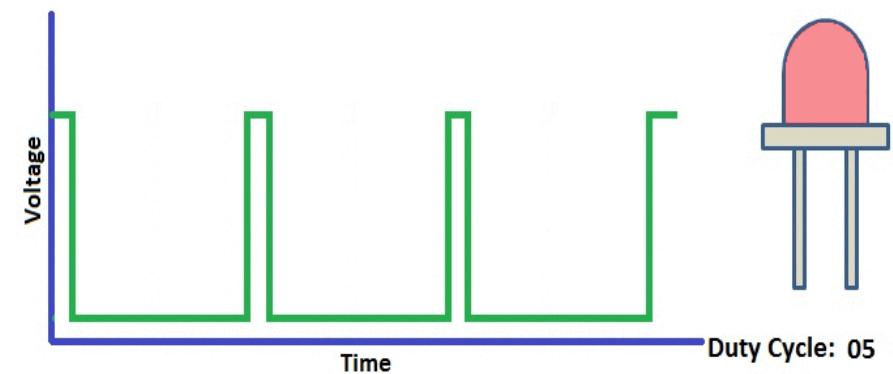
The Period of a PWM Cycle is the sum of duration for which the Pulse is HIGH and the duration for which the Pulse is LOW. This is usually represented by T_{ON} and T_{OFF} . So, Period of PWM = $T_{ON} + T_{OFF}$.

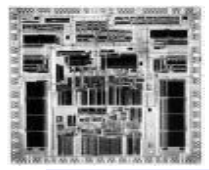
An important parameter of a PWM Signal is its Duty cycle. Duty cycle is the ratio of duration for which Pulse is HIGH to the total period of the PWM Signal.

$$\text{Duty cycle} = T_{ON} / (T_{ON} + T_{OFF})$$

Duty cycle can also be represented as percentage.

$$\text{Duty cycle \%} = T_{ON} * 100 / (T_{ON} + T_{OFF})$$





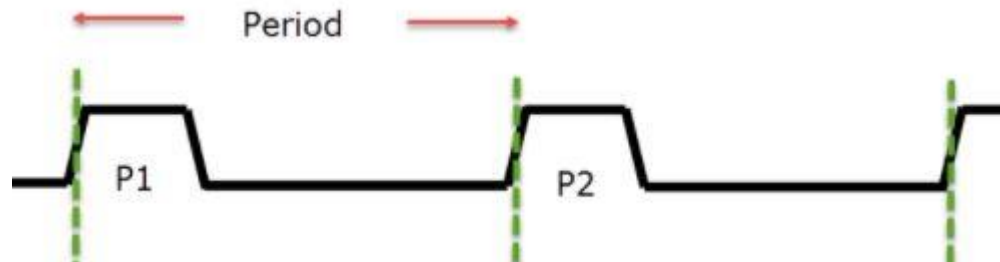
Pulse Width Modulation (PWM)

Two types of PWM

:

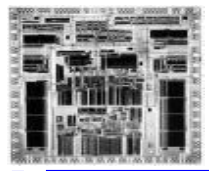
- Single Edge PWM
- Double Edge PWM

In **Single Edge PWM**, the Pulse starts at the beginning of the period



In **Double Edge PWM**, both the edges can be modulated and hence, the pulse is placed anywhere in the period. Double Edge PWM is generally used in multi-phase motor control applications.





Pulse Width Modulation (PWM)

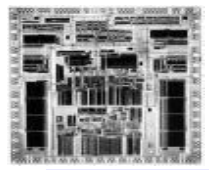
Duty cycle of the PWM determines the average power of a PWM Signal and it is calculated using the following formula.

$$V_{AVG} = \text{Duty Cycle} * V_H$$

Where V_H is the maximum voltage level of the PWM Signal.

In LPC1768, there are 6 PWM outputs called PWM1.1, PWM1.2, ...PWM1.6

PWM1.1	P1.18 / P2.0
PWM1.2	P1.20 / P2.1 / P3.25
PWM1.3	P1.21 / P2.2 / P3.26
PWM1.4	P1.23 / P2.3
PWM1.5	P1.24 / P2.4
PWM1.6	P1.26 / P2.5



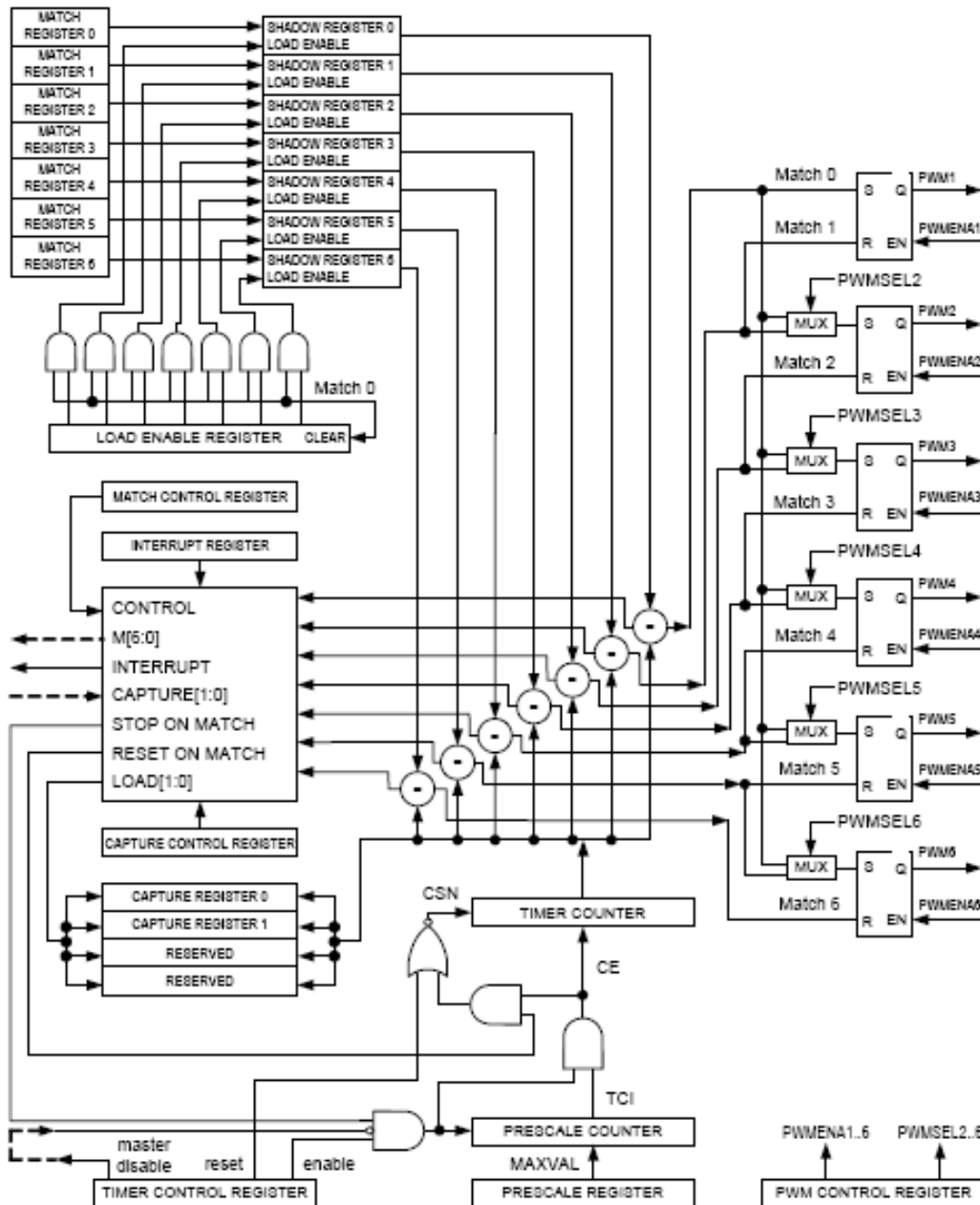
Pulse Width Modulation (PWM)

The PWM is based on the standard Timer block and inherits all of its features, although only the PWM function is pinned out.

The Timer is designed to count cycles of the peripheral clock (PCLK) and optionally generate interrupts or perform other actions when specified timer values occur, based on seven match registers.

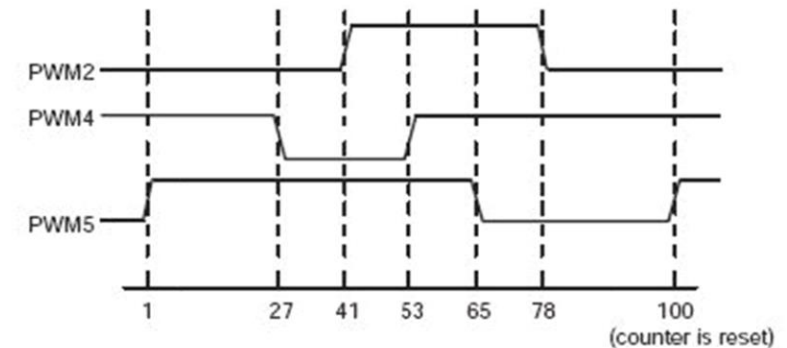
The PWM function is in addition to these features, and is based on match register events.

Pulse Width Modulation (PWM)



Two match registers can be used to provide a single edge controlled PWM output. One match register (PWMMR0) controls the PWM cycle rate, by resetting the count upon match. The other match register controls the PWM edge position.

Three match registers can be used to provide a PWM output with both edges controlled. Again, the PWMMR0 match register controls the PWM cycle rate. The other match registers control the two PWM edge positions.



The waveforms show one PWM cycle and demonstrate PWM outputs under the following conditions:

The timer is configured for PWM mode (counter resets to 1).

Match 0 is configured to reset the timer/counter when a match event occurs.

Control bits PWMSEL2 and PWMSEL4 are set.

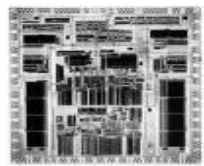
The Match register values are as follows:

MR0 = 100 (PWM rate)

MR1 = 41, MR2 = 78 (PWM2 output)

MR3 = 53, MR4 = 27 (PWM4 output)

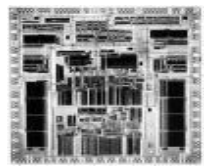
MR5 = 65 (PWM5 output)



Pulse Width Modulation (PWM)

Set and reset inputs for PWM Flip-Flops

PWM Channel	Single Edge PWM (PWMSELn = 0)		Double Edge PWM (PWMSELn = 1)	
	Set by	Reset by	Set by	Reset by
1	Match 0	Match 1	Match 0 ^[1]	Match 1 ^[1]
2	Match 0	Match 2	Match 1	Match 2
3	Match 0	Match 3	Match 2 ^[2]	Match 3 ^[2]
4	Match 0	Match 4	Match 3	Match 4
5	Match 0	Match 5	Match 4 ^[2]	Match 5 ^[2]
6	Match 0	Match 6	Match 5	Match 6

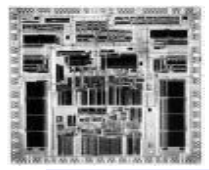


Pulse Width Modulation (PWM)

PWM Timer Control Register (PWM1TCR)

The PWM Timer Control Register (PWMTCR) is used to control the operation of the PWM Timer Counter.

Bit 0	Counter Enable	When 1, PWM Timer Counter and Prescale Counter are enabled.
Bit 1	Counter Reset	When 1, PWM Timer Counter and Prescale Counter are reset on next positive edge of PCLK.
Bit 3	PWM Enable	When 1, PWM Mode is enabled. TC will reset to 1.

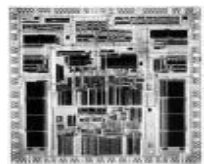


Pulse Width Modulation (PWM)

PWM Count Control Register (PWM1CTCR)

The Count Control Register (CTCR) is used to select between Timer and Counter mode

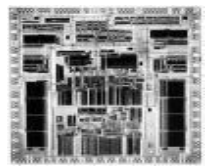
Bit	Symbol	Value	Description
1:0	Counter/ Timer Mode	00	Timer Mode: the TC is incremented when the Prescale Counter matches the Prescale Register.
		01	Counter Mode: the TC is incremented on rising edges of the PCAP input selected by bits 3:2.
		10	Counter Mode: the TC is incremented on falling edges of the PCAP input selected by bits 3:2.
		11	Counter Mode: the TC is incremented on both edges of the PCAP input selected by bits 3:2.
3:2	Count Input Select		When bits 1:0 of this register are not 00, these bits select which PCAP pin which carries the signal used to increment the TC.
		00	PCAP1.0
		01	PCAP1.1 (Other combinations are reserved)



Pulse Width Modulation (PWM)

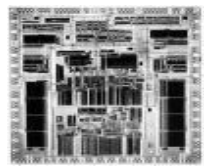
PWM Match Control Register (PWM1MCR)

Bit	Symbol	Value	Description
0	PWMMR0I	1	Interrupt on PWMMR0: an interrupt is generated when PWMMR0 matches the value in the PWMTC.
		0	This interrupt is disabled.
1	PWMMR0R	1	Reset on PWMMR0: the PWMTC will be reset if PWMMR0 matches it.
		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	This feature is disabled
3	PWMMR1I	1	Interrupt on PWMMR1: an interrupt is generated when PWMMR1 matches the value in the PWMTC.
		0	This interrupt is disabled.
4	PWMMR1R	1	Reset on PWMMR1: the PWMTC will be reset if PWMMR1 matches it.
		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	This feature is disabled.
6	PWMMR2I	1	Interrupt on PWMMR2: an interrupt is generated when PWMMR2 matches the value in the PWMTC.
		0	This interrupt is disabled.
7	PWMMR2R	1	Reset on PWMMR2: the PWMTC will be reset if PWMMR2 matches it.
		0	This feature is disabled.



Pulse Width Modulation (PWM)

Bit	Symbol	Value	Description
8	PWMMR2S	1	Stop on PWMMR2: the PWMTc and PWMPC will be stopped and PWMTcR[0] will be set to 0 if PWMMR2 matches the PWMTc.
		0	This feature is disabled
9	PWMMR3I	1	Interrupt on PWMMR3: an interrupt is generated when PWMMR3 matches the value in the PWMTc.
		0	This interrupt is disabled.
10	PWMMR3R	1	Reset on PWMMR3: the PWMTc will be reset if PWMMR3 matches it.
		0	This feature is disabled
11	PWMMR3S	1	Stop on PWMMR3: The PWMTc and PWMPC will be stopped and PWMTcR[0] will be set to 0 if PWMMR3 matches the PWMTc.
		0	This feature is disabled
12	PWMMR4I	1	Interrupt on PWMMR4: An interrupt is generated when PWMMR4 matches the value in the PWMTc.
		0	This interrupt is disabled.
13	PWMMR4R	1	Reset on PWMMR4: the PWMTc will be reset if PWMMR4 matches it.
		0	This feature is disabled.
14	PWMMR4S	1	Stop on PWMMR4: the PWMTc and PWMPC will be stopped and PWMTcR[0] will be set to 0 if PWMMR4 matches the PWMTc.
		0	This feature is disabled
15	PWMMR5I	1	Interrupt on PWMMR5: An interrupt is generated when PWMMR5 matches the value in the PWMTc.
		0	This interrupt is disabled.
16	PWMMR5R	1	Reset on PWMMR5: the PWMTc will be reset if PWMMR5 matches it.
		0	This feature is disabled.
17	PWMMR5S	1	Stop on PWMMR5: the PWMTc and PWMPC will be stopped and PWMTcR[0] will be set to 0 if PWMMR5 matches the PWMTc.
		0	This feature is disabled
18	PWMMR6I	1	Interrupt on PWMMR6: an interrupt is generated when PWMMR6 matches the value in the PWMTc.
		0	This interrupt is disabled.
19	PWMMR6R	1	Reset on PWMMR6: the PWMTc will be reset if PWMMR6 matches it.
		0	This feature is disabled.
20	PWMMR6S	1	Stop on PWMMR6: the PWMTc and PWMPC will be stopped and PWMTcR[0] will be set to 0 if PWMMR6 matches the PWMTc.

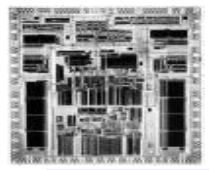


Pulse Width Modulation (PWM)

PWM Capture Control Register (PWM1CCR)

The Capture Control Register is used to control whether one of the two Capture Registers is loaded with the value in the Timer Counter when a capture event occurs, and whether an interrupt is generated by the capture event. Setting both the rising and falling bits at the same time is a valid configuration, resulting in a capture event for both edges.

Bit	Symbol	Value	Description
0	Capture on PCAP1.0 rising edge	0	This feature is disabled.
		1	A synchronously sampled rising edge on the PCAP1.0 input will cause CR0 to be loaded with the contents of the TC.
1	Capture on PCAP1.0 falling edge	0	This feature is disabled.
		1	A synchronously sampled falling edge on PCAP1.0 will cause CR0 to be loaded with the contents of TC.
2	Interrupt on PCAP1.0 event	0	This feature is disabled.
		1	A CR0 load due to a PCAP1.0 event will generate an interrupt.
3	Capture on PCAP1.1 rising edge	0	This feature is disabled.
		1	A synchronously sampled rising edge on the PCAP1.1 input will cause CR1 to be loaded with the contents of the TC.
4	Capture on PCAP1.1 falling edge	0	This feature is disabled.
		1	A synchronously sampled falling edge on PCAP1.1 will cause CR1 to be loaded with the contents of TC.
5	Interrupt on PCAP1.1 event	0	This feature is disabled.
		1	A CR1 load due to a PCAP1.1 event will generate an interrupt



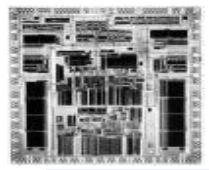
Pulse Width Modulation (PWM)

PWM Control Register (PWM1PCR)

The PWM Control Register is used to enable and select the type of each PWM channel.

Bit	Symbol	Value	Description
1:0	Unused		Unused, always zero.
2	PWMSEL2	1	Selects double edge controlled mode for the PWM2 output.
		0	Selects single edge controlled mode for PWM2.
3	PWMSEL3	1	Selects double edge controlled mode for the PWM3 output.
		0	Selects single edge controlled mode for PWM3.
4	PWMSEL4	1	Selects double edge controlled mode for the PWM4 output.
		0	Selects single edge controlled mode for PWM4.
5	PWMSEL5	1	Selects double edge controlled mode for the PWM5 output.
		0	Selects single edge controlled mode for PWM5.
6	PWMSEL6	1	Selects double edge controlled mode for the PWM6 output.
		0	Selects single edge controlled mode for PWM6.
8:7	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

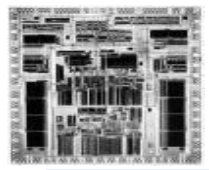
Bit	Symbol	Value	Description
9	PWМЕНА1	1	The PWM1 output enabled.
		0	The PWM1 output disabled.
10	PWМЕНА2	1	The PWM2 output enabled.
		0	The PWM2 output disabled.
11	PWМЕНА3	1	The PWM3 output enabled.
		0	The PWM3 output disabled.
12	PWМЕНА4	1	The PWM4 output enabled.
		0	The PWM4 output disabled.
13	PWМЕНА5	1	The PWM5 output enabled.
		0	The PWM5 output disabled.
14	PWМЕНА6	1	The PWM6 output enabled.
		0	The PWM6 output disabled.
31:15	Unused		Unused, always zero.



Pulse Width Modulation (PWM)

PWM Latch Enable Register (PWM1LER)

Bit	Symbol	Description
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. See Section 24.6.4 "PWM Match Control Register (PWM1MCR - 0x4001 8014)" .
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. See Section 24.6.4 "PWM Match Control Register (PWM1MCR - 0x4001 8014)" .
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. See Section 24.6.4 "PWM Match Control Register (PWM1MCR - 0x4001 8014)" .
3	Enable PWM Match 3 Latch	Writing a one to this bit allows the last value written to the PWM Match 3 register to be become effective when the timer is next reset by a PWM Match event. See Section 24.6.4 "PWM Match Control Register (PWM1MCR - 0x4001 8014)" .
4	Enable PWM Match 4 Latch	Writing a one to this bit allows the last value written to the PWM Match 4 register to be become effective when the timer is next reset by a PWM Match event. See Section 24.6.4 "PWM Match Control Register (PWM1MCR - 0x4001 8014)" .
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. See Section 24.6.4 "PWM Match Control Register (PWM1MCR - 0x4001 8014)" .
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. See Section 24.6.4 "PWM Match Control Register (PWM1MCR - 0x4001 8014)" .



Pulse Width Modulation (PWM)

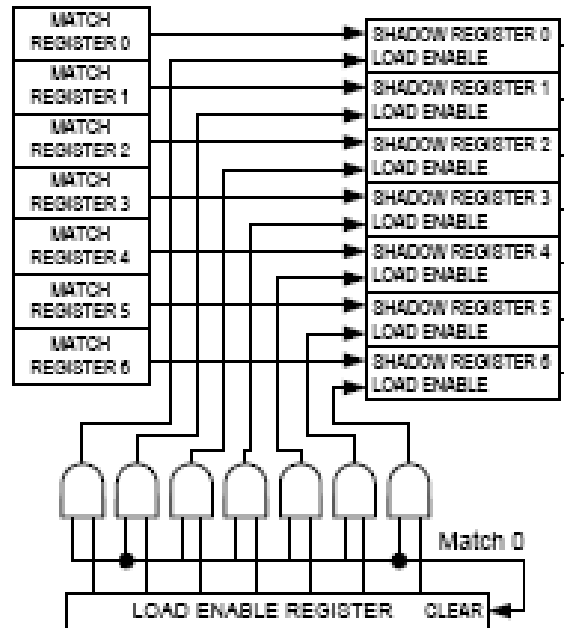
PWM Latch Enable Register (PWM1LER)

The PWM Latch Enable Registers are used to control the update of the PWM Match registers when they are used for PWM generation.

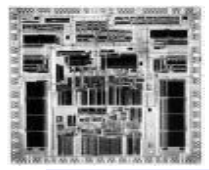
When a PWM Match 0 event occurs (normally also resetting the timer in PWM mode), the contents of shadow registers will be transferred to the shadow registers if the corresponding bit in the Latch Enable Register has been set.

At that point, the new values will take effect and determine the course of the next PWM cycle. Once the transfer of new values has taken place, all bits of the LER are automatically cleared.

Until the corresponding bit in the PWMLER is set and a PWM Match 0 event occurs, any value written to the PWM Match registers has no effect on PWM operation.

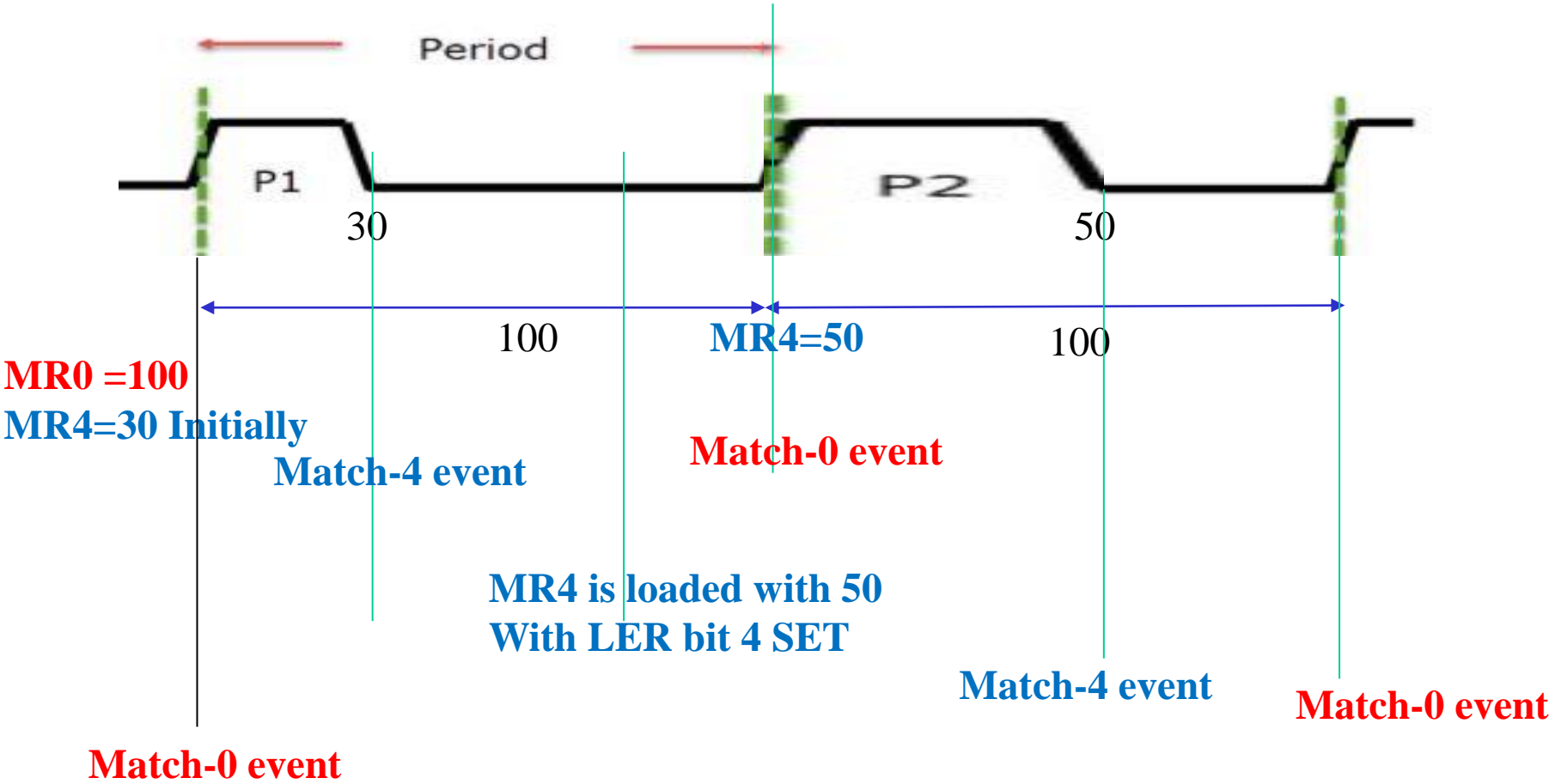


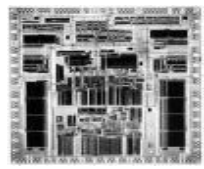
To be compared with TC to generate Match0-Match6 outputs



Pulse Width Modulation (PWM)

Single edge PMM1.4



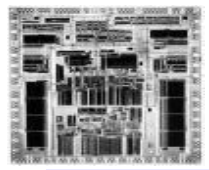


Pulse Width Modulation (PWM)

PWM Interrupt Register (PWM1IR)

The PWM Interrupt Register consists of 9 INTR Flags (7 Match, 2 Capture) If an interrupt is generated then the corresponding bit in the PWMIR will be high. Otherwise, the bit will be low. Writing a logic 1 to the corresponding IR bit will reset the interrupt.

Bit	Symbol	Description
0	PWMMR0 Interrupt	Interrupt flag for PWM match channel 0.
1	PWMMR1 Interrupt	Interrupt flag for PWM match channel 1.
2	PWMMR2 Interrupt	Interrupt flag for PWM match channel 2.
3	PWMMR3 Interrupt	Interrupt flag for PWM match channel 3.
4	PWMCAP0 Interrupt	Interrupt flag for capture input 0
5	PWMCAP1 Interrupt	Interrupt flag for capture input 1.
7:6	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.
8	PWMMR4 Interrupt	Interrupt flag for PWM match channel 4.
9	PWMMR5 Interrupt	Interrupt flag for PWM match channel 5.
10	PWMMR6 Interrupt	Interrupt flag for PWM match channel 6.
31:11	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.

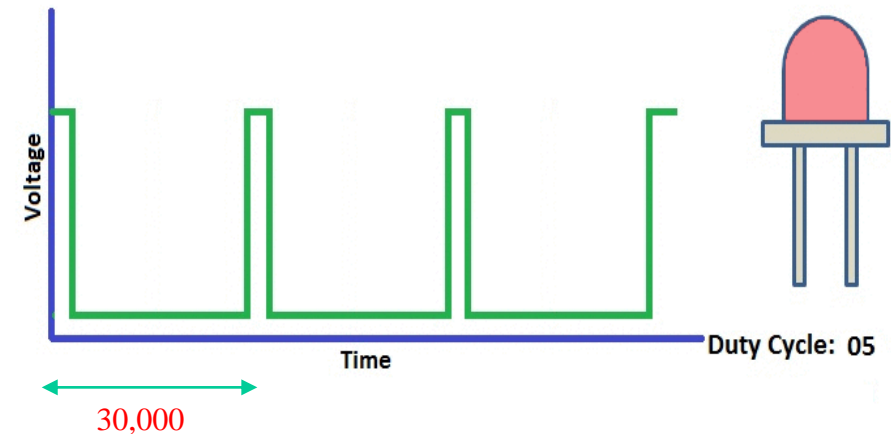


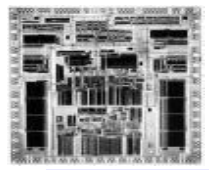
Pulse Width Modulation (PWM)

PWM program to change the intensity of LED connected to (PWM1.4) P1.23 continuously

```
#include <LPC17xx.H>
void pwm_init(void);
void PWM1_IRQHandler(void);
unsigned int a;
unsigned long int i;
unsigned char flag, flag1;
int main(void)
{
    SystemInit();
    SystemCoreClockUpdate();
    pwm_init();
    while(1);
}

void pwm_init(void)
{
    LPC_PINCON->PINSEL3 = 2<<14;           //pwm1.4 is selected
    LPC_PWM1->PR = 0x00000000; //Count frequency : Fpclk
    LPC_PWM1->PCR = 0x00001000; //select PWM1 single edge
    LPC_PWM1->MCR = 0x00000003; //Reset and interrupt on PWMMR0
    LPC_PWM1->MR0 = 30000; //setup match register 0 count
    LPC_PWM1->MR4 = 50; //setup match register MR4
    LPC_PWM1->LER = 0x000000FF; //enable shadow copy register
    LPC_PWM1->TCR = 0x00000002; //RESET TC and PC
    LPC_PWM1->TCR = 0x00000009; //enable PWM and counter
    NVIC_EnableIRQ(PWM1_IRQn);
    return;
}
```





Pulse Width Modulation (PWM)

```
void PWM1_IRQHandler(void)
{
    LPC_PWM1->IR = 0x01; //clear the interrupts
    if(flag == 0x00)
    {
        LPC_PWM1->MR4 += 50;
        LPC_PWM1->LER = 0x000000FF;
        if(LPC_PWM1->MR4 > 29900)
        {
            flag = 0xff;
            LPC_PWM1->LER = 0x000000FF;
        }
    }
    else if(flag == 0xff)
    {
        LPC_PWM1->MR4 -= 50;
        LPC_PWM1->LER = 0x000000ff;
        if(LPC_PWM1->MR4 < 100)
        {
            flag = 0x00;
            LPC_PWM1->LER = 0x000000FF;
        }
    }
}
```