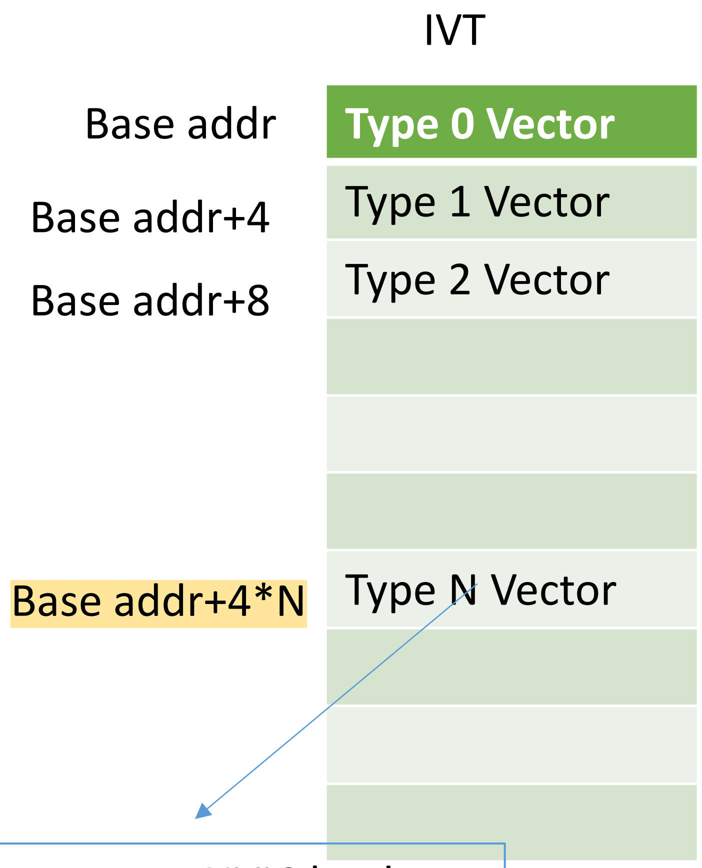
Interrupts, LCD and ADC

Nested Vectored Interrupt Controller(NVIC)

- Controls system exceptions and peripheral interrupts
- In the LPC176x, the NVIC supports 35 vectored interrupts
- Each peripheral device may have one or more interrupt lines to the Vectored Interrupt Controller.
- Interrupt numbers relate to where entries are stored in the Interrupt vector table.
- Interrupt Vector – Is the address of Interrupt Service Subroutine (ISR)
- Interrupt vector of Interrupt Type N is stored at an offset N*4 from the base address of Interrupt Vector Table(IVT)
- If the peripheral device is enabled to generate Interrupt when some event



When Interrupt occurs, NVIC loads vector to PC and executes the ISR to provide the service to peripheral

Nested Vectored Interrupt Controller(NVIC)

- If the peripheral device is enabled to generate Interrupt when some event occurs, the INTR request is sent to NVIC
- If NVIC is enabled to service the INTR request from the peripheral, it services the INTR request by executing ISR pertaining to the peripheral.(i.e Save the return address, Get the Interrupt Vector from IVT and load that address to PC. Upon completion of ISR execution resumes the calling function)

Nested Vectored Interrupt Controller(NVIC)

- In case of Timer, there are 6 INTR enable flags (4 in MCR and 2 in CCR. i.e 4 Match events and 2 Capture events can generate the Interrupt when the event occurs)
- When the event occurs the corresponding bit is set automatically in the IR register. This indicates the NVIC about the event
- If the NVIC is enabled to service the Timer Interrupt, it executes the corresponding ISR and gives the desired service to the Timer.
- In the ISR, clear the corresponding bit, by writing back 1.

Timer/Counter Interrupt Programming

Interrupt Register (IR)

The Interrupt Register consists of 4 bits for the match interrupts and 2 bits for the capture interrupts. If an interrupt is generated then the corresponding bit in the IR will be high. Otherwise, the bit will be low.

Writing a logic one to the corresponding IR bit will reset the interrupt. Writing a zero has no effect.

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.

Timer/Counter Interrupt Programming

```
#include<stdio.h>
#include<LPC17xx.h>
unsigned int ticks=0,x;
void TIMER0_IRQHandler(void)
{
LPC_TIM0->IR = 1;
        ticks++;
        if(ticks==1000)
        {
                ticks=0;
                LPC_GPIO0->FIOPIN=~(LPC_GPIO0->FIOPIN & 0x00000004);
        }
                }
void init_timer0(void)
{
        LPC_TIM0->TCR = 0x00000002;     // Timer0 Reset
        LPC_TIM0->CTCR =0x00;//Timer
        LPC_TIM0->MR0 = 2999; //  For 1ms
        LPC_TIM0->EMR = 0X00;//Do nothing for EM0
        LPC_TIM0->PR = 0;
        LPC_TIM0->MCR = 0x00000003;   //Reset TC upon Match-0 and generate INTR
        LPC_TIM0->TCR = 0x00000001;     // Timer0 Enable

        return;
```

Toggle LED connected to p0.2 every second while displaying the status of switch connected to P1.0 on the LED connected to P2.0

Timer/Counter Interrupt Programming

```
int main(void)
{
        LPC_GPIO0->FIODIR=0x00000004;
        LPC_GPIO2->FIODIR=0x00000001;
        init_timer0();
        NVIC_EnableIRQ(TIMER0_IRQn);//timer 0 intr enabled in NVIC
                while(1)
                {
                        LPC_GPIO2->FIOPIN=(LPC_GPIO1->FIOPIN & 0x01) ;
                }

}
```

Timer/Counter Interrupt Programming

Toggle P0.2 whenever counter value reaches 3.   I. e for every 4 edges using counter interrupt.

```
#include<stdio.h>
#include<LPC17xx.h>
void TIMER0_IRQHandler(void)
{
                LPC_TIM0->IR = 1; //Clear the interrupt
                LPC_GPIO0->FIOPIN=~(LPC_GPIO0->FIOPIN & 0x00000004);

}
void init_timer0(void)
{        LPC_TIM0->TCR = 0x00000002;    // Timer0 Reset
         LPC_TIM0->CTCR =0x05; // Counter at +ve edge of CAP0.1
         LPC_TIM0->MR0 = 3;
         LPC_TIM0->EMR = 0X00;
         LPC_TIM0->PR = 0;
         LPC_TIM0->MCR = 0x00000003;
         LPC_TIM0->TCR = 0x00000001;    // Timer0 Enable
         return;
         }

int main(void)
{
         LPC_GPIO0->FIODIR=0x00000004;
         LPC_PINCON->PINSEL3 |=((3<<22)|(3<<24));
         init_timer0();
         NVIC_EnableIRQ(TIMER0_IRQn);
         while(1);
}
```

Timer/Counter Interrupt Programming

Timer interrupt for rectangular waveform generation (1.5 second HIGH and 0.5 second LOW)

```
include<stdio.h>
#include<LPC17xx.h>
unsigned char flag=1;
void TIMER0_IRQHandler(void)
{
LPC_TIM0->IR = 1;

if(flag)
        {
                flag=0;
                LPC_TIM0->TCR = 0x00000002;    // Timer0 Reset
                LPC_GPIO0->FIOCLR=0x00000004;
                LPC_TIM0->MR0 = 500;
                LPC_TIM0->TCR = 0x00000001;    // Timer0 Enable
        }
        else
                {
                flag=1;
                LPC_TIM0->TCR = 0x00000002;    // Timer0 Reset
                LPC_GPIO0->FIOSET=0x00000004;
                LPC_TIM0->MR0 = 1500;
                LPC_TIM0->TCR = 0x00000001;    // Timer0 Enable
                }

}
```

Timer/Counter Interrupt Programming

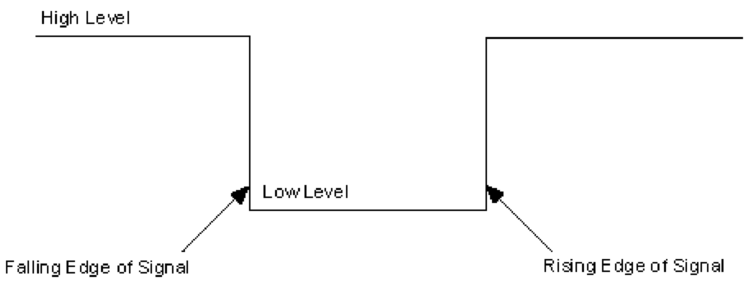
```
void init_timer0(void)
{
        LPC_TIM0->TCR = 0x00000002;     // Timer0 Reset
        LPC_TIM0->CTCR =0x00;
        LPC_TIM0->MR0 = 1500;
        LPC_TIM0->EMR = 0X00;
        LPC_TIM0->PR = 3000;
        LPC_TIM0->MCR = 0x00000005;
        LPC_TIM0->TCR = 0x00000001;     // Timer0 Enable
        LPC_GPIO0->FIOSET=0x00000004;
        return;
}
int main(void)
{
        LPC_GPIO0->FIODIR=0x00000004;
        init_timer0();
        NVIC_EnableIRQ(TIMER0_IRQn);
        while(1);
}
```

External Hardware Interrupts

System Control Block of ARM has SFRs to handle External Hardware Interrupts.

- Level Triggered- Level 0 or Level 1 triggered
- Edge triggered – Rising Edge or Falling Edge

LPC1768 has four external interrupts EINT0-EINT3



Port Pin	PINSEL_FUNC_0	PINSEL_FUNC_1
P2.10	GPIO	**EINT0**
P2.11	GPIO	**EINT1**
P2_12	GPIO	**EINT2**
P2.13	GPIO	**EINT3**

External Hardware Interrupts

EINT Registers

Register	Description
PINSELx	To configure the pins as External Interrupts
EXTINT	External Interrupt Flag Register contains interrupt flags for EINT0,EINT1, EINT2 & EINT3.
EXTMODE	External Interrupt Mode register(Level/Edge Triggered)
EXTPOLAR	External Interrupt Polarity(Falling/Rising Edge, Active Low/High)

External Hardware Interrupts

EXTINT				
31:4	3	2	1	0
RESERVED	EINT3	EINT2	EINT1	EINT0

EINTx: Bits will be set whenever the interrupt is detected on the particular interrupt pin. If the interrupts are enabled then the control goes to ISR.

Writing one to specific bit will clear the corresponding interrupt.

EXTMODE				
31:4	3	2	1	0
RESERVED	EXTMODE3	EXTMODE2	EXTMODE1	EXTMODE0

EXTMODEx: These bits are used to select whether the EINTx pin is level or edge Triggered

0: EINTx is Level Triggered.
1: EINTx is Edge Triggered.

External Hardware Interrupts

EXTPOLAR				
31:4	3	2	1	0
RESERVED	EXTPOLAR3	EXTPOLAR2	EXTPOLAR1	EXTPOLAR0

EXTPOLARx: These bits are used to select polarity(LOW/HIGH, FALLING/RISING) of the EINTx interrupt depending on the EXTMODE register.
0: EINTx is Active Low or Falling Edge (depending on EXTMODEx).
1: EINTx is Active High or Rising Edge (depending on EXTMODEx).

EXTMODEx	EXTPOLARx	EINTx
0	0	Level 0
0	1	Level 1
1	0	Falling Edge
1	1	Rising Edge

External Hardware Interrupts

Steps to Configure External Hardware Interrupts

- Configure the pins as external interrupts in PINSELx register.
- Configure the EINTx as Edge/Level triggered in EXTMODE register.
- Select the polarity(Falling/Rising Edge, Active Low/High) of the interrupt in EXTPOLAR register.
- Finally enable the interrputs by calling NVIC_EnableIRQ(EINTx_IRQn)
- Clear the interrupt in EXTINT after entering ISR.

External Hardware Interrupts

Toggle LED connected to P1.23 at each negative edge of the input applied at P2.12 (EINT2, Function-01)

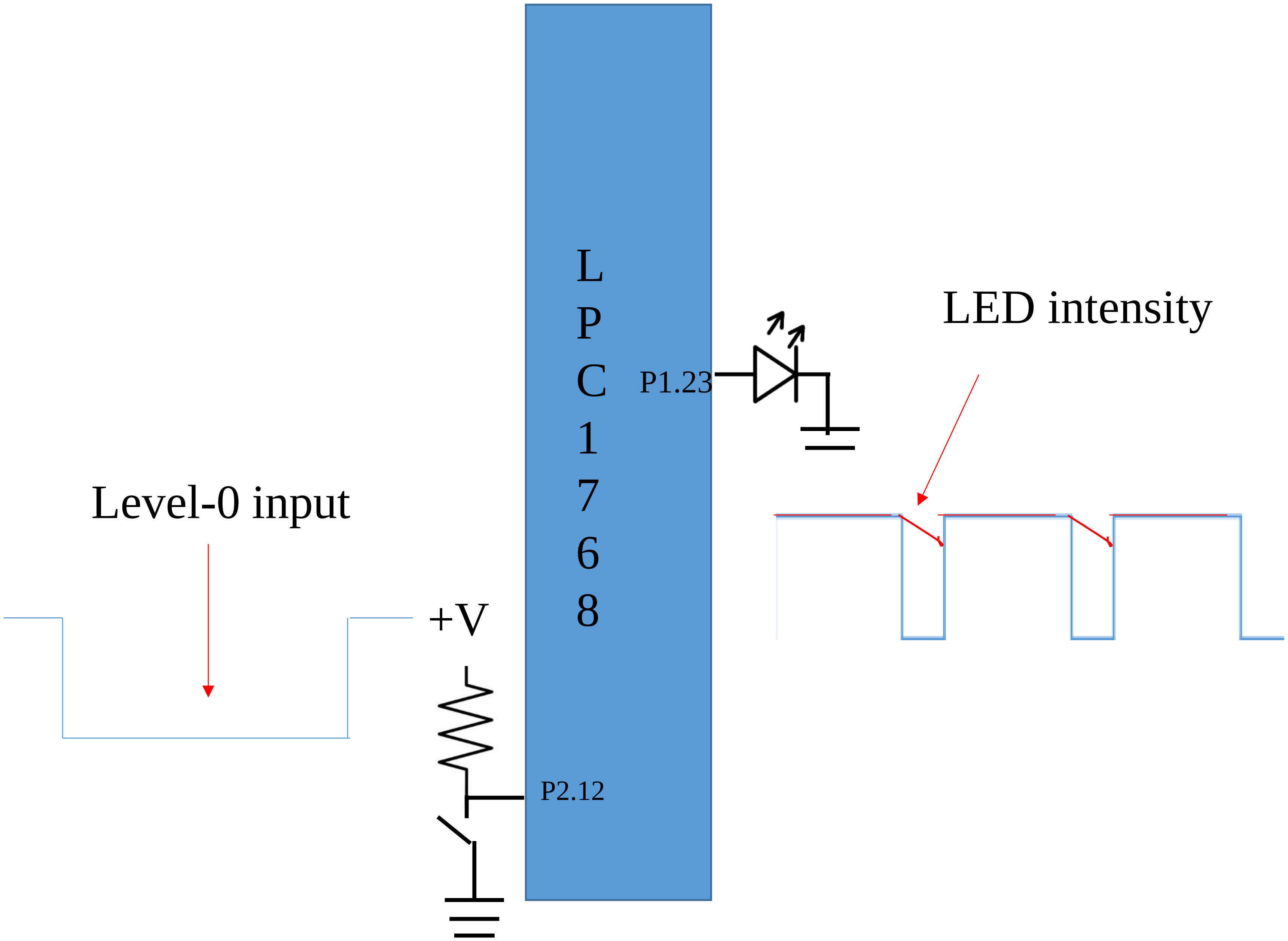
```
include<LPC17xx.h>
 void EINT2_IRQHandler(void);
 int main(void)
 {

            SystemInit();
            SystemCoreClockUpdate();

            LPC_PINCON->PINSEL4 |= (1<<24);          //P2.12 as EINT2 i.e FUNCTION-01
            LPC_GPIO1->FIODIR = 0x00800000;          //P1.23 is assigned output
            LPC_SC->EXTMODE = 0x00000004;            //EINT2 is initiated as edge sensitive, 0 for level
            LPC_SC->EXTPOLAR = 0x00000000;           //EINT2 is falling edge sensitive, 1 for rising edge
            NVIC_EnableIRQ(EINT2_IRQn);
            while(1) ;
             }

 void EINT2_IRQHandler(void)
 {
LPC_SC->EXTINT = 0x00000004;    //clears the interrupt
LPC_GPIO1->FIOPIN = ~ LPC_GPIO1->FIOPIN ;

 }
```

External Hardware Interrupts

Turn ON the LED connected to P1.23 whenever the switch connected to P2.12 (EINT2, Function-01) is pressed LED remains ON as long as the switch is pressed (Assume, when the switch is pressed Logic-0 is INPUT.

```
#include<LPC17xx.h>
 void EINT2_IRQHandler(void);
 int main(void)
 {
		LPC_PINCON->PINSEL4 |= (1<<24);			//P2.12 as EINT2 i.e FUNCTION-01
		LPC_GPIO1->FIODIR = 0x00800000;				//P1.23 is assigned output
		LPC_SC->EXTMODE = 0x00000000;				//EINT2  as level-0 sensitive
		LPC_SC->EXTPOLAR = 0x00000000;
		NVIC_EnableIRQ(EINT2_IRQn);
		 while(1) ;

 }
 void EINT2_IRQHandler(void)
 {
		LPC_SC->EXTINT = 0x00000004;	//clear the interrupt
		LPC_GPIO1->FIOSET = 1<<23; //LED ON
		for (i=0;i<255;i++);
		LPC_GPIO1->FIOCLR = 1<<23; LED OFF

 }
```

External Hardware Interrupts



GPIO Interrupts

The pins of Port-0 and Port-2 can generate GPIO interrupts.
GPIO interrupts are mapped to EINT3 ISR

GPIO overall Interrupt Status register (IOIntStatus)

Bit	Symbol	Value	Description
0	P0Int		Port 0 GPIO interrupt pending.
		0	There are no pending interrupts on Port 0.
		1	There is at least one pending interrupt on Port 0.
1	-	-	Reserved. The value read from a reserved bit is not defined.
2	P2Int		Port 2 GPIO interrupt pending.
		0	There are no pending interrupts on Port 2.
		1	There is at least one pending interrupt on Port 2.
31:2	-	-	Reserved. The value read from a reserved bit is not defined.

GPIO Interrupts

GPIO Interrupt Enable for port 0 Rising Edge (IO0IntEnR)

Bit	Symbol	Value	Description
0	P0.0ER		Enable rising edge interrupt for P0.0.
		0	Rising edge interrupt is disabled on P0.0.
		1	Rising edge interrupt is enabled on P0.0.
1	P0.1ER		Enable rising edge interrupt for P0.1.
2	P0.2ER		Enable rising edge interrupt for P0.2.
3	P0.3ER		Enable rising edge interrupt for P0.3.
4	P0.4ER[1]		Enable rising edge interrupt for P0.4.
5	P0.5ER[1]		Enable rising edge interrupt for P0.5.
6	P0.6ER		Enable rising edge interrupt for P0.6.
7	P0.7ER		Enable rising edge interrupt for P0.7.
8	P0.8ER		Enable rising edge interrupt for P0.8.
9	P0.9ER		Enable rising edge interrupt for P0.9.
10	P0.10ER		Enable rising edge interrupt for P0.10.
11	P0.11ER		Enable rising edge interrupt for P0.11.
14:12	-		Reserved
15	P0.15ER		Enable rising edge interrupt for P0.15.
16	P0.16ER		Enable rising edge interrupt for P0.16.

Bit	Symbol	Value	Description
17	P0.17ER		Enable rising edge interrupt for P0.17.
18	P0.18ER		Enable rising edge interrupt for P0.18.
19	P0.19ER[1]		Enable rising edge interrupt for P0.19.
20	P0.20ER[1]		Enable rising edge interrupt for P0.20.
21	P0.21ER[1]		Enable rising edge interrupt for P0.21.
22	P0.22ER		Enable rising edge interrupt for P0.22.
23	P0.23ER[1]		Enable rising edge interrupt for P0.23.
24	P0.24ER[1]		Enable rising edge interrupt for P0.24.
25	P0.25ER		Enable rising edge interrupt for P0.25.
26	P0.26ER		Enable rising edge interrupt for P0.26.
27	P0.27ER[1]		Enable rising edge interrupt for P0.27.
28	P0.28ER[1]		Enable rising edge interrupt for P0.28.
29	P0.29ER		Enable rising edge interrupt for P0.29.
30	P0.30ER		Enable rising edge interrupt for P0.30.
31	-		Reserved.

Similarly ----- GPIO Interrupt Enable for port 2 (P2.0-P2.13) Rising Edge (IO2IntEnR)

GPIO Interrupts

GPIO Interrupt Enable for port 0 Falling Edge (IO0IntEnF)

Bit	Symbol	Value	Description
0	P0.0EF		Enable falling edge interrupt for P0.0
		0	Falling edge interrupt is disabled on P0.0.
		1	Falling edge interrupt is enabled on P0.0.
1	P0.1EF		Enable falling edge interrupt for P0.1.
2	P0.2EF		Enable falling edge interrupt for P0.2.
3	P0.3EF		Enable falling edge interrupt for P0.3.
4	P0.4EF[1]		Enable falling edge interrupt for P0.4.
5	P0.5EF[1]		Enable falling edge interrupt for P0.5.
6	P0.6EF		Enable falling edge interrupt for P0.6.
7	P0.7EF		Enable falling edge interrupt for P0.7.
8	P0.8EF		Enable falling edge interrupt for P0.8.
9	P0.9EF		Enable falling edge interrupt for P0.9.
10	P0.10EF		Enable falling edge interrupt for P0.10.
11	P0.11EF		Enable falling edge interrupt for P0.11.
14:12	-		Reserved.
15	P0.15EF		Enable falling edge interrupt for P0.15.
16	P0.16EF		Enable falling edge interrupt for P0.16.
17	P0.17EF		Enable falling edge interrupt for P0.17.
18	P0.18EF		Enable falling edge interrupt for P0.18.
19	P0.19EF[1]		Enable falling edge interrupt for P0.19.
20	P0.20EF[1]		Enable falling edge interrupt for P0.20.
21	P0.21EF[1]		Enable falling edge interrupt for P0.21.
22	P0.22EF		Enable falling edge interrupt for P0.22.
23	P0.23EF[1]		Enable falling edge interrupt for P0.23.
24	P0.24EF[1]		Enable falling edge interrupt for P0.24.
25	P0.25EF		Enable falling edge interrupt for P0.25.
26	P0.26EF		Enable falling edge interrupt for P0.26.
27	P0.27EF[1]		Enable falling edge interrupt for P0.27.
28	P0.28EF[1]		Enable falling edge interrupt for P0.28.
29	P0.29EF		Enable falling edge interrupt for P0.29.
30	P0.30EF		Enable falling edge interrupt for P0.30.
31	-		Reserved.

Similarly ----- GPIO Interrupt Enable for port 2 (P2.0-P2.13) Falling Edge (IO2IntEnF)

GPIO Interrupts

GPIO Interrupt Status for port 0 Rising Edge Interrupt (IO0IntStatR)

Bit	Symbol	Value	Description
0	P0.0REI		Status of Rising Edge Interrupt for P0.0
		0	A rising edge has not been detected on P0.0.
		1	Interrupt has been generated due to a rising edge on P0.0.
1	P0.1REI		Status of Rising Edge Interrupt for P0.1.
2	P0.2REI		Status of Rising Edge Interrupt for P0.2.
3	P0.3REI		Status of Rising Edge Interrupt for P0.3.
4	P0.4REI[1]		Status of Rising Edge Interrupt for P0.4.
5	P0.5REI[1]		Status of Rising Edge Interrupt for P0.5.
6	P0.6REI		Status of Rising Edge Interrupt for P0.6.
7	P0.7REI		Status of Rising Edge Interrupt for P0.7.
8	P0.8REI		Status of Rising Edge Interrupt for P0.8.
9	P0.9REI		Status of Rising Edge Interrupt for P0.9.
10	P0.10REI		Status of Rising Edge Interrupt for P0.10.
11	P0.11REI		Status of Rising Edge Interrupt for P0.11.
14:12	-		Reserved.
15	P0.15REI		Status of Rising Edge Interrupt for P0.15.
16	P0.16REI		Status of Rising Edge Interrupt for P0.16.
17	P0.17REI		Status of Rising Edge Interrupt for P0.17.
18	P0.18REI		Status of Rising Edge Interrupt for P0.18.
19	P0.19REI[1]		Status of Rising Edge Interrupt for P0.19.
20	P0.20REI[1]		Status of Rising Edge Interrupt for P0.20.
21	P0.21REI[1]		Status of Rising Edge Interrupt for P0.21.
22	P0.22REI		Status of Rising Edge Interrupt for P0.22.
23	P0.23REI[1]		Status of Rising Edge Interrupt for P0.23.
24	P0.24REI[1]		Status of Rising Edge Interrupt for P0.24.
25	P0.25REI		Status of Rising Edge Interrupt for P0.25.
26	P0.26REI		Status of Rising Edge Interrupt for P0.26.
27	P0.27REI[1]		Status of Rising Edge Interrupt for P0.27.
28	P0.28REI[1]		Status of Rising Edge Interrupt for P0.28.
29	P0.29REI		Status of Rising Edge Interrupt for P0.29.
30	P0.30REI		Status of Rising Edge Interrupt for P0.30.
31	-		Reserved.

Similarly ----- GPIO Interrupt Status for port 2 (P2.0-P2.13) Rising Edge Interrupt (IO2IntStatR)

GPIO Interrupts

GPIO Interrupt Status for port 0 Falling Edge Interrupt (IO0IntStatF)

Bit	Symbol	Value	Description
0	P0.0FEI		Status of Falling Edge Interrupt for P0.0
		0	A falling edge has not been detected on P0.0.
		1	Interrupt has been generated due to a falling edge on P0.0.
1	P0.1FEI		Status of Falling Edge Interrupt for P0.1.
2	P0.2FEI		Status of Falling Edge Interrupt for P0.2.
3	P0.3FEI		Status of Falling Edge Interrupt for P0.3.
4	P0.4FEI[1]		Status of Falling Edge Interrupt for P0.4.
5	P0.5FEI[1]		Status of Falling Edge Interrupt for P0.5.
6	P0.6FEI		Status of Falling Edge Interrupt for P0.6.
7	P0.7FEI		Status of Falling Edge Interrupt for P0.7.
8	P0.8FEI		Status of Falling Edge Interrupt for P0.8.
9	P0.9FEI		Status of Falling Edge Interrupt for P0.9.
10	P0.10FEI		Status of Falling Edge Interrupt for P0.10.
11	P0.11FEI		Status of Falling Edge Interrupt for P0.11.
14:12	-		Reserved.
15	P0.15FEI		Status of Falling Edge Interrupt for P0.15.
16	P0.16FEI		Status of Falling Edge Interrupt for P0.16.
17	P0.17FEI		Status of Falling Edge Interrupt for P0.17.
18	P0.18FEI		Status of Falling Edge Interrupt for P0.18.
19	P0.19FEI[1]		Status of Falling Edge Interrupt for P0.19.
20	P0.20FEI[1]		Status of Falling Edge Interrupt for P0.20.
21	P0.21FEI[1]		Status of Falling Edge Interrupt for P0.21.
22	P0.22FEI		Status of Falling Edge Interrupt for P0.22.
23	P0.23FEI[1]		Status of Falling Edge Interrupt for P0.23.
24	P0.24FEI[1]		Status of Falling Edge Interrupt for P0.24.
25	P0.25FEI		Status of Falling Edge Interrupt for P0.25.
26	P0.26FEI		Status of Falling Edge Interrupt for P0.26.
27	P0.27FEI[1]		Status of Falling Edge Interrupt for P0.27.
28	P0.28FEI[1]		Status of Falling Edge Interrupt for P0.28.
29	P0.29FEI		Status of Falling Edge Interrupt for P0.29.
30	P0.30FEI		Status of Falling Edge Interrupt for P0.30.
31	-		Reserved.

Similarly ----- GPIO Interrupt Status for port 2 (P2.0-P2.13) Falling Edge Interrupt (IO2IntStatF)

GPIO Interrupts

GPIO Interrupt Clear register for port 0 (IO0IntClr)

Bit	Symbol	Value	Description
0	P0.0CI		Clear GPIO port Interrupts for P0.0
		0	Corresponding bits in IOxIntStatR and IOxIntStatF are unchanged.
		1	Corresponding bits in IOxIntStatR and IOxStatF are cleared
1	P0.1CI		Clear GPIO port Interrupts for P0.1.
2	P0.2CI		Clear GPIO port Interrupts for P0.2.
3	P0.3CI		Clear GPIO port Interrupts for P0.3.
4	P0.4CI[1]		Clear GPIO port Interrupts for P0.4.
5	P0.5CI[1]		Clear GPIO port Interrupts for P0.5.
6	P0.6CI		Clear GPIO port Interrupts for P0.6.
7	P0.7CI		Clear GPIO port Interrupts for P0.7.
8	P0.8CI		Clear GPIO port Interrupts for P0.8.
9	P0.9CI		Clear GPIO port Interrupts for P0.9.
10	P0.10CI		Clear GPIO port Interrupts for P0.10.
11	P0.11CI		Clear GPIO port Interrupts for P0.11.
14:12	-		Reserved.
15	P0.15CI		Clear GPIO port Interrupts for P0.15.
16	P0.16CI		Clear GPIO port Interrupts for P0.16.
17	P0.17CI		Clear GPIO port Interrupts for P0.17.
18	P0.18CI		Clear GPIO port Interrupts for P0.18.
19	P0.19CI[1]		Clear GPIO port Interrupts for P0.19.
20	P0.20CI[1]		Clear GPIO port Interrupts for P0.20.
21	P0.21CI[1]		Clear GPIO port Interrupts for P0.21.
22	P0.22CI		Clear GPIO port Interrupts for P0.22.
23	P0.23CI[1]		Clear GPIO port Interrupts for P0.23.
24	P0.24CI[1]		Clear GPIO port Interrupts for P0.24.
25	P0.25CI		Clear GPIO port Interrupts for P0.25.
26	P0.26CI		Clear GPIO port Interrupts for P0.26.
27	P0.27CI[1]		Clear GPIO port Interrupts for P0.27.
28	P0.28CI[1]		Clear GPIO port Interrupts for P0.28.
29	P0.29CI		Clear GPIO port Interrupts for P0.29.
30	P0.30CI		Clear GPIO port Interrupts for P0.30.
31	-		Reserved.

Similarly ----- GPIO Interrupt Clear Register for port 2 (P2.0-P2.13) (IO2IntClr)

GPIO Interrupts

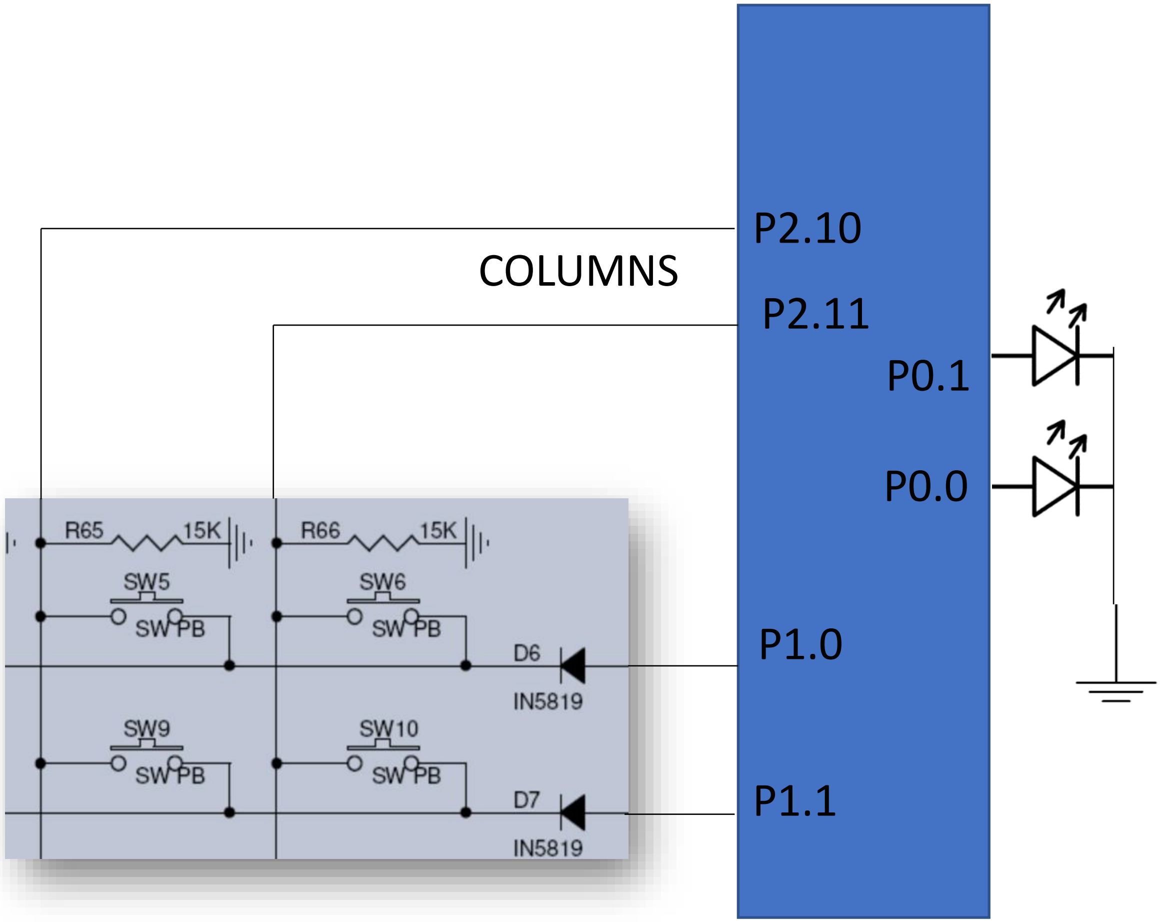
Turn ON the LED connected to P1.23 whenever the +ve edge applied to P0.0 and Turn OFF whenever the +ve edge applied at P0.1

```
#include<LPC17xx.h>
 void EINT3_IRQHandler(void);
  int main(void)
 {

        SystemInit();
        SystemCoreClockUpdate();
        LPC_GPIO1->FIODIR = 1<<23;                 //P1.23 is assigned output
        LPC_GPIO1->FIOCLR 1<<23;          //Initially LED is kept OFF
        LPC_GPIOINT->IO0IntEnR=0x00000003;    //P0.0 and P0.1 - Enable Rising Edge
        NVIC_EnableIRQ(EINT3_IRQn);                //Enable EINT3
        while(1) ;
         }
 void EINT3_IRQHandler(void)
 {

         unsigned int x=LPC_GPIOINT->IO0IntStatR; // Get the status of Rising Edge Interrupts
        LPC_GPIOINT->IO0IntClr |=x; //Clear the Interrupt
         if (x==0x00000001) //If  Rising Edge at P0.0
                LPC_GPIO1->FIOSET = 0x00800000;
         else   //If Rising edge at P0.1
                LPC_GPIO1->FIOCLR = 0x00800000;


 }
```

GPIO Interrupts

Assume that columns of a 2x2 matrix keyboard are connected to P2.10-P2.11 and rows are connected to P1.0-P1.1. Write an embedded C program using GPIO interrupt to display the keycode of the key pressed on LEDs connected to P0.0 to P0.1.

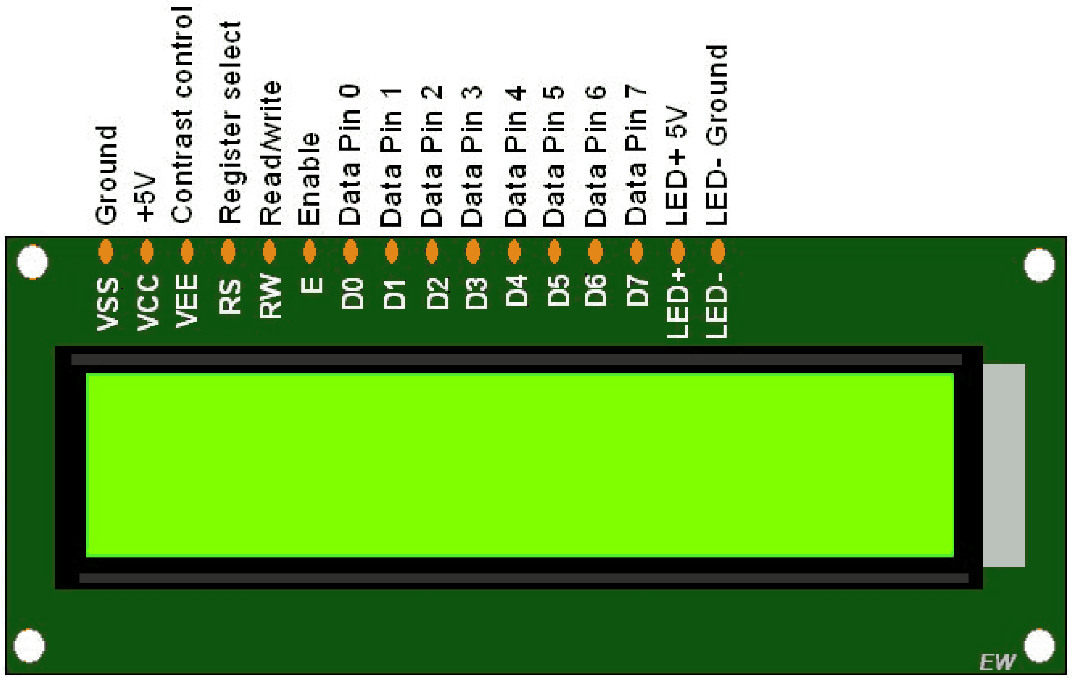
```
#include <lpc17xx.h>
void EINT3_IRQHandler (void);
unsigned int row, col ;
int main(void)
{
LPC_GPIO0->FIODIR =0x03;//p0.0 and p0.1 output
LPC_GPIO1->FIODIR =0x03;// p1.0 to p1.2 output
 LPC_GPIO1->FIOSET =0x03;// Facilitate any key press detection
LPC_GPIOINT->IO2IntEnR= 1<<10 | 1<<11; //Rising edge- P2.10,P2.11
NVIC_EnableIRQ(EINT3_IRQn);//Enable GPIO INTR
  while(1);
 }
```

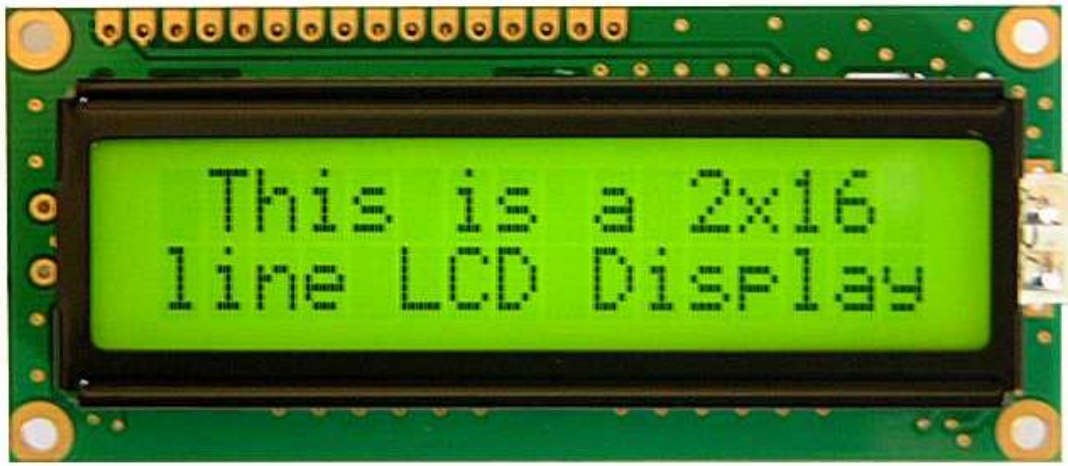


GPIO Interrupts

```
void EINT3_IRQHandler (void)
{
unsigned int temp3;
temp3 = LPC_GPIOINT->IO2IntStatR; /
if (temp3 == 1<<10)
col = 0;
else if (temp3 == 1<<11)
 col = 1;
LPC_GPIOINT->IO2IntClr=1<<10 | 1<<11;//Clear Interrupt
for (row=0;row <2;row++)
{
if (row==0)
temp=0x01;
else if (row==1)
temp=0x02;
LPC_GPIO1->FIOPIN=temp;
if ( (LPC_GPIO2->FIOPIN & (1<<10) | (1<<11)) !=  0) //Read the columns
{
LPC_GPIO0->FIOPIN=row *2+col; //Display the keycode
LPC_GPIO1->FIOSET=0x03; // Facilitate any keypress detection
break;
}
}
}
```

Liquid Crystal Display (LCD) Interfacing





16×2 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

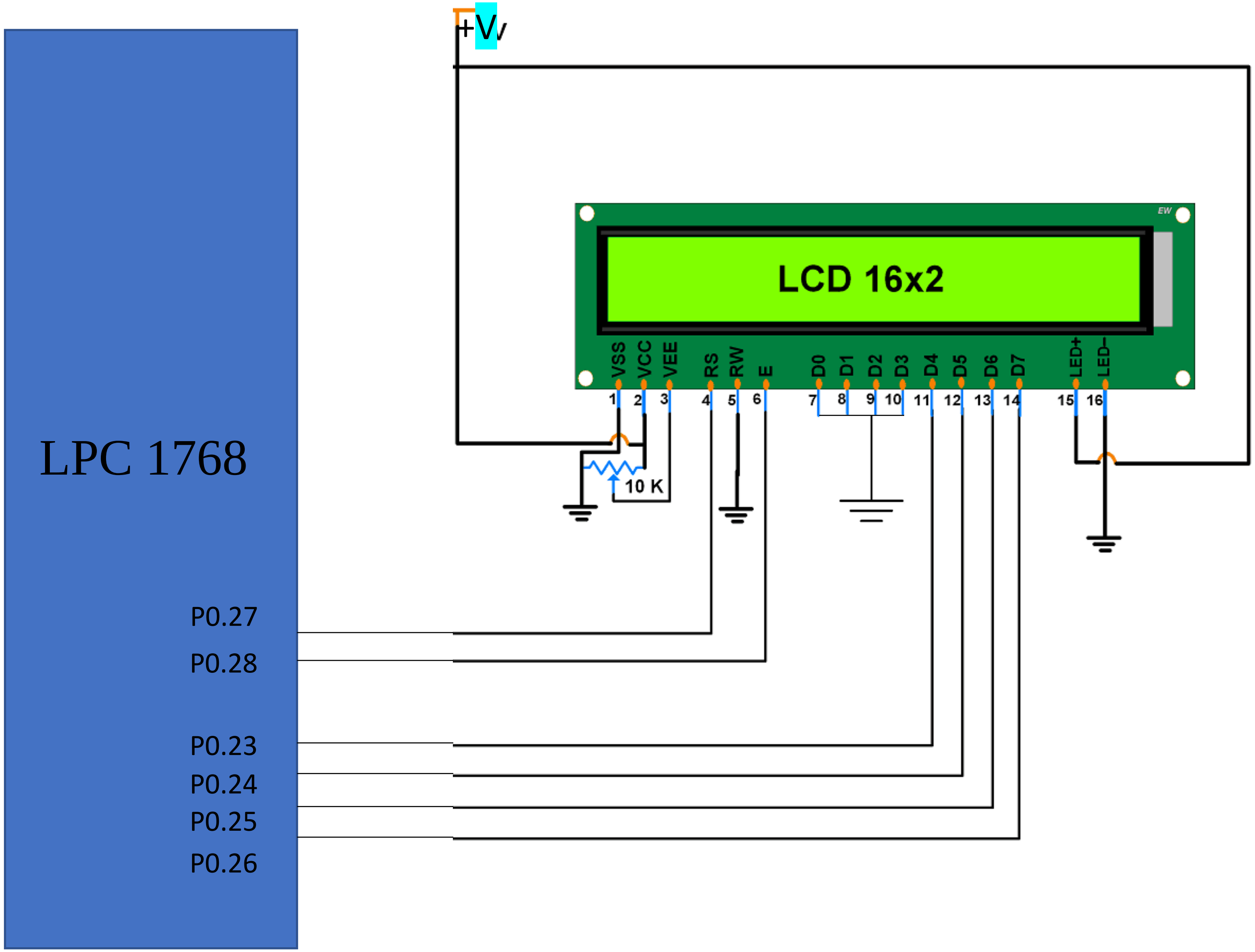
	Code											Execution Time (max) (when f_{cp} or
Instruction	RS	R/$\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description	f_{osc} is 270 kHz)
Clear display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DDRAM address 0 in address counter.	
Return home	0	0	0	0	0	0	0	0	1	—	Sets DDRAM address 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

LCD Command Codes

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-bt 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)

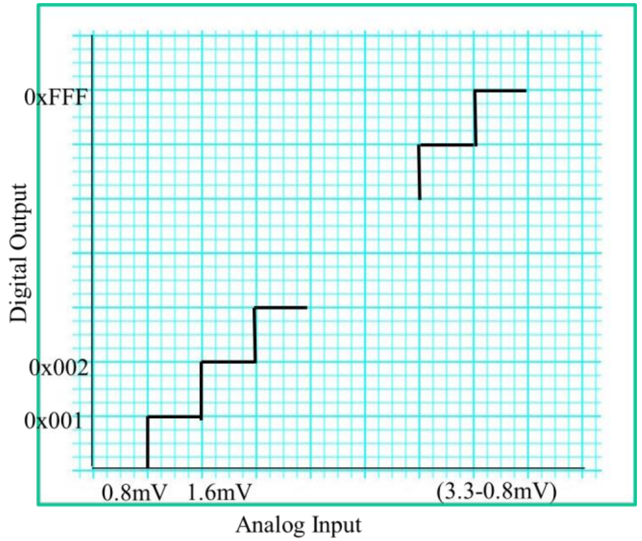
Analog voltage and Digital value of the ADC are related as follows:

$V_A = (V_{REFP} / 2^N)$ (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 (0.805x2 =1.6) mV output decimal value is 2
When analog voltage is (0.805x3 =2.4) mV output decimal value is 3
When analog voltage is (0.805x4095 = 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

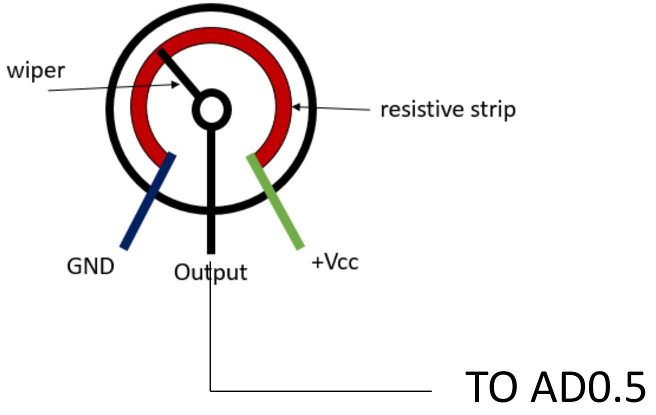
```
#include<LPC17xx.h>
#include<stdio.h>
int main(void)

{                                  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
```



```
		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
		   }
```

```
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;
}
}
```