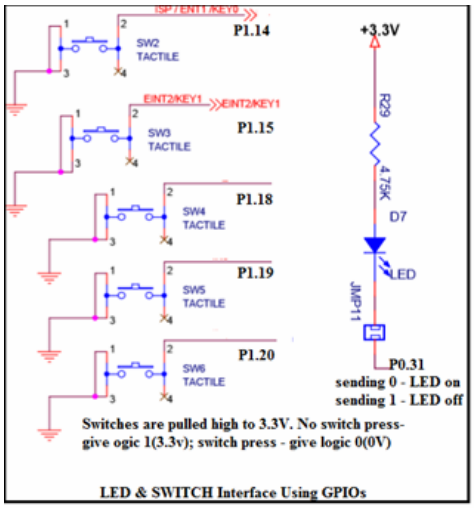
Sample program : Interfacing LED and Switches

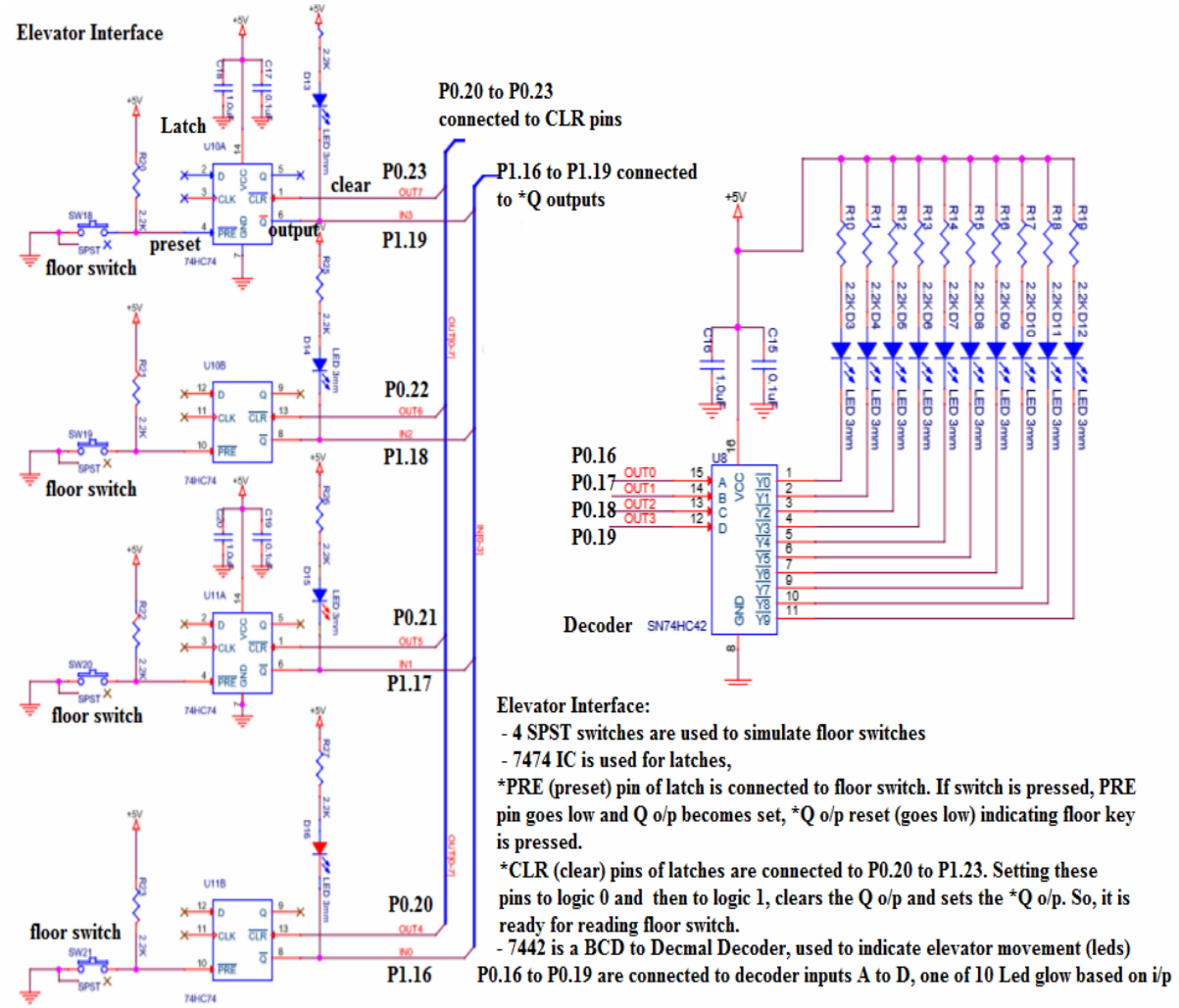
Interfacing Diagram




```
//Sample Program 1:  Interfacing LED and Switch to LPC2148 using GPIO pins
  //P0.31 connected to LED - D7 in CPU board(common anode)
  //P1.14 connected to Switch - SW2 in CPU board
#include <lpc214x.h>
#define LED_OFF (IO0SET = 1U << 31)
#define LED_ON (IO0CLR = 1U << 31)
#define SW2 (IO0PIN & (1 << 14))
void delay_ms(unsigned int j);
int main( )
{
    IO0DIR = 1U << 31;
    IO0SET = 1U << 31;
    while(1)
     {
          if (!(IO0PIN & (1 << 14)))//(if(!SW2 )
           {
                    IO0CLR = 1U << 31; //LED_ON
                    delay_ms(250);
                    IO0SET = 1U << 31; //LED_OFF
                    delay_ms(250);
           }
     }
}
void delay_ms(unsigned int j)
{
 unsigned int x, i;
 for(i=0; i<j; i++)
  {
    for(x=0; x<10000; x++);    /* loop to generate 1 milisecond delay with CCLK = 60MHz */
  }
}
```

Program 1: Interface Logic Controller and write Embedded C programs to generate BCD up / down and Ring counters. Input is read from the DIP switch.

Interfacing Diagram



//Elevator Program:

// P0.16 - P0.19 are connected to decoder inputs, it makes one of the o/p LEDs 0 to 9 on

*// P0.20-P0.23 are connected to *CLR pins of latches: make it '0' and then '1' to clear*

*// elevator keys: *Q outputs of latches connected to P1.16 TO P1.19*

```
#include<lpc214x.h>
#define IS_ON(pin) (IO1PIN & (1U << (pin)))

void delay_ms(unsigned int x);
void reset_values(int y);

int contUP = 0;
int contDN = 99;
unsigned int rightSFT = 1U<<7;
unsigned int leftSFT = 1;
const int key0 = 16;
const int key1 = 17;
const int key2 = 18;
const int key3 = 19;

int main()
{
	IO0DIR |= 0xFF;

	while(1)
	{
		if(IS_ON(key0))
		{
			reset_values(0);
			IO0CLR = 0xFF<<16;
			IO0SET |= ((contUP/10)<<4 | (contUP%10))<<16;
			contUP++;
			if(contUP > 99) contUP = 0;
		}
		else if(IS_ON(key1))
		{
			reset_values(1);
			IO0CLR = 0xFF<<16;
			IO0SET |= ((contDN/10)<<4 | (contDN%10)) << 16;
			contDN--;
			if(contDN < 0) contDN = 99;
		}
		else if(IS_ON(key2))
		{
			reset_values(2);
			IO0CLR = 0xFF<<16;
			IO0SET |= leftSFT<<16;
			leftSFT<<=1;
			if(leftSFT > 1U<<7) leftSFT = 1;
		}
		else if(IS_ON(key3))
		{
			reset_values(3);
			IO0CLR = 0xFF<<16;
			IO0SET |= rightSFT<<16;
```

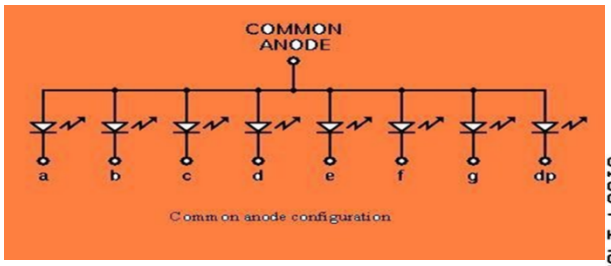
```
                rightSFT>>=1;
                if(rightSFT < 1) rightSFT = 1U<<7;
            }
            delay_ms(100);
        }
}

void reset_values(int y)
{
        switch(y)
        {
            case 0: contDN = 99;
                                    rightSFT = 1U<<7;
                                    leftSFT = 1;
                                    break;
            case 1: contUP = 0;
                                    rightSFT = 1U<<7;
                                    leftSFT = 1;
                                    break;
            case 2: contUP = 0;
                                    contDN = 99;
                                    rightSFT = 1U<<7;
                                    break;
            case 3: contUP = 0;
                                    contDN = 99;
                                    leftSFT = 1;
                                    break;
        }
}

void delay_ms(unsigned int ms) {
        for(int i = 0; i < ms; i++) {
            for(int x = 0; x < 10000; x++);
        }
}
```

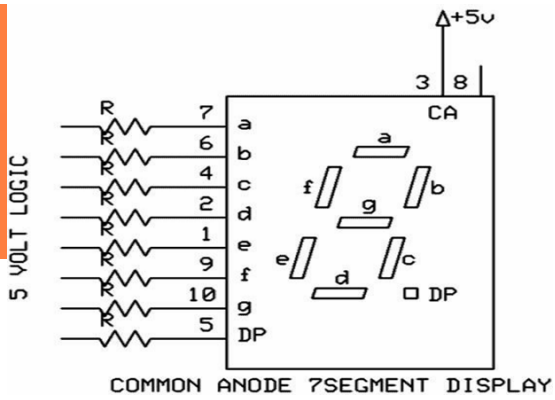
Interfacing Circuit working Explanation:

Output Observation:

Program 2: Seven Segment Display Interface: Write a C program to display messages "FIRE" & "HELP" on 4 digit seven segment display alternately with a suitable delay.

Serial In Parallel Out mode of Shift Register (74HC4094) is used to send 8 bits of data to seven segment display. Seven segment display used is of common anode type i.e. we have to send 0 to make corresponding segment ON and 1 to make it OFF.

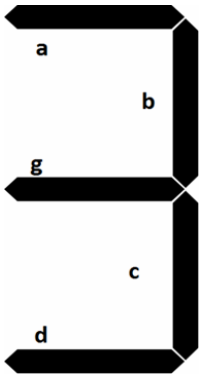




To display 3, we have to send following bit pattern,

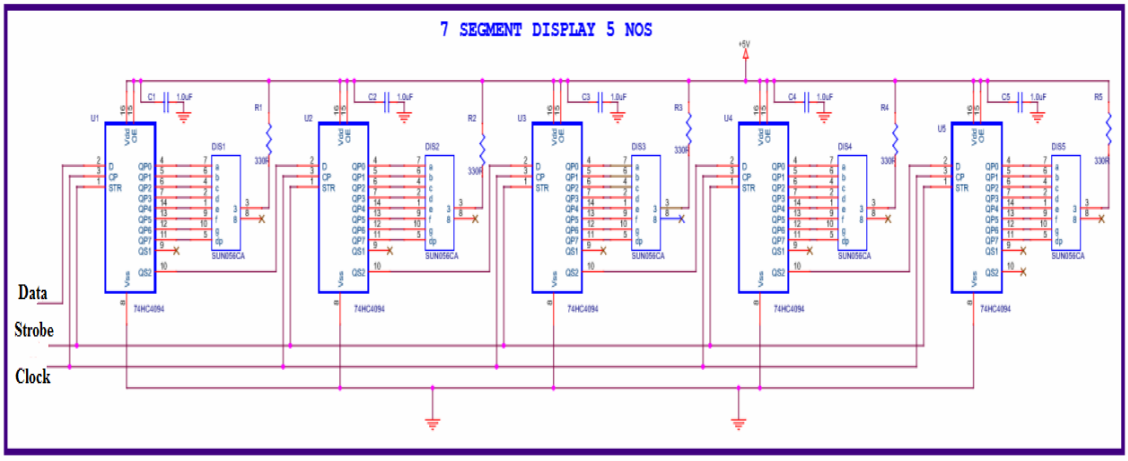
DP	G	f	e	d	c	b	a
1	0	1	1	0	0	0	0

This is B0 in hexadecimal. To send B0H we have to start sending the bits from MSB onwards i.e D7 first, D6 next and so on with D0 being the last



Clock pulses are required to clock in the data, 8 clock pulses for one byte of data. As shift registers are cascaded, 8*4=32 clocks are required to clock in 4 bytes of data. To send "12345", first we have to send '1', then '2','3' ,'4' and lastly '5'. All the shift registers are cascaded, the data is fed to the shift register using serial in parallel out method. Strobe is used to copy the shifted data to the output pins. STB is generated after shifting is comleted.

Interfacing Diagram




```
//Seven Segment Display Program:
//P0.19 Data pin of 1st shift register
//P0.20 Clock pin of shift registers, make 1 to 0
//P0.30 Strobe pin of shift registers: 1 to 0

#include <lpc214x.h>
#define LED_OFF (IO0SET = 1U << 31)
#define LED_ON (IO0CLR = 1U << 31)
#define PLOCK 0x00000400
void delay_ms(unsigned int j);
void SystemInit(void);
unsigned char getAlphaCode(unsigned char alphachar);
void alphadisp7SEG(char *buf);
int main()
{
      IO0DIR |= 1U << 31 | 1U << 19 | 1U << 20 | 1U << 30 ; // to set as o/ps
       LED_ON; // make D7 Led on .. just indicate the program is running
       SystemInit();
       while(1)
          {
            alphadisp7SEG("fire ");
            delay_ms(500);
            alphadisp7SEG("help ");
            delay_ms(500);
          }

}
```

```
unsigned char getAlphaCode(unsigned char alphachar)
{
  switch (alphachar)
        {
        // dp g f e d c b a - common anode: 0 segment on, 1 segment off
        case 'f': return 0x8e;
        case 'i': return 0xf9;
        case 'r': return 0xce;
        case 'e':return 0x86;  // 1000 0110
        case 'h':return 0x89;
        case 'l': return 0xc7;
        case 'p':return 0x8c;
        case ' ': return 0xff;
        //simmilarly add for other digit/characters
        default : break;
        }
  return 0xff;
}
void alphadisp7SEG(char *buf)
{
  unsigned char  i,j;
  unsigned char seg7_data,temp=0;
  for(i=0;i<5;i++) // because only 5 seven segment digits are present
    {
      seg7_data = getAlphaCode(*(buf+i));  //instead of this look up table can be used
                  //to shift the segment data(8bits)to the hardware (shift registers) using
Data,Clock,Strobe
        for (j=0 ; j<8; j++)
        {
              //get one bit of data for serial sending
              temp = seg7_data & 0x80;  // shift data from Most significan bit (D7)
              if(temp == 0x80)
                    IOSET0 |= 1 << 19; //IOSET0 | 0x00080000;
              else
                    IOCLR0 |= 1 << 19; //IOCLR0 | 0x00080000;
              //send one clock pulse
              IOSET0 |= 1 << 20;  //IOSET0 | 0x00100000;
              delay_ms(1);
              IOCLR0 |= 1 << 20;  //IOCLR0 | 0x00100000;
              seg7_data = seg7_data << 1; // get next bit into D7 position
        }
    }
```

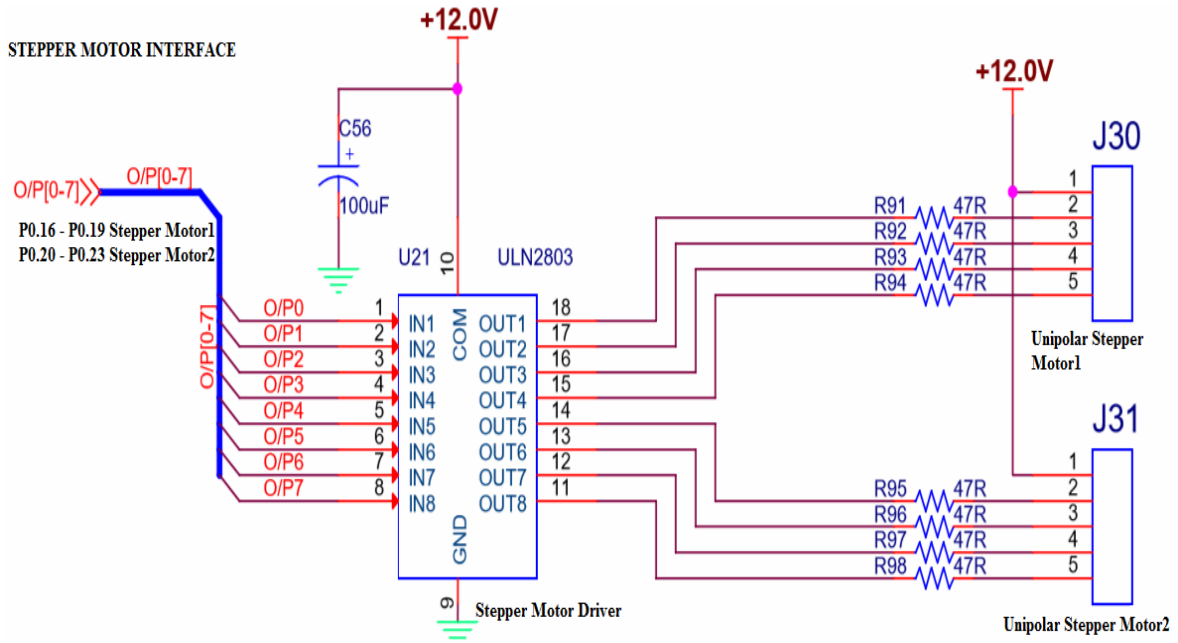
```
    // send the strobe signal
    IOSET0 |= 1 << 30; //IOSET0 | 0x40000000;
    delay_ms(1);      //nop();
    IOCLR0 |= 1 << 30;  //IOCLR0 | 0x40000000;
    return;
}
 void SystemInit(void)
 {
    PLL0CON = 0x01;
    PLL0CFG = 0x24;
    PLL0FEED = 0xAA;
    PLL0FEED = 0x55;
    while( !( PLL0STAT & PLOCK ))
    { ; }
    PLL0CON = 0x03;
    PLL0FEED = 0xAA;  // lock the PLL registers after setting the required PLL
    PLL0FEED = 0x55;
    VPBDIV = 0x01;      // PCLK is same as CCLK i.e 60Mhz
 }

 void delay_ms(unsigned int j)
 {
   unsigned int x,i;
   for(i=0;i<j;i++)
     {
        for(x=0; x<10000; x++);
     }
 }



 // CODE to display an integer number/long integer number
 // long int dig_value;
 // unsigned char buf[5];
 // sprintf(buf,"%05lu",dig_value);
 // alphadisp7SEG(&buf[0]);
```

Interfacing Circuit working Explanation:

Output Observation:

Program No.3: Stepper Motor Interface: Write an Embedded C program to rotate stepper motor in clockwise direction for "M" steps, anti-clock wise direction for "N" steps.

Interfacing circuit diagram



- Total number of steps for one revolution = 200 steps (200 teeth shaft)

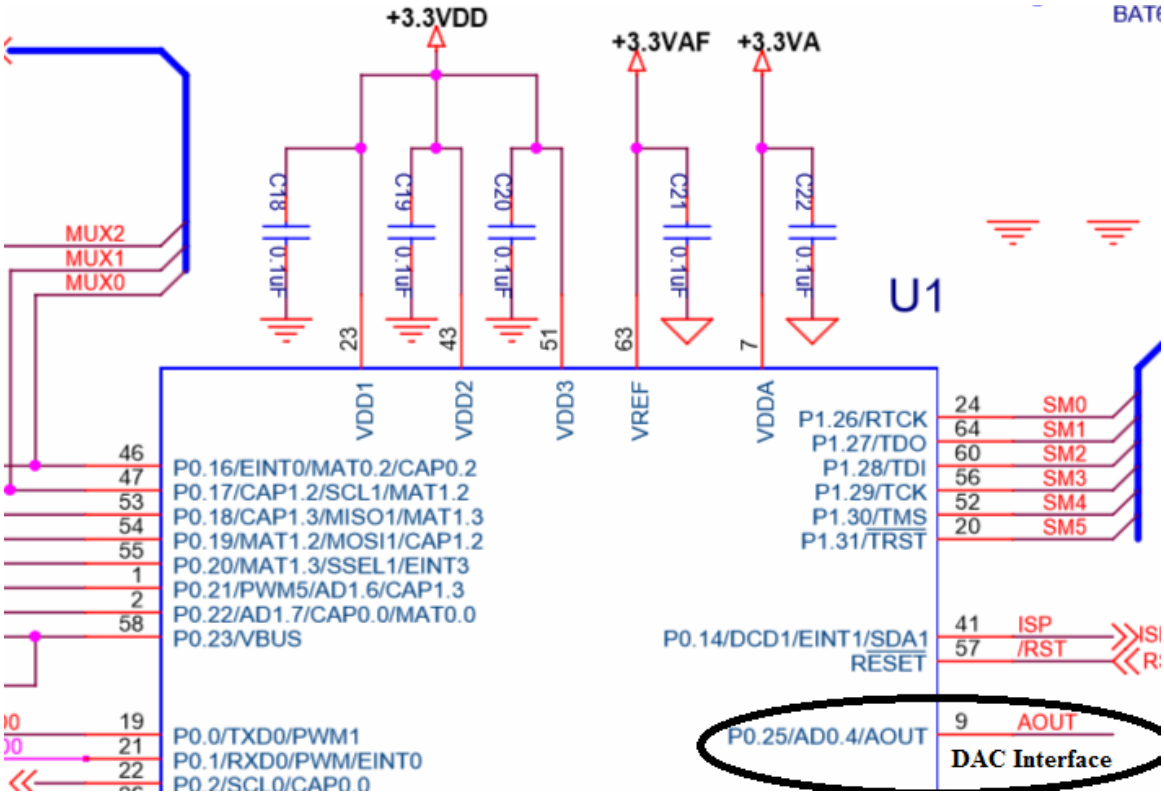
  Step angle = 360°/200 = 1.8°
- Use appropriate delay in between consequent steps
- 2Phase, 4winding stepper motor is used, along with driver circuit(ULN 2803) built on the RV All-In- One Card, 12v power is used to drive the stepper motor. Digital input generated by the microcontroller, is used to drive and control the direction and rotation of stepper motors. If it is required to drive bigger/higher torque stepper motors only change is- use MOSFETS or higher power stepper driver ICs to drive motors

```
//Stepper Motor Program:
//P0.16 to P0.19 are connected to Windings of SMotor

#include <lpc214x.h>
#define LED_OFF (IO0SET = 1U << 31)
#define LED_ON (IO0CLR = 1U << 31)
#define PLOCK 0x00000400
void delay_ms(unsigned int j);
void SystemInit(void);

int main()
{
    unsigned int  no_of_steps_clk = 100, no_of_steps_aclk = 100;
    IO0DIR |= 1U << 31 | 0x00FF0000 | 1U << 30; // to set P0.16 to P0.23 as o/ps
    LED_ON; delay_ms(500);LED_OFF; // make D7 Led on .. just indicate the program is running
    SystemInit( );
   do{
   IO0CLR = 0X000F0000;IO0SET = 0X00010000;delay_ms(10);if(--no_of_steps_clk == 0) break;
   IO0CLR = 0X000F0000;IO0SET = 0X00020000;delay_ms(10);if(--no_of_steps_clk == 0) break;
   IO0CLR = 0X000F0000;IO0SET = 0X00040000;delay_ms(10);if(--no_of_steps_clk == 0) break;
   IO0CLR = 0X000F0000;IO0SET = 0X00080000;delay_ms(10);if(--no_of_steps_clk == 0) break;
   }while(1);
   do{
   IO0CLR = 0X000F0000;IO0SET = 0X00080000;delay_ms(10);if(--no_of_steps_aclk == 0) break;
   IO0CLR = 0X000F0000;IO0SET = 0X00040000;delay_ms(10);if(--no_of_steps_aclk == 0) break;
   IO0CLR = 0X000F0000;IO0SET = 0X00020000;delay_ms(10);if(--no_of_steps_aclk == 0) break;
   IO0CLR = 0X000F0000;IO0SET = 0X00010000;delay_ms(10);if(--no_of_steps_aclk == 0) break;
   }while(1);
   IO0CLR = 0X00FF0000;
   while(1);
}

void delay_ms(unsigned int j)
{
 unsigned int x,i;
 for(i=0;i<j;i++)
  {
    for(x=0; x<10000; x++);
  }
}
```

Interfacing Circuit working Explanation:

Output Observation:

Program No.4: DAC Interface : Write an Embedded C program to generate sine, full rectified sine, Triangular, Sawtooth and Square waveforms using DAC module



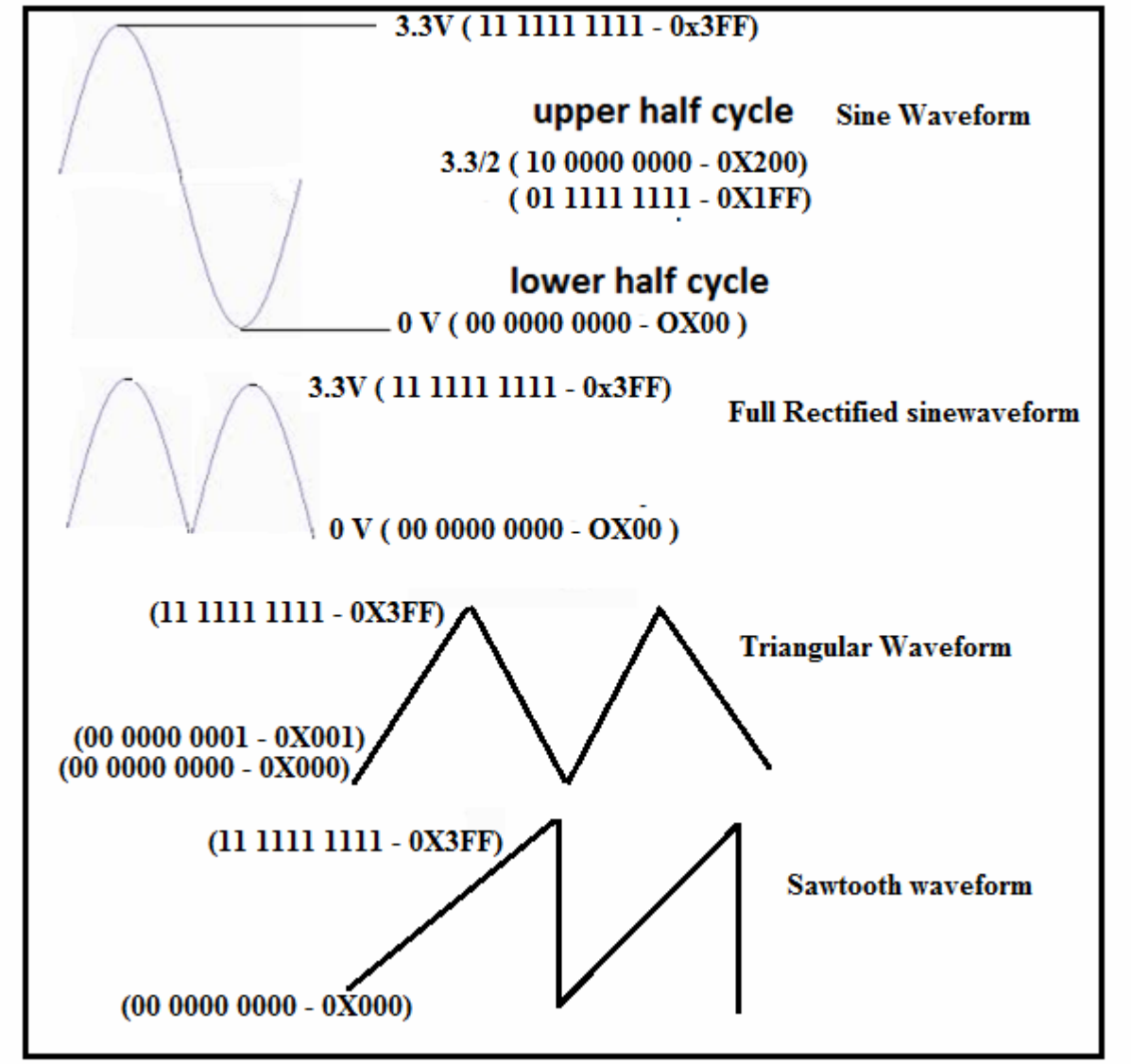
- DAC module of LPC 2148 is 10 bit Digital to Analog converter used to convert 10 bit Digital data to corresponding Analog voltage.
- Digital I/P : **000 to 3FF (0 to 1023)**, corresponding Analog O/P : **0V to 3.3V**
- Resolution = (3.3/1024) ≈ **3.2mili volts**

Look up Table Creation: Look up tables are used extensively in embedded systems, to store precomputed digital values, corresponding to analog voltages and used to generate different waveforms using DAC Module. Here the explanation about creating sine table is given.

Formula for calculation of the sine table entries: **512 + 511 x Sin Θ**
(512 Corresponds to 1FFh, i.e. 3.3/2 V, 511 x SIN 90 gives 511, so 512 + 511 = 1023 (for 3.3V). Calculate the digital values to be outputted to DAC for angles in the steps of 6°,

511 x sin 0 = 0	511 x sin 48 = 380
511 x sin 6 = 53	511 x sin 54 = 413
511 x sin 12 = 106	511 x sin 60 = 442
511 x sin 18 = 158	511 x sin 66 = 467
511 x sin 24 = 208	511 x sin 72 = 486
511 x sin 30 = 256	511 x sin 80 = 503
511 x sin 36 = 300	511 x sin 86 = 510
511 x sin 42 = 342	511 x sin 90 = 511

Output the above values in the reverse order to get other portion of the top half cycle, (add 512 for top half cycle, and subtract from 512 for the lower half cycle, refer the table declaration).




```
//Alpha-numeric LCD Interface (4Lines,20characters)
//Connected in 4bit nibble mode
//LCD handshaking:RS->P0.20,EN->P0.25 ,R/W -Gnd
//LCD data:D4,D5,D6,D7 -> P0.16,P0.17,P0.18,P0.19
```

```
#include <lpc214x.h>
#include <stdio.h>
#define PLOCK 0x00000400
#define LED_OFF (IO0SET = 1U << 31)
#define LED_ON  (IO0CLR = 1U << 31)
#define SW2 (IO0PIN & (1 << 14))
#define SW3 (IO0PIN & (1 << 15))
#define SW4 (IO1PIN & (1 << 18))
#define SW5 (IO1PIN & (1 << 19))
#define SW6 (IO1PIN & (1 << 20))
void SystemInit(void);
static void delay_ms(unsigned int j);//millisecond delay
short int sine_table[ ] =
{512+0,512+53,512+106,512+158,512+208,512+256,512+300,512+342,512+380,512+413,
512+442,512+467,512+486,512+503,512+510,512+511,
512+510,512+503,512+486,512+467,512+442,512+413,512+380,512+342,512+300,512+25
6,512+208,512+158,512+106,512+53,512+0,
512-53,512-106,512-158,512-208,512-256,512-300,512-342,512-380,512-413,512-442,512-
467,512-486,512-503,512-510,512-511,
512-510,512-503,512-486,512-467,512-442,512-413,512-380,512-342,512-300,512-
256,512-208,512-158,512-106,512-53};
short int sine_rect_table[ ] =
{512+0,512+53,512+106,512+158,512+208,512+256,512+300,512+342,512+380,512+413,
512+442,512+467,512+486,512+503,512+510,512+511,
512+510,512+503,512+486,512+467,512+442,512+413,512+380,512+342,512+300,512+25
6,512+208,512+158,512+106,512+53,512+0};

int main()
{
 short int value,i=0;
 SystemInit();
 PINSEL1 |= 0x00080000;     /* P0.25 as DAC output :option 3 - 10 (bits18,19)*/
 IO0DIR |= 1U << 31 | 0x00FF0000 ; // to set P0.16 to P0.23 as o/ps

 while(1)
        {
          if (!SW2)    /* If switch for sine wave is pressed */
            {
                while (i!=60 )
                 {
                    value = sine_table[i++];
                    DACR = ( (1<<16) | (value<<6) );
                    delay_ms(1);
                 }
                i=0;
             }
```

```
        else if (!SW3)
            {
                while ( i!=30 )
                 {
                   value = sine_rect_table[i++];
                   DACR = ( (1<<16) | (value<<6) );
                   delay_ms(1);
                 }
                i=0;
            }
        else if ( !SW4)          /* If switch for triangular wave is pressed */
            {
                value = 0;
                while ( value != 1023 )
                  {
                    DACR = ( (1<<16) | (value<<6) );
                    value++;
                  }
                while ( value != 0 )
                  {
                   DACR = ( (1<<16) | (value<<6) );
                   value--;
                  }
            }
        else if ( !SW5 )         /* If switch for sawtooth wave is pressed */
           {
                value = 0;
                while ( value != 1023 )
                  {
                   DACR = ( (1<<16) | (value<<6) );
                   value++;
                  }
           }
        else if ( !SW6 )         /* If switch for square wave is pressed */
           {
                value = 1023;
                DACR = ( (1<<16) | (value<<6) );
                delay_ms(1);
                value = 0;
                DACR = ( (1<<16) | (value<<6) );
                delay_ms(1);
           }
        else     /* If no switch is pressed, 3.3V DC */
         {
                value = 1023;
                DACR = ( (1<<16) | (value<<6) );
        }
    }
}
```

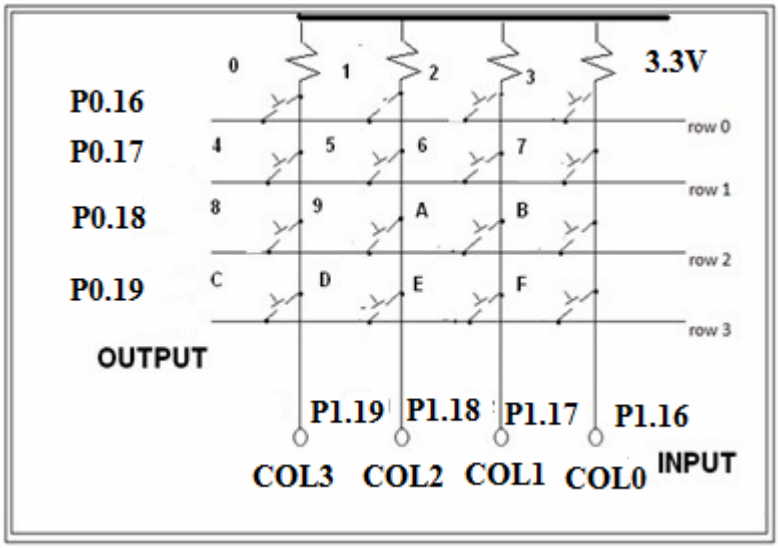
```
void SystemInit(void)
{
  PLL0CON = 0x01;
  PLL0CFG = 0x24;
  PLL0FEED = 0xAA;
  PLL0FEED = 0x55;
  while( !( PLL0STAT & PLOCK ))
  {  ;  }
  PLL0CON = 0x03;
  PLL0FEED = 0xAA;
  PLL0FEED = 0x55;
}
void delay_ms(unsigned int j)
{
 unsigned int x,i;
 for(i=0;i<j;i++)
  {
    for(x=0; x<10000; x++);
  }
}
```

Interfacing Circuit working Explanation:

Output Observation:

Program No.5: Matrix Keyboard Interface : Write an embedded C program to interface 4 X 4 matrix keyboard using lookup table and display the key pressed on the Terminal.

Interfacing Diagram



Working method:

- If no key is pressed, we will have on columns 0-3, '1111' on P1.16 to P1.19, as all the inputs are pulled up by pull up resistors.
- If we press any key, let '0' key be pressed, it will short row0 and col0 lines (P0.16 & P1.19), so whatever data (0 or 1) available at row0 (P0.16) is available at col0 (P1.19). Since already columns are pulled high, it is required to apply logic '0' to see change in col0 when the key is pressed.
- To identify which key is pressed,
  - Check for a key press in first row by out putting – '0111'on row's, check which column data is changed, if no key press go for next row
  - Check for a key press in second row by out putting – '1011'on row's, check which column data is changed, if no key press go for next row
  - Check for a key press in third row by out putting – '1101'on row's, check which column data is changed, if no key press go for next row
  - Check for a key press in last row by out putting – '1110'on row's, if no key is pressed go for the first row again
- Once the key press is found, use the row number and column number and look up table to convert the key position corresponding to ascii code. Use appropriate delay for debouncing.

```
//Matrix 4 x 4 Keyboard
//Columns & Rows are pulled to +5v,if dont press key, we receive '1' on columns
//Method: Sending '0' to a selected row, checking for '0' on each column
//ROWS - ROW0-ROW3 -> P0.16,P0.17,P0.18,P0.19
//COLS - COL0-COL3 -> P1.19,P1.18,P1.17,P1.16
#include <lpc214x.h>
#define PLOCK 0x00000400
#define LED_OFF (IO0SET = 1U << 31)
#define LED_ON  (IO0CLR = 1U << 31)
#define COL0 (IO1PIN & 1 <<19)
#define COL1 (IO1PIN & 1 <<18)
#define COL2 (IO1PIN & 1 <<17)
#define COL3 (IO1PIN & 1 <<16)
void SystemInit(void);
void delay_ms(unsigned int j);
void uart_init(void);
unsigned char lookup_table[4][4]={  {'0', '1', '2','3'},
                                    {'4', '5', '6','7'},
                                    {'8', '9', 'a','b'},
                                    {'c', 'd', 'e','f'}};
unsigned char rowsel=0,colsel=0;
int main( )
{
 SystemInit();
 uart_init();//initialize UART0 port
 IO0DIR |= 1U << 31 | 0x00FF0000; // to set P0.16 to P0.23 as o/ps
//make D7 Led on off for testing
 LED_ON; delay_ms(500);LED_OFF;delay_ms(500);
 do
   {
     while(1)
       {
        //check for keypress in row0,make row0 '0',row1=row2=row3='1'
        rowsel=0;IO0SET = 0X000F0000;IO0CLR = 1 << 16;
        if(COL0==0){colsel=0;break;};if(COL1==0){colsel=1;break;};
        if(COL2==0){colsel=2;break;};if(COL3==0){colsel=3;break;};
        //check for keypress in row1,make row1 '0'
        rowsel=1;IO0SET = 0X000F0000;IO0CLR = 1 << 17;
        if(COL0==0){colsel=0;break;};if(COL1==0){colsel=1;break;};
        if(COL2==0){colsel=2;break;};if(COL3==0){colsel=3;break;};
        //check for keypress in row2,make row2 '0'
        rowsel=2;IO0SET = 0X000F0000;IO0CLR = 1 << 18;//make row2 '0'
        if(COL0==0){colsel=0;break;};if(COL1==0){colsel=1;break;};
```

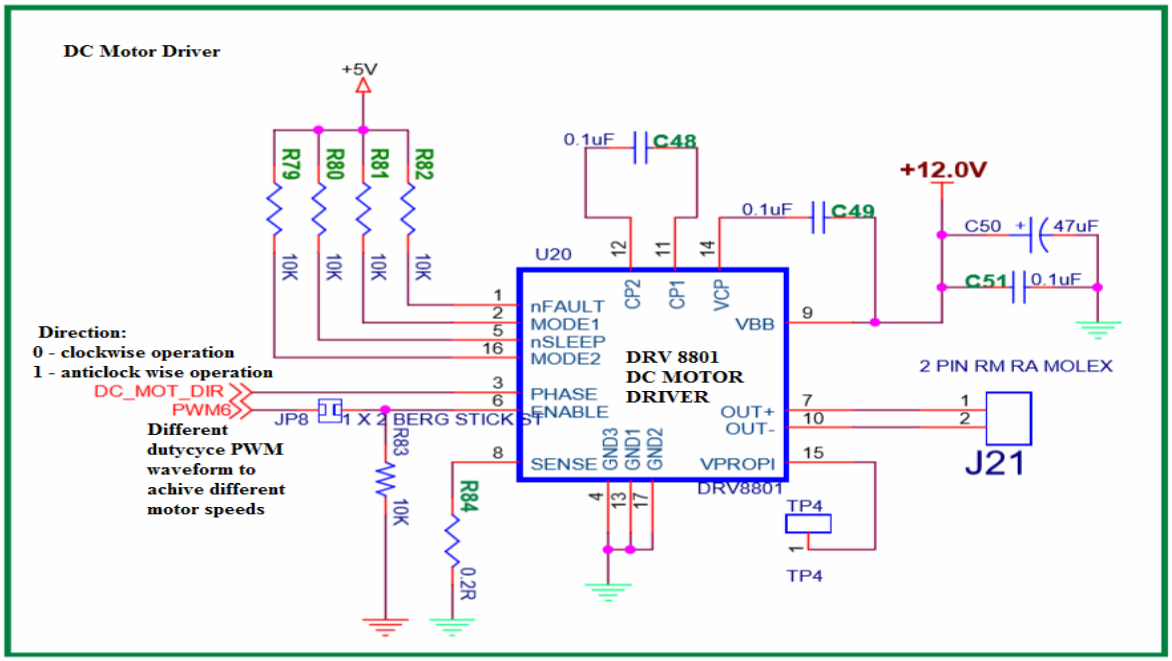
```
        if(COL2==0){colsel=2;break;};if(COL3==0){colsel=3;break;};
        //check for keypress in row3,make row3 '0'
        rowsel=3;IO0SET = 0X000F0000;IO0CLR = 1 << 19;//make row3 '0'
        if(COL0==0){colsel=0;break;};if(COL1==0){colsel=1;break;};
        if(COL2==0){colsel=2;break;};if(COL3==0){colsel=3;break;};
      };
  delay_ms(50);  //allow for key debouncing
  while(COL0==0 || COL1==0 || COL2==0 || COL3==0);//wait for key release
  delay_ms(50);   //allow for key debouncing
  IO0SET = 0X000F0000; //disable all the rows
  U0THR = lookup_table[rowsel][colsel];  //send to serial port(check on the terminal)
 }
 while(1);
}
void uart_init(void)
{
 //configurations to use serial port
 PINSEL0 |= 0x00000005;  // P0.0 & P0.1 ARE CONFIGURED AS TXD0 & RXD0
 U0LCR = 0x83;   /* 8 bits, no Parity, 1 Stop bit    */
 U0DLM = 0; U0DLL = 8; // 115200 baud rate
 U0LCR = 0x03;  /* DLAB = 0                         */
 U0FCR = 0x07;  /* Enable and reset TX and RX FIFO. */
}
void SystemInit(void)
{
  PLL0CON = 0x01;
  PLL0CFG = 0x24;
  PLL0FEED = 0xAA;
  PLL0FEED = 0x55;
  while( !( PLL0STAT & PLOCK ))
  { ; }
  PLL0CON = 0x03;
  PLL0FEED = 0xAA;  // lock the PLL registers after setting the required PLL
  PLL0FEED = 0x55;
  VPBDIV = 0x01;      // PCLK is same as CCLK i.e 60Mhz
}
void delay_ms(unsigned int j)
{
 unsigned int x,i;
 for(i=0;i<j;i++)
  {
    for(x=0; x<10000; x++);
  }
}
```

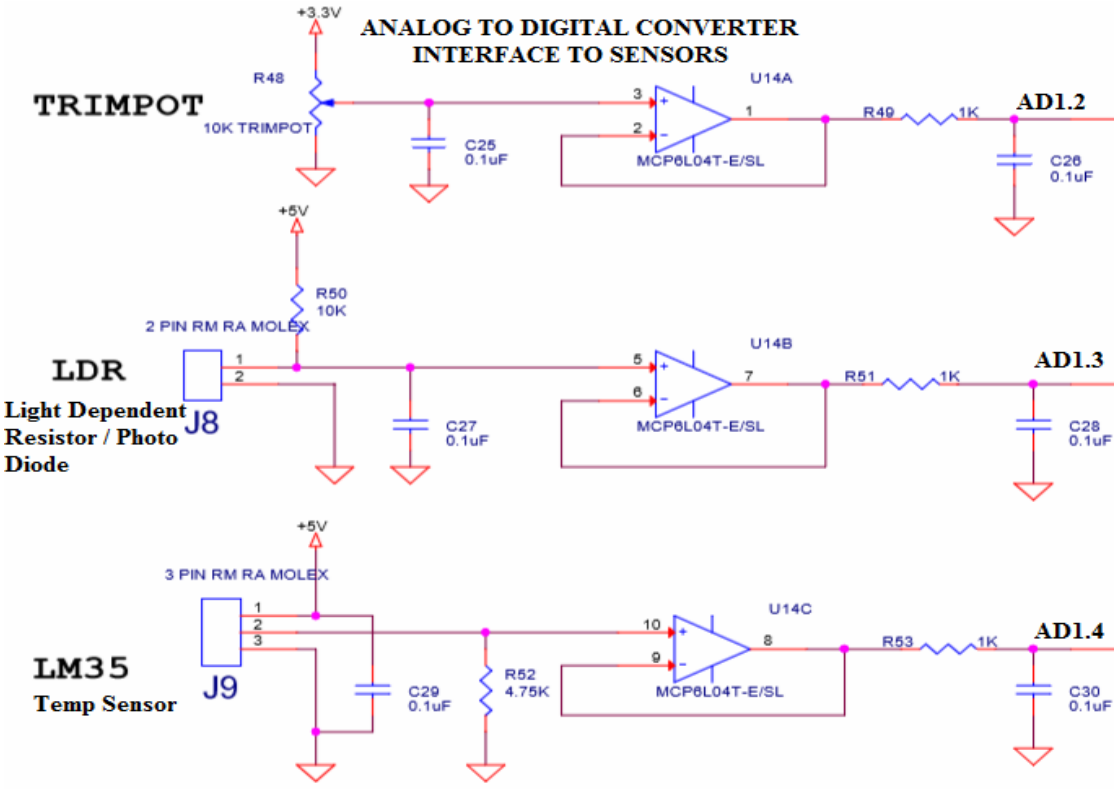
Interfacing Circuit working Explanation:

Output Observation:

Program No. 6: DC Motor Interface: Write an Embedded C program to generate PWM wave to control speed of DC motor. Control the duty cycle by analog input fed from potentiometer.

Interfacing Diagram





```
//DC Motor Speed Control
//P0.28 - used for direction control
//P0.9  - used for speed,generated by PWM6
//duty cycle - 0 to 100 controlled by PWM, fed from Potentiameter connected to ADC

#include <lpc214x.h>
#define LED_OFF (IO0SET = 1U << 31)
#define LED_ON (IO0CLR = 1U << 31)
#define PLOCK 0x00000400
void delay_ms(unsigned int j);
void SystemInit(void);
void runDCMotor(int direction,int dutycycle);
unsigned int adc(int no,int ch);

int main()
{
      int dig_val;
      IO0DIR |= 1U << 31 | 0x00FF0000 | 1U << 30; // to set P0.16 to P0.23 as o/ps
      LED_ON; delay_ms(500);LED_OFF; // make D7 Led on / off for program checking
      SystemInit( );
      do{
         dig_val = adc(1,2) / 10;
         if(dig_val > 100) dig_val =100;
         runDCMotor(2,dig_val); // run at 10% duty cycle
      }
      while(1);
}

void runDCMotor(int direction,int dutycycle)
{
        IO0DIR |= 1U << 28;     //set P0.28 as output pin
          PINSEL0 |= 2 << 18;   //select P0.9 as PWM6 (option 2)
        if (direction == 1)
                IO0SET = 1 << 28;   //set to 1, to choose anti-clockwise direction
         else
                IO0CLR = 1 << 28;  //set to 0, to choose clockwise direction

        PWMPCR = (1 << 14);  // enable PWM6
        PWMMR0 = 1000;   // set PULSE rate to value suitable for DC Motor operation
        PWMMR6 = (1000U*dutycycle)/100;  // set PULSE period
        PWMTCR = 0x00000009;  // bit D3 = 1 (enable  PWM), bit D0=1 (start the timer)
PWMLER = 0X70;  // load the new values to PWMMR0 and PWMMR6 registers
}
```

```
unsigned int adc(int no,int ch)
{
        // adc(1,4) for temp sensor LM34, digital value will increase as temp increases
        // adc(1,3) for LDR - digival value will reduce as the light increases
        // adc(1,2) for trimpot - digital value changes as the pot rotation
        unsigned int val;
        PINSEL0 |=  0x0F300000;   /* Select the P0_13 AD1.4 for ADC function */
                                  /* Select the P0_12 AD1.3 for ADC function */
                                  /* Select the P0_10 AD1.2 for ADC function */
     switch (no)       //select adc
        {
          case 0: AD0CR=0x00200600|(1<<ch);      //select channel
                AD0CR|=(1<<24);                  //start conversion
                while((AD0GDR& (1U<<31))==0);
                val=AD0GDR;
                break;

          case 1: AD1CR=0x00200600|(1<<ch);      //select channel
                AD1CR|=(1<<24);                  //start conversion
                while((AD1GDR&(1U<<31))==0);
                val=AD1GDR;
                break;
        }
   val=(val >> 6) & 0x03FF;        // bit 6:15 is 10 bit AD value
   return val;
}

void SystemInit(void)
{
  PLL0CON = 0x01;
  PLL0CFG = 0x24;
  PLL0FEED = 0xAA;
  PLL0FEED = 0x55;
  while( !( PLL0STAT & PLOCK ))
  { ; }
  PLL0CON = 0x03;
  PLL0FEED = 0xAA;  // lock the PLL registers after setting the required PLL
  PLL0FEED = 0x55;
  VPBDIV = 0x01;      // PCLK is same as CCLK i.e 60Mhz
}
void delay_ms(unsigned int j)
{
 unsigned int x,i;
 for(i=0;i<j;i++)
  {
    for(x=0; x<10000; x++);
  }
}
```