CHAPTER - 2

HARDWARE DETAILS

ViARM-2378 Development Board features a number of peripheral Devices. In order to enable these devices before programming, you need to check if appropriate jumpers have been properly set.

ViARM-2378 Development Board populated with ARM7 Core LPC2378 CPU.

On Board Peripherals

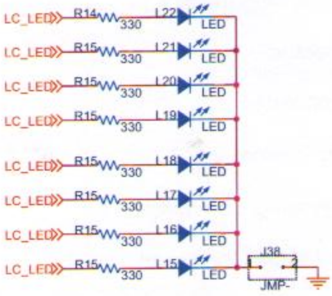
1. 8 Digital Outputs – LED.
2. 8 Digital Inputs - Switch.
3. 4 x 4 Matrix Keypad.
4. Character Based LCD (16 x 2 & 20 x 4).
5. Graphics LCD (128 x 64).
6. Two RS232 Port.
7. SD Card Interface.
8. I2C Peripherals.
    * Real Time Clock.
    * Serial EEPROM.
    * 7 Segment Display.
9. Two SPDT Relay.
10. Temperature Sensor.
11. Stepper Motor Interface.
12. IrDA.
13. 10/100 Base T Ethernet Interface.
14. USB 2.0 Interface.

15. 50-Pin Expansion Header.
16. Two CAN Port.
17. Accelerometer.
18. Joystick.
19. PS/2 Keyboard connector.
20. Jtag Connector.
21. J-Trace Connector.
22. USB Audio Device.
23. Digital To Analog Converter.
24. Analog To Digital Converter.
25. SPI Interface
26. Zigbee
27. OLED Interface
28. TFT LCD Interface

1. LED's

Light Emitting Diodes are the most commonly used components, usually for displaying Pin's Digital State. ViARM-2378 has 8 LED's that are connected to the Microcontroller Port Line.

While we using LEDs, close the Jumper J38.

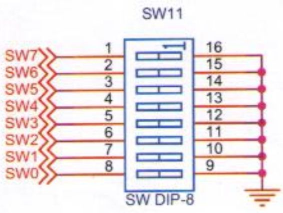


Used Port Lines:

LED0 – LED7 : P3.0 – P3.7

2. SWITCHES

Switches are devices that have two positions - ON and OFF, which have a toggle to establish or break a connection between two contacts. The ViARM-2378 Development Board has one 8-Way Dip Switch.

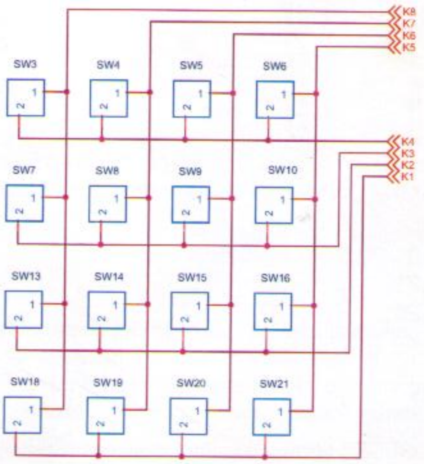


Used Port Lines:

SW0 – SW7 : P4.0 - P4.7

3. Matrix KEY

For Pin reduction, we can connect the Keys in the form of Matrix. In ViARM-2378 has 16 keys which are connected in Matrix format.

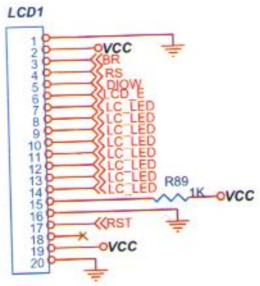


Used Port lines

K1 - K8	:	P4.8 - P4.15
Scan Lines	:	P4.8 - P4.11
Read lines	:	P4.12 - P4.15

4. LCD (Liquid Crystal Display) - Character Based LCD

A standard character LCD is probably the most widely used data Visualization component. Usually it can display four lines of 20 alphanumeric characters, each made up of 5x8 Pixels. The Character LCD communicates with the microcontroller via 8-bit data bus. The Connection to the Microcontroller is shown in below.



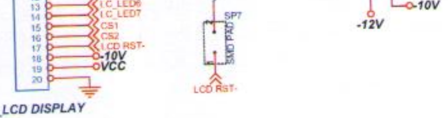
Used Port lines:

Data lines	:	P3.0 - P3.7
RS	:	P0.21
DIOW	:	P1.28
LCDEN	:	P3.23

5. Graphics LCD

A Graphics LCD (GLCD) allows advanced visual messages to be displayed. While a character LCD can display only alphanumeric characters, a GLCD can be used to display messages in the form of drawings and bitmaps.

The Most commonly used graphic LCD has the screen resolution of 128x64 Pixels. Before a GLCD is connected, the user needs to set the Jumpers. The GLCD's Contrast can be adjusted using the Potentiometer.



G_LCD DISPLAY

While we using the GLCD, close the jumpers J13 to J16 as Downward Direction and close the jumper J42 as left side direction

Used Port lines:

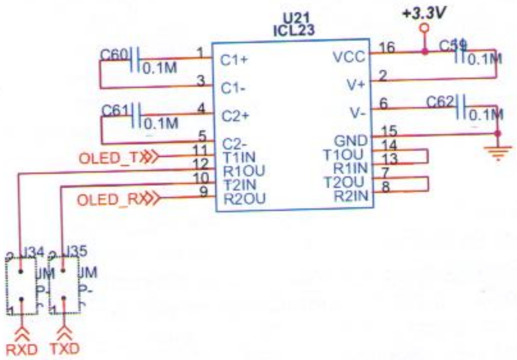
Data lines	:	P3.0 - P3.7
RS	:	P0.12
DIOW	:	P1.28
LCDEN	:	P3.23
CS1	:	P4.29
CS2	:	P4.28

Note:

Make sure to turn off the Power supply before placing GLCD on development board. If the Power supply is connected while placing, GLCD unit can be permanently damaged.

6. RS-232 Communication

RS-232 Communication enables point-to-point Data transfer. It is commonly used in data acquisition applications, for the transfer of data between the microcontroller and a PC. Since the Voltage levels of a microcontroller and PC are not directly compatible with each other, a level transition buffer such as the MAX232 must be used.



ViARM-2378 Development board has Two UART Termination at 9 pin D-type male connector.

While we using the UART1, close the jumpers J1 & J2

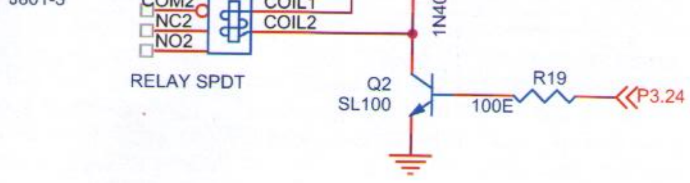
While we using the UART2, close the jumpers J3 & J4

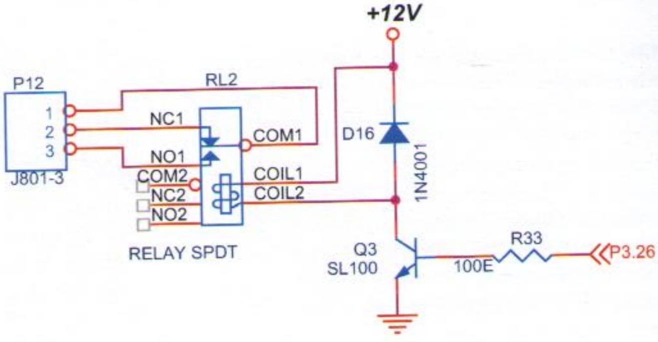
9. Relay

A **relay** is an electrical switch that opens and closes under the control of other an electrical circuit. In the original form, the switch is operated by an electromagnet to open or close one or many sets of contacts.

When a current flows through the coil, the resulting magnetic field attracts an armature that is mechanically linked to a moving contact. The movement either makes or breaks a connection with a fixed contact. When the current to the coil is switched off, a force approximately half as strong as the magnetic force returns the armature to its relaxed position. Usually this is a spring, but gravity is also used commonly in industrial motor starters. Most relays are manufactured to operate quickly. In a low voltage application, this is to reduce noise. In a high voltage or high current application, this is to reduce arcing.

If the coil is energized with DC, a diode is frequently installed across the coil, to dissipate the energy from the collapsing magnetic field at deactivation, which would otherwise generate a spike of voltage and might cause damage to circuit components. If the coil is designed to be energized with AC, a small copper ring can be crimped to the end of the solenoid. This "shading ring" creates a small out-of-phase current, which increases the minimum pull on the armature during the AC cycle. By analogy with the functions of the original electromagnetic device, a solid-state relay is made with a thyristor or other solid-state switching device. To achieve electrical isolation, a light-emitting diode (LED) is used with a phototransistor.



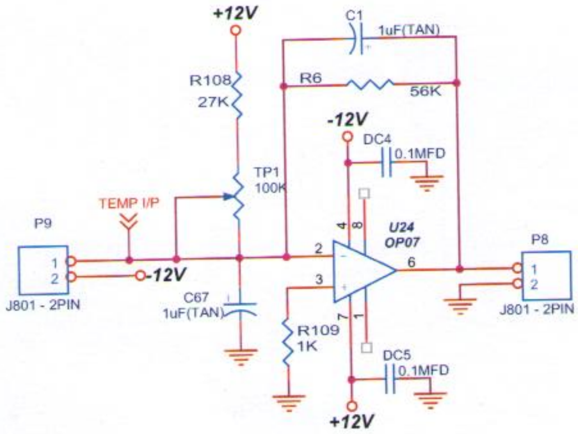


Used Port lines

P3.24 and P3.26.

10. Temperature Sensor

A signal conditioner is a device that converts one type of electronic signal into a another type of signal. Its primary use is to convert a signal that may be difficult to read by conventional instrumentation into a more easily read format. In performing this conversion a number of functions may take place. They include:



1. Amplification

When a signal is amplified, the overall magnitude of the signal is increased. Converting a 0-10mV signal to a 0 -10V signal is an example of amplification.

Electrical Isolation

Electrical isolation breaks the galvanic path between the input and output signal. That is, there is no physical wiring between the input and output. The input is normally transferred to the output by converting it to an optical or magnetic signal then it is reconstructed on the output. By breaking the galvanic path between input and output, unwanted signals on the input line are prevented from passing through to the output. Isolation is required when a measurement must be made on a surface with a voltage potential far above ground. Isolation is also used to prevent ground loops.

Linearization

Converting a non-linear input signal to a linear output signal. This is common for thermocouple signals.

Cold Junction Compensation

Used for thermocouples. The thermocouple signal is adjusted for fluctuations in room temperature.

Excitation

Many sensors require some form of excitation for them to operate. Strain gages and RTDs are two common examples. thermocouple input

11. Stepper Motor

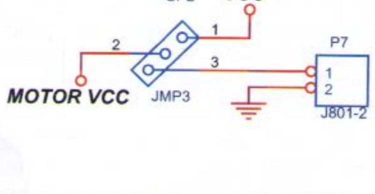
Stepper motors operate differently from normal DC motors, which simply spin when voltage is applied to their terminals. Stepper motors, on the other hand, effectively have multiple "toothed" electromagnets arranged around a central metal gear, as shown at right. To make the motor shaft turn, first one electromagnet is given power, which makes the gear's teeth magnetically attracted to the electromagnet's teeth. When the gear's teeth are thus aligned to the first electromagnet, they are slightly offset from the next electromagnet. So when the next electromagnet is turned on and the first is turned off, the gear rotates slightly to align with the next one, and from there the process is repeated. Each of those slight rotations is called a "step." In that way, the motor can be turned a precise angle. There are two basic arrangements for the electromagnetic coils: bipolar and unipolar.

Theory

A step motor can be viewed as a DC motor with the number of poles (on both rotor and stator) increased, taking care that they have no common denominator. Additionally, soft magnetic material with many teeth on the rotor and stator cheaply multiplies the number of poles (reluctance motor). Like an AC synchronous motor, sinusoidal current, allowing a stepless operation, ideally drives it but this puts some burden on the controller. When using an 8-bit digital controller, 256 micro steps per step are possible. As a digital-to-analog converter produces unwanted Ohmic heat in the controller, pulse-width modulation is used instead to regulate the mean current. Simpler models switch voltage only for doing a step, thus needing an extra current limiter: for every step, they switch a single cable to the motor. Bipolar controllers can switch between supply.

voltage, ground, and unconnected. Unipolar controllers can only connect or disconnect a cable, because the voltage is already hard wired. Unipolar controllers need center-tapped windings. It is possible to drive Unipolar stepper motors with bipolar drivers. The idea is to connect the output pins of the driver to 4 transistors. The transistor must be grounded at the emitter and the driver pin must be connected to the base. Collector is connected to the coil wire of the motor.

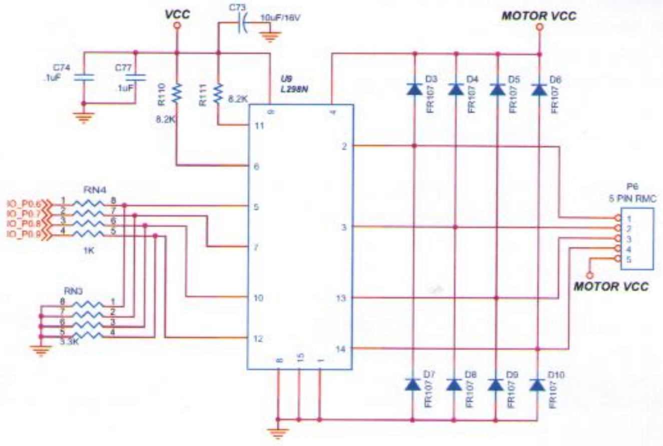
The torque they produce rates stepper motors. Synchronous electric motors using soft magnetic materials (having a core) have the ability to provide position-holding torque (called detent torque, and sometimes included in the specifications) while not driven electrically. To achieve full rated torque, the coils in a stepper motor must reach their full rated current during each step. The voltage rating (if there is one) is almost meaningless. The motors also suffer from EMF, which means that once the coil is turned off it starts to generate current because the motor is still rotating. Their needs to be an explicit way to handle this extra current in a circuit otherwise it can cause damage and affect performance of the motor.



Jumper Position

Closed 1 and 2	-	Internal voltage for stepper motor.
Closed 2 and 3	-	External voltage for stepper motor.

In ViARM-2378 Development board consists of stepper motor driver, which is shown in below.



Used Port Lines

Coil 1	:	P0.6
Coil 2	:	P0.7
Coil 3	:	P0.8
Coil 4	:	P0.9

While we using the Stepper motor close Jumpers: J12-J15 as upward direction.

12. IrDA

Infrared Data Association, a group of device manufacturers that developed a standard for transmitting data via infrared light waves. Increasingly, computers and other devices (such as printers) come with IrDA ports. This enables you to transfer data from one device to another without any cables. For example, if both your laptop computer and printer have IrDA ports, you can simply put your computer in front of the printer and output a document, without needing to connect the two with a cable.

CHAPTER – 5
SOFTWARE EXAMPLE

Example – 1 Program For 8 - Bit LED Interface

```
#include <iolpc2378.h>
#include "irq.h"
#include "config.h"

void delay()
{
 unsigned int i,j;
 for(i=0;i<0x3fff;i++)
 for(j=0;j<0xff;j++);
}
int main (void)
{
 unsigned int Fdiv;
 TargetResetInit();

   PINSEL0 = 0x00000050;                /* RxD0 and TxD0 */
   U0LCR = 0x83;                        /* 8 bits, no Parity, 1 Stop bit */
   Fdiv = ( Fpclk / 16 ) / 19200 ;      /*baud rate */
   U0DLM = Fdiv / 256;
   U0DLL = Fdiv % 256;
   U0LCR = 0x03;                        /* DLAB = 0 */
   U0FCR = 0x07;                        /* Enable and reset TX and RX
    FIFO.
   FIO3DIR = 0X008000FF;

   while(1)
   {
   FIO3PIN = 0X000000ff;
   delay();
   FIO3PIN = 0X00000000;
   delay();
   }

}
```

0x

Example – 2 Program For 8-Way DIP Switch Interface

```
#include <iolpc2378.h>
#include "irq.h"
#include "config.h"

unsigned int k;
unsigned int dat[] = {0x00,0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};
void delay()
{
 unsigned int i,j;
 for(i=0;i<0x1Fff;i++)
 for(j=0;j<0xff;j++);
}
int main (void)
{
 TargetResetInit();
 FIO4PIN = 0X0;
 FIO4DIR = 0XFFFF0000;
 FIO3DIR = 0X008000FF;
 while(1)
 {
  FIO3PIN = FIO4PIN ;
 }

}
```

2
1
0

1
49
1

0x10

Example – 3 Program For 4 x 4 Matrix Keypad Interface

```
#include <iolpc2378.h>
#include "irq.h"
#include "config.h"

unsigned int k;
unsigned int Read_Key;
unsigned char scan [] = {
0x00000E00,0x00000D00,0x00000B00,0x00000700};
unsigned int i;
void delay()
{
 unsigned int i,j;
 for(i=0;i<0xff;i++)
 for(j=0;j<0xff;j++);
}
```

```
void send_serial_data(unsigned char serial)
{

   while((U0LSR & 0x20)==0);
  U0THR = serial;
 }

int main (void)
{
   unsigned int Fdiv;
   TargetResetInit();

   PINSEL0 = 0x00000050;              /* RxD0 and TxD0 */
   U0LCR = 0x83;                      /* 8bits, no Parity, 1 Stop bit */
   Fdiv = ( Fpclk / 16 ) / 19200 ;    /*baud rate */
   U0DLM = Fdiv / 256;
   U0DLL = Fdiv % 256;
   U0LCR = 0x03;                      /* DLAB = 0 */
   U0FCR = 0x07;                      /* Enable and reset TX and RX FIFO. */
   FIO4DIR = 0X00000FFF;
   FIO3DIR = 0X008000FF;

while(1)
{

   FIO4SET = 0X00000e00;        /*First Row*/
   Read_Key = FIO4PIN;
   Read_Key = (Read_Key & 0xf000) >> 12 ;

  if((Read_Key==0x07))
   {
   send_serial_data('0');
   FIO3PIN = 0X00000000;
   }
  if((Read_Key==0x0b))
   {
   send_serial_data('1');
   FIO3PIN = 0X00000001;
   }
 if( (Read_Key==0x0d))
  {
  send_serial_data('2');
  FIO3PIN = 0X00000002;
  }
```

```
if((Read_Key==0x0e))
 {
 send_serial_data('3');
 FIO3PIN = 0X00000003;
 }

FIO4CLR = 0X00000e00;
delay();
FIO4SET = 0X00000d00;                /*Second Row*/
Read_Key = FIO4PIN;
Read_Key = (Read_Key & 0xf000) >> 12 ;

 if((Read_Key==0x07))
  {
  send_serial_data('4');
  FIO3PIN = 0X00000004;
  }
 if((Read_Key==0x0b))
  {
  send_serial_data('5');
  FIO3PIN = 0X00000005;
  }
 if( (Read_Key==0x0d))
  {
  send_serial_data('6');
  FIO3PIN = 0X00000006;
  }
 if( (Read_Key==0x0e))
  {
  send_serial_data('7');
  FIO3PIN = 0X00000007;
  }
FIO4CLR = 0X00000d00;
delay();

FIO4SET = 0X00000b00;                /*Third Row*/
Read_Key = FIO4PIN;
Read_Key = (Read_Key & 0xf000) >> 12 ;

 if((Read_Key==0x07))
   {
   send_serial_data('8');
   FIO3PIN = 0X00000008;
   }
```

```
if((Read_Key==0x0b))
 {
  send_serial_data('9');
  FIO3PIN = 0X00000009;
 }
if( (Read_Key==0x0d))
 {
  send_serial_data('a');
  FIO3PIN = 0X0000000a;
 }
 if( (Read_Key==0x0e))
  {
  send_serial_data('b');
  FIO3PIN = 0X0000000b;
  }
 FIO4CLR = 0X00000b00;
 delay();

FIO4SET = 0X00000700;                    /*Fourth Row*/
Read_Key = FIO4PIN;
Read_Key = (Read_Key & 0xf000) >> 12 ;

    if((Read_Key==0x07))
     {
     send_serial_data('c');
     FIO3PIN = 0X0000000c;
      }
    if((Read_Key==0x0b))
      {
      send_serial_data('d');
      FIO3PIN = 0X0000000d;
      }
    if((Read_Key==0x0d))
      {
      send_serial_data('e');
      FIO3PIN = 0X0000000e;
      }
     if( (Read_Key==0x0e))
      {
      send_serial_data('f');
      FIO3PIN = 0X0000000f;
      }
     FIO4CLR = 0X00000700;
     delay();
  }
 return 0;
}
```

Example – 4 Program For Serial Interface

```
#include<iolpc2378.h>
#include "type.h"
#include "target.h"
#include "irq.h"
#include "uart.h"

#include <intrinsics.h>
unsigned int ADC_VALUE=0;
#define ADC_CLK     1000000        /* set to 1Mhz */
#include "config.h"

void  PLL_Init (void)
{
 unsigned int  m;
 unsigned int  n;
 unsigned int  clk_div;
 unsigned int  clk_div_usb;

  m        = 24;                 /PLL Multiplier = 20, MSEL bits = 12 - 1=11 */
  n        = 1;                  /* PLL Divider =  1, NSEL bits =  1 - 1 =  0   */
  clk_div  = 4;                  /*Configure the ARM Core clock divCCLKSEL=
                                 6 -1 */
  clk_div_usb =                  /* Configure the USB clock divider to 6, SBSEL
                                 = 6 - 1   */

 if ((PLLSTAT & 0x02000000) > 0)
 {
  PLLCON  &= ~0x02;              /* Disconnect the PLL  */
  PLLFEED  =  0xAA;              /* PLL register update sequence, 0xAA, 0x55 */
  PLLFEED  =  0x55;
 }
 PLLCON   &= ~0x01;              /* Disable the PLL    */
 PLLFEED   =  0xAA;              /* PLL register update sequence, 0xAA, 0x55  */
 PLLFEED   =  0x55;
 SCS     &= ~0x10;               /* OSCRANGE = 0, Main OSC is between 1 and
                                  20 Mhz */
 SCS     |=  0x20;               /* OSCEN = 1, Enable the main oscillator */

 while ((SCS &  0x40) == 0)

 CLKSRCSEL = 0x01;               /* Select main OSC, 12MHz, as the PLL clock
                                  source */
 PLLCFG   = (m << 0)
          | (n << 16);         /* Configure the PLL multiplier and divider */
 //PLLCFG   = 11;              /* Configure the PLL multiplier and divider*/
 PLLFEED  = 0xAA;              /* PLL register update sequence, 0xAA, 0x55  */
 PLLFEED  = 0x55;
 PLLCON  |= 0x01;              /* Enable the PLL */
 PLLFEED  = 0xAA;              /* PLL register update sequence, 0xAA, 0x55  */
 PLLFEED  = 0x55;
 CCLKCFG  = clk_div;           /*Configure the ARM Core Processor clock
                               divider   */
 USBCLKCFG = clk_div_usb; /* Configure the USB clock divider */

 while ((PLLSTAT & 0x04000000) == 0)
 PCLKSEL0  = 0xAAAAAAAA;/*Set peripheral clocks to be half of main clock
*/
 PCLKSEL1  = 0x22AAA8AA;
 PLLCON   |= 0x02;              /* Connect the PLL. The PLL is now the active
                                clock source  */
 PLLFEED  = 0xAA;              /* PLL register update sequence, 0xAA, 0x55  */
 PLLFEED  = 0x55;
 while ((PLLSTAT & 0x02000000) == 0)

 PCLKSEL0 = 0x55555555;  /* PCLK is the same as CCLK */
 PCLKSEL1 = 0x55555555;
}
void send_serial_data(unsigned char serial)
{

 while((U0LSR & 0x20)==0);
  U0THR = serial;
}

void adc_serial_tx(unsigned int ch)
{

   unsigned int t1000,t100,t10,t1,temp;
   t1000 = ch   / 1000;
   temp  = ch   % 1000;
   t100  = temp / 100;
   temp  = temp % 100;
   t10   = temp / 10;
   t1    = temp % 10;
   send_serial_data(t1000+0x30);
   send_serial_data(t100 +0x30);
```

```
    send_serial_data(t10+0x30);
    send_serial_data(t1+0x30);
    send_serial_data(0x0d);
    send_serial_data(0x0a);

  }

    int putchar(int ch)
  {
   if (ch == '\n')
   {
    while (!(U0LSR & 0x20));
    U0THR = 0x0d;
   }
   while (!(U0LSR & 0x20));
   return (U0THR = ch);
  }

void main()
{
    unsigned long int val;
    unsigned int Fdiv;
    PLL_Init();

    PCONP |= (1 << 12);
    PINSEL1 = 0X00054000;          //AD0.0 AS ANALALOG INPUT
    PINSEL0 = 0x00000050;          // RxD0 and TxD0

    U0LCR = 0x83;                  // 8 bits, no Parity, 1 Stop bit
    Fdiv = (60000000 / 16 ) / 19200 ; //baud rate
    U0DLM = Fdiv / 256;
    U0DLL = Fdiv % 256;
    U0LCR = 0x03;                  // DLAB = 0
    U0FCR = 0x07;                  // Enable and reset TX and RX FIFO.
  send_serial_data('a');
 install_irq( UART0_INT, (void *)UART0Handler, HIGHEST_PRIORITY ) ;
  while(1)
  {
   send_serial_data('a');
   printf("\n\r Welcome");
  }

    }
```

Example – 5 Program for SPDT Relay Interface

```
#include <iolpc2378.h>
#include "irq.h"
#include "config.h"

void delay()
{
 unsigned int i,j;
 for(i=0;i<0x2Fff;i++)
   for(j=0;j<0xff;j++);
}
int main (void)
{
 TargetResetInit();

  FIO3PIN = 0X0;
  FIO3DIR = 0XFFFFFFFF;

  while(1)
  {
   FIO3SET = 0x05000000;
   delay();
   FIO3CLR = 0x05000000;
   delay();

  }
 }
```

Example – 6 Program for Stepper Motor Interface

```
#include <iolpc2378.h>
#include "irq.h"
#include "config.h"

void delay()
{
 unsigned int i,j;
 for(i=0;i<0xff;i++)
   for(j=0;j<0xff;j++);
}
int main (void)
{
 TargetResetInit();
```

```
  IO0DIR  = 0XFFFFFFFF;

  while(1)
  {
   for(k=0;k<10;k++)
   {
    IO0SET = 0X00000240;
    delay();
    IO0CLR = 0X00000240;
    IO0SET = 0X00000140;
    delay();
    IO0CLR = 0X00000140;
    IO0SET = 0X00000180;

    delay();
    IO0CLR = 0X00000180;
    IO0SET = 0X00000280;
    delay();
    IO0CLR = 0X00000280;
   }
  }
}
```

Example – 8 Program for ADC Interface

```
#include<iolpc2378.h>
#include "irq.h"
#include "config.h"

unsigned int ADC_VALUE=0;
#define ADC_CLK     1000000                          set to 1Mhz */
void send_serial_data(unsigned char serial)

{
 while((U0LSR & 0x20)==0);
    U0THR = serial;
}

void adc_serial_tx(unsigned int ch)
{
   unsigned int t1000,t100,t10,t1,temp;
   t1000 = ch   / 1000;
   temp  = ch   % 1000;
   t100  = temp / 100;
   temp  = temp % 100;
   t10   = temp / 10;
   t1    = temp % 10;

   send_serial_data(t1000+0x30);
   send_serial_data(t100 +0x30);
   send_serial_data(t10+0x30);
   send_serial_data(t1+0x30);
   send_serial_data(0x0d);
   send_serial_data(0x0a);
}
void main()
{
    unsigned long int val;
    unsigned int Fdiv;
    TargetResetInit();
    PCONP |= (1 << 12);
    PINSEL1 = 0X00054000;          //AD0.0 AS ANALALOG INPUT
    PINSEL0 = 0x00000050;          // RxD0 and TxD0
    U0LCR = 0x83;                  // 8 bits, no Parity, 1 Stop bit
    Fdiv = ( Fpclk / 16 ) / 19200 ;  // baud rate
    U0DLM = Fdiv / 256;
    U0DLL = Fdiv % 256;
    U0LCR = 0x03;                  // DLAB = 0
    U0FCR = 0x07;                  // Enable and reset TX and RX FIFO.

    AD0CR = ( 0x01 << 0 ) | /*SEL=1,select channel 0~7 on ADC0 */
      ( ( Fpclk / ADC_CLK - 1 ) << 8 ) |  /* CLKDIV = Fpclk / 1000000 - 1 */
      ( 1 << 16 ) |                /* BURST = 0, no BURST, software controlled */
      ( 0 << 17 ) |                /* CLKS = 0, 11 clocks/10 bits */
      ( 1 << 21 ) |                /* PDN = 1, normal operation */
      ( 0 << 22 ) |                /* TEST1:0 = 00 */
      ( 1 << 24 ) |                 /* START = 0 A/D conversion stops */
      ( 0 << 27 );                 /* EDGE = 0 (CAP/MAT singal falling,trigger A/D
                                   conversion) */

      while(1)
    {
      while((AD0GDR & 0X80000000)!=0X80000000);
             val = (AD0GDR>>6)& 0x3ff
       adc_serial_tx(val);
    }
}
```

Example – 9 Program DAC Interface

```
#include <iolpc2378.h>
#include "dac.h"

void DACInit( void )
{
  PINSEL1 = 0x00200000;         /* set p0.26 to DAC output */
  return;
}
```