3. FLASHING OF LEDS USING ARM PROCESSOR

```
#include <LPC214x.H>                    /* LPC214x definitions */
void wait (void)                        /* wait function */
{
 int  d;
 for (d = 0; d < 1000000; d++);         /* only to delay for LED flashes */
}
int main (void)
 {
 IODIR0 = 0x80002000;                   /* P0.13 and P0.31 defined as Outputs */
 while (1)                              /* Loop forever */
  {
        IOCLR0 = 0x80002000;            /*Active Low outputs makes the LEDs ON*/
   wait ();

        IOSET0 = 0x80002000;                    /* High outputs makes the LEDs OFF*/
        wait ();
  }
 }
```

4. INTERFACING DIGITAL TO ANALOG CONVERTER WITH ARM PROCESSOR

```
#include<LPC214X.H>
#define DAC_BIAS 0x00010000         // DAC_BIAS for speed variation 5th hex may be 0 or 1
//#define DIGITAL_VAL 1001011000     //10 bit digital value(0000000000->0v ac to 11111111111-
>3.3v ac)
#define DIGITAL_VAL 1023 //in decimal 0-1023

void wait_long (void)
{                   /* wait function */
 int  d;
 for (d = 0; d < 1000000; d++);         /* only to delay */
}

int main()
 {
        wait_long();
        wait_long();
        //IODIR0 = 0X00000FFF;
        //IODIR1 = 0XFFFF0000;
        //IOSET0 = 0XFFFFFFFF;
        //IOCLR1 = 0XFFFF0000;
   PINSEL1 |= 0x00080000;  //Enable pin 0.25   as DAC
//      DACR = 0X00017FC0;            // 000 = 0V(min),7FC = 1.6V,7FF = 3.3V(max)
        DACR = (DIGITAL_VAL<<6)|DAC_BIAS;//32 bit resister(15 to 6th bit resister value,16th
bit for speed)
        while(1);
}
```

5. INTERFACING LED AND PWM AND TO VERIFY THE OUTPUT IN THE ARM7

```
/* Place lcd.c file into following directories C:\Keil\ARM\INC\Philips.********/
/* This progrm is used to Generate the PWM.You can change the Freq and DutyCyc*/
/* If you want. ************************************************************/

/*****************************************************************************
SM MICRRO SYSTEMS
DEVELOPED BY SIVAKUMAR.V DATE:08-07-2016 TIME:12:37
Duty Cycle as a Ratio:
Duty Cycle =     T-ON/T-ON + T-OFF

DutyCycle in Percentage is :
Duty Cycle % =            T-ON/(T-ON + T-OFF)*100;

Vaverage = DutyCycle x VH
In case the Low State Represents a Negative Voltage then the above equation can be generalized as
follows :
Vaverage = (DutyCycle x VH) + ((1-D) x VL)
Where , VH = Voltage for High State & VL = Voltage for Low State

Delay=clock cycles/frequency(in MHz)

*****************************************************************************/
#include <lpc214x.h>
#define PLOCK 0x00000400
#define PWMPRESCALE 60   //60 PCLK cycles to increment TC by 1 i.e 1 Micro-second

void initPWM(void);

void initClocks(void);
void setupPLL0(void);
void feedSeq(void);
void connectPLL0(void);

int main(void)
{
        initClocks(); //Initialize CPU and Peripheral Clocks @ 60Mhz
  initPWM(); //Initialize PWM

  //IO0DIR = 0x1; This is not needed!
        //Also by default all pins are configured as Inputs after MCU Reset.

    while(1)
    {
                    }

}

void initPWM(void)
{
```

```
PINSEL1 |= 0x00000400;  //Enable pin0.21   as PWM5
PWMPR = 60-1; // 1 micro-second resolution
PWMPCR = 0x00002000;   //PWM channel single edge control, output enabled PWM5 ENABLE
PWMMCR = (1<<1); // Reset PWMTC on PWMMR0 match
PWMMR0 = 10000; // 10ms for 1 period duration
PWMMR5 = 2500; // 2.5ms - pulse duration i.e width (Brigtness level) ON time
PWMLER = 0x00000021; //enable shadow latch for match 0 - 2
PWMTCR = 0x00000002; //Reset counter and prescaler
PWMTCR = 0x00000009; //enable counter and PWM, release counter from reset
}

void initClocks(void)
{
   setupPLL0();
   feedSeq(); //sequence for locking PLL to desired freq.
   connectPLL0();
   feedSeq(); //sequence for connecting the PLL as system clock
   //SysClock is now ticking @ 60Mhz!
   VPBDIV = 0x01; // PCLK is same as CCLK i.e 60Mhz
  //Using PLL settings as shown in : http://www.ocfreaks.com/lpc214x-pll-tutorial-for-cpu-and-
peripheral-clock/
   //PLL0 Now configured!
}


//---------PLL Related Functions :----------------

void setupPLL0(void)
{
   //Note : Assuming 12Mhz Xtal is connected to LPC2148.

   PLL0CON = 0x01; // PPLE=1 & PPLC=0 so it will be enabled
               // but not connected after FEED sequence
   PLL0CFG = 0x24; // set the multipler to 5 (i.e actually 4)
                        //So for our calculation M = 5 and P = 2, then PLLCFG = 0b001 00100 = 0x24;
               // i.e 12x5 = 60 Mhz (M - 1 = 4)!!!
               // Set P=2 since we want FCCO in range!!!
               // So , Assign PSEL =01 in PLL0CFG as per the table.
}

void feedSeq(void)
{
   PLL0FEED = 0xAA;
   PLL0FEED = 0x55;
}

void connectPLL0(void)
{
   // check whether PLL has locked on to the  desired freq by reading the lock bit
   // in the PPL0STAT register
```

```
   while( !( PLL0STAT & PLOCK ));

   // now enable(again) and connect
   PLL0CON = 0x03;// PPLE=1 & PPLC=1 so it will be enabled and connected;
         }
```

6. INTERFACING REAL TIME CLOCK WITH ARM PROCESSOR

```
/* Place lcd.c file into following directories C:\Keil\ARM\INC\Philips.******/
/* This progrm is used to interface the RTC.You can change the date and time*/
/* If you want. This Program can both Read and write data into RTC.RTC has a*/
/* Battery backup for continous Running. ***********************************/
/*-----SM MICRRO SYSTEM-------*/
/*DEVELOPED BY SIVAKUMAR.V
pclk = 30,000,000 Hz
PREINT = (int)(pclk/32768)-1
PREFRAC = pclk - ((PREINT+1) x 32768)
*/

#include<LPC214X.H>
#include"lcd.h"
#include "mat_7seg.h"

//global variable declaration
unsigned int mnt,dt,dtyr;
unsigned int hrs_p,min_p,sec_p,i,j;
unsigned int key;
char En_alarm_enter,alarm_enter_mode,char_count;
char hrs[2];
char mins[2];
char secs[2];
//char flag=0;
void set_dt(void);
void delay(unsigned int x)
{
         int i;
         while(x--)
         {
                  for(i=0;i<=2000;i++);
         }
}


unsigned char flag=0;


void rtc_int(void)__irq{

ILR = 0X01;

flag = 1;
```

```
VICVectAddr = 0X00000000;

}


void init_rtc(){

ILR = 0X01;

CCR  = 0X13;

CCR =  0X11;

CIIR = 0X01;

VICIntEnable = 0x00002000;

VICVectCntl0 = 0x0000002D;

VICVectAddr0 = (unsigned)rtc_int;

}


int main()
{
  wait();
            wait();
            wait();
            wait();
            lcdinit();
           init_rtc();
            init_Matrix_7seg();   // Initialize matrix keyboard and 7segment dispaly
            clrscr(2);
            printstr("SM MICRRO SYSTEM",0,0);
            printstr("   ARM DEV KIT  ",0,1);
  clrscr(3);
           printstr("   RTC CLOCK  ",0,1);
           lcdcmd(0x01); // clear screen
           printstr("SET TIME KEY1",0,0);//press sw3 to set period
           printstr("SET YEAR KEY2",0,1);//press sw4 to set time
           delay(2000);
           printstr("                  ",0,1);
           printstr("                  ",0,1);

           while(1)
           {

                    dt = DOM;
```

```
            mnt = MONTH;
            dtyr = DOY;

            hrs_p = HOUR;
      min_p = MIN;
      sec_p = SEC;

             key = catch_key();

    if(key != 0)
      {
            if(key==13)//press sw1 (key13)for setting sec,min,hour if need
            {

                  clrscr(3);
                  printstr("SET TIME",0,1);//press sw3 to set period
                  delay(10);
                  set_dt();
                        HOUR  = (hrs[0]*10)+hrs[1];
                    MIN= (mins[0]*10)+mins[1];
                    SEC= (secs[0]*10)+secs[1];

                    hrs[0]=hrs[1]=mins[0]=mins[1]=secs[0]=secs[1]=0;
            }//after period set press sw4(key16)for setting date
            if(key==14)//press sw2(key14) for setting date,year,month if need
            {
                  clrscr(3);
                  printstr("SET DATE",0,1);
                  delay(10);
                  set_dt();
               DOM = (hrs[0]*10)+hrs[1];
                        MONTH= (mins[0]*10)+mins[1];
                        DOY= (secs[0]*10)+secs[1];
hrs[0]=hrs[1]=mins[0]=mins[1]=secs[0]=secs[1]=0;
            }//after time set press sw4(key16)for setting date
    }

            gotoxy(0,0);
            printstr("DATE=",0,0);

            split_numbers(dt);
      lcddat(tens+0x30);
      lcddat(ones+0x30);
      lcddat(':');
      split_numbers(mnt);
      lcddat(tens+0x30);
      lcddat(ones+0x30);
      lcddat(':');
      split_numbers(dtyr);
      lcddat(tens+0x30);
      lcddat(ones+0x30);
```

```
        gotoxy(0,1);

         printstr("TIME=",0,1);
         split_numbers(hrs_p);
         lcddat(tens+0x30);
         lcddat(ones+0x30);
         lcddat(':');
         split_numbers(min_p);
         lcddat(tens+0x30);
         lcddat(ones+0x30);
         lcddat(':');
         split_numbers(sec_p);
         lcddat(tens+0x30);
         lcddat(ones+0x30);
        }
}

        void set_dt(void)
        {
         key = catch_key();
         if(key != 0)
          {
           if(key == 13 || key == 14)
                En_alarm_enter = 1;
              else
                En_alarm_enter = 0;

              if(En_alarm_enter)
                {
                        char_count = 0;
                        alarm_enter_mode =1;
                        printstr("                ",0,0);
                        lcdcmd(0x80); // start of 1st line
                        while(alarm_enter_mode)
                         {
                                key = catch_key();
                                if(key!=0)
                                  {
                                        if(key<=10 || key == 16)
                                         {
                                          char_count++;
                                                if(key == 16)
                                                  char_count = 7;

                                          switch(char_count)
                                          {
                                           case 1: hrs[0] = key-1;
                                                        lcddat(hrs[0]+0x30);
                                                                break;
                                           case 2: hrs[1] = key-1;
```

```
                                        lcddat(hrs[1]+0x30);
                                                lcddat(':');
                                                break;
                                    case 3: mins[0]= key-1;
                                                lcddat(mins[0]+0x30);
                                                break;
                                    case 4:         mins[1]= key-1;
                                                  lcddat(mins[1]+0x30);
                                                lcddat(':');
                                                break;
                                          case 5: secs[0]= key-1;
                                                lcddat(secs[0]+0x30);
                                                flag=1;
                                                break;
                                          case 6: secs[1]= key-1;
                                                lcddat(secs[1]+0x30);
                                                break;
                                          default: if(key == 16)
                                                  alarm_enter_mode =0;

                                                }
                                          }
                                         }
                                    for(i=0;i<20;i++)
                                      for(j=0;j<65000;j++);
                                    }

                                    }

          }

        }
```

7. INTERFACING KEYBOARD AND LCD

```
#include <LPC214x.h>
#define RS     0x00000400  /* P0.10 */
#define CE     0x00001800  /* P1.11 */

void clrscr(char ch);
void lcdinit(void);
void lcdcmd(char);
void lcddat(char);
void gotoxy(char,char); //x,y ; x-char position(0 - 16) y-line number 0 or 1
void printstr(char *,char,char);    //string,column(x),line(y)
void wait (void);
void split_numbers(unsigned int number);


#define SET 1
#define OFF 0
```

```
unsigned int thousands,hundreds,tens,ones;

void wait (void)  {                     /* wait function */
  int  d;
  for (d = 0; d < 100000; d++);          /* only to delay for LED flashes */
}


void lcdinit()
{
   IODIR0 |= 0x0000FFFF;
            IOCLR0 |= 0X00000FFF;
            lcdcmd(0x28);
   lcdcmd(0x28);
   lcdcmd(0x0c);
   lcdcmd(0x06);
   lcdcmd(0x01);
   lcdcmd(0x0f);
   wait();
}


void gotoxy(char x, char y)
{
   if(y == 0)
      lcdcmd(0x80+x);
   else
      lcdcmd(0xc0+x);

}


void printstr(char *str, char x, char y)
{
   char i;
   gotoxy(x,y);
   wait();//(500);
   for(i=0;str[i]!='\0';i++)
      lcddat(str[i]);

}
void lcdcmd(char cmd)
{
unsigned char LCDDAT;
            LCDDAT = (cmd & 0xf0);       //higher nibble
            IOSET0 = LCDDAT;
    IOCLR0 = RS;
            IOSET0 = CE;
    wait();//(100);                        //enable lcd
            IOCLR0 = CE;
```

```
        IOCLR0 = 0X00000FFF;


    LCDDAT = ((cmd<<0x04) & 0xf0);          //lower nibble
        IOSET0 = LCDDAT;
    IOCLR0 = RS;
        IOSET0 = CE;
    wait();//(100);                         //enable lcd
        IOCLR0 = CE;
        IOCLR0 = 0X00000FFF;

}

void lcddat(char cmd)
{
  unsigned char LCDDAT;
        LCDDAT = (cmd & 0xf0);      //higher nibble
        IOSET0 = LCDDAT;
    IOSET0 = RS;
        IOSET0 = CE;
    wait();//(100);                         //enable lcd
        IOCLR0 = CE;
        IOCLR0 = 0X00000FFF;

    LCDDAT = ((cmd<<0x04) & 0xf0);          //lower nibble
        IOSET0 = LCDDAT;
    IOSET0 = RS;
        IOSET0 = CE;
    wait();//(100);                         //enable lcd
        IOCLR0 = CE;
        IOCLR0 = 0X00000FFF;
 }

void clrscr(char ch)
{
   if(ch==0)
   {
      printstr("                ",0,0);
      gotoxy(0,0);
   }
   else if(ch == 1)
   {
      printstr("                ",0,1);
      gotoxy(0,1);
   }
   else
   {
      lcdcmd(0x01);
     // delay(100);
   }
}
```

```
void split_numbers(unsigned int number)
{
  thousands = (number /1000);
  number %= 1000;
  hundreds = (number / 100);
  number %= 100;
  tens = (number / 10);
  number %= 10;
  ones = number ;
}

void Wait_Msg(void)
{
 lcdcmd(0x01);
 printstr("   WELCOME TO   ", 0, 0);
 printstr("SM MICRRO SYSTEM", 0, 1);

}
void Welcome_Msg(void)
{
 lcdcmd(0x01);
 printstr("  ARM-7 LPC2148 ", 0, 0);
 printstr("32-Bitcontroller", 0, 1);
}
```

8. INTERRUPT PERFORMANCE CHARACTERISTICS OF ARM AND FPGA

```
/*****************************************************************************
***/
/* FILE       : interrupt_buzzer.c                                           */
/* AUTHOR     : Sivakumar.V, SM Micrro System, Tamabaram, Chennai            */
/* DESCRIPTION: This file is part of example projects given with SM Micrro system's */
/* ARM LPC2148 development Board. The example projects should be used only for      */
/* educational purpose and not for product development                        */
/*****************************************************************************
***/
/* This is a test program to make the interrupt signal for ON buzzer in the ARM LPC2148
          */
/* development board itself                                                   */
/*****************************************************************************
***/


/**********************START PROGRAM*****************************/
#include <LPC214x.h>

 /**Prototypes**/
 void init_VIC(void);
 void init_Interrupt(void);
 void init_ports(void);
```

```
 void wait_for_turnoffRelay(void);
 void delay(int count);

void init_VIC(void)
  {
            /* initialize VIC*/
    VICIntEnClr  = 0x00010000;   //VICIntEnable expect writing a 1 here will disabled External
interrupt2 (EINT2)
    VICVectAddr  = 0;          //no address of ISR
    VICIntSelect = 0;         //Writing a 0 interrupt as IRQ and writing a 1 will make it FIQ.
  }

 void ExtInt_ISR(void)__irq   // Interrupt Service Routine-ISR
 {

           IOCLR1 |= 0x00040000;           // Turn ON Buzzer
   delay(100000);
           delay(100000);
  IOSET1 |=  0x00040000; // Turn OFF Buzzer
          EXTINT = (1<<2);     /*(External Interrupt Flag Register)external interrupt occurs.*/
 VICVectAddr = 0;                  /* Acknowledge Interrupt */
 }
 void init_Interrupt(void) // initialize the external interrupt
    {
            PINSEL0 = 0x80000000;          // select P0.15 for EINT2 Vectored Interrupt Controller(VIC)
            VICIntEnable = (1 << 16);   // External interrupt2 (EINT2) table
            VICVectCntl0 = (1<<5)|(16);  // set the VIC control reg for EINT2
            VICVectAddr0 = (unsigned long)ExtInt_ISR; // address of the ISR
            EXTMODE &= ~(1<<2);   // set VIC for level-sensitive for EINT2(External Interrupt Mode
Register)
           }

           int main()  /*press sw10 single big switch for making interrupt*/
 {
   init_VIC();
   init_Interrupt();
           IODIR0 = 0x80002000;                 /* P0.13 and P0.31 defined as Outputs */
           IODIR1 = 0x00040000;              /* p1.16 BUZZER direction output*/
           IOSET1 = 0x00040000;              /* p1.16 BUZZER OFF*/
     while(1)
             {
              IOCLR0 = 0x80002000;                                    /*Active Low outputs makes the
LEDs ON*/
       delay(100000);
                     IOSET0 = 0x80002000;                                     /* High outputs makes the
LEDs OFF*/
               delay(100000);
             }
 }

void delay(int count)
```

```
{
 int j=0,i=0;

 for(j=0;j<count;j++)
 {
  /* At 60Mhz, the below loop introduces
  delay of 10 us */
  for(i=0;i<35;i++);
 }
}
/**********************END PROGRAM*****************************/
```

9. INTERFACING STEPPER MOTOR WITH ARM PROCESSOR

```
/*****************************************************************************
***/
/* FILE       : main_LCD_Test.c                                              */
/* AUTHOR     : Rajasekaran.K, SM Micrro System, Tamabaram, Chennai          */
/* DESCRIPTION: This file is part of example projects given with SM Micrro system's */
/* ARM LPC2148 development Board. The example projects should be used only for      */
/* educational purpose and not for product development                        */
/*****************************************************************************
***/
/* This is a test program to display strings in LCD module in the ARM LPC2148     */
/* development board itself                                                   */
/*****************************************************************************
***/
#include <LPC214x.H>     /* LPC214x definitions */

#define step1     0x00010000  /* P1.16 */
#define step2     0x00020000  /* P1.17 */


void wait (void)
{                    /* wait function */
 int  d;
 for (d = 0; d < 10000; d++);          /* only to delay for LED flashes */
}

void call_stepper_forw()
{
 IOCLR1 = 0X00FF0000;
 IOSET1 = 0X00040000;
 wait();
 wait();
// wait();
 //wait();
 IOCLR1 = 0X00FF0000;
 IOSET1 = 0X00060000;
 wait();
```

```
 wait();
// wait();
// wait();
 IOCLR1 = 0X00FF0000;
 IOSET1 = 0X00070000;
 wait();
 wait();
// wait();
// wait();
 IOCLR1 = 0X00FF0000;
 IOSET1 = 0X00050000;
 wait();
 wait();
// wait();
// wait();
}

/*void call_reverse(void)
{
 IOCLR1 = 0X00FF0000;
 IOSET1 = 0X00050000;
 wait();
 wait();
// wait();
 //wait();
 IOCLR1 = 0X00FF0000;
 IOSET1 = 0X00070000;
 wait();
// wait();
 //wait();
 //wait();
 IOCLR1 = 0X00FF0000;
 IOSET1 = 0X00060000;
 wait();
// wait();
 //wait();
 //wait();
 IOCLR1 = 0X00FF0000;
 IOSET1 = 0X00040000;
 wait();
// wait();
 //wait();
 //wait();
} */



int main (void)
 {
  IODIR1 |= 0xFFFFFFFF;
```

```
            IOCLR1 |= 0X00FF0000;
            wait();
  while(1)                        /*Loop Forever*/
    {
              call_stepper_forw();
    //call_reverse();
              wait();
            //  wait();
//              wait();
             // wait();
              IOCLR1 = 0X00FF0000;
            }
 }
```