ภาคผนวก

ภาคผนวก

ก

โปรแกรมไมโครคอนโทรลเลอร์

```
////////////////////////////////////////////////////////////////////////////
/*
  * Project name:
      Radiation Meter
  * File name:
      Mainboard4.c
  * Copyright:
      (c) Taewan Pleansaithong, 2010.
  * Revision History:
      05-24-2010: Last update
  * Description:
      This is a software for radiation meter on PIC18F2550 series MCU.
  * Test configuration:
      MCU:           PIC18F2550
      Dev.Board:     Radiation meter board
      Oscillator:    HS, 20.0000 MHz
      Ext. Modules:  LCD 8x2, switch
      SW:            mikroC for PIC v.7.0.0.3
*/
////////////////////////////////////////////////////////////////////////////
#include "F:\Thesis\Microchip\Projects\Mainboard\Mainboard5\USBdsc.c"
unsigned char Read_buffer[4];
unsigned char Write_buffer[4];
unsigned char num, number;
char *text = "Geiger";
```

```
unsigned int countloop;
unsigned int timer_0;
unsigned long timer_0F;
unsigned long full_loop;
unsigned long Pulse;
char outtolcd[4];
unsigned char show_mode;
unsigned char show_time;
unsigned char flag = 0;
unsigned char modeflag = 1;
unsigned short timeflag = 1;
unsigned char timeflag_txt[3];
unsigned char stop_count_flag = 0;
char Timer_txt[6];
char Pulse_txt[12];
char full_loop_txt[12];
char Rem_per_Hour_Txt[13];
char counter_txt[4];
char temp1;
unsigned short i, j, k;
char i_txt[4];
unsigned int CountRate;
unsigned int CPS_count;
unsigned float Roentgen_per_Hour;
unsigned float Rem_per_Hour;
unsigned int         seconds = 0;
unsigned int         minutes = 0;
unsigned int           hours = 0;
unsigned int     tempseconds = 0;
```

```
unsigned int     tempminutes = 0;
unsigned int    time_to_stop = 1;
unsigned int       stop_time = 0;
unsigned char         reload = 0;
unsigned char  updateDisplay = 0;
unsigned short        counter = 0;
char       *CPM_mode = "CPM";
char      *mRem_mode = "mRem";
unsigned char  txt[16];
unsigned short address;
unsigned short data;
unsigned short save_time;
unsigned char address_txt[7];
unsigned short data_read, empty, last_address;
unsigned char data_read_txt[5];
unsigned char empty_txt[5];
#define TIMER_START_VALUE_L 0xA0
#define TIMER_START_VALUE_H 0x15
#define  TIMER_RELOAD_VALUE 200
#define  GM_signal  PORTA.F4
#define   mode_key  PORTA.F3
#define     ok_key  PORTA.F2
#define   time_key  PORTB.F1
#define        LED  PORTC.F7
#define    pressed  0
#define  unpressed  1
////////////////////////////////////////////////////////////////////////////
void USB();
void HID_InterruptProc();
```

```
void Counttorem();
void Convert_for_LCD();
void save();
void SendAddress();
void MemoryReadDataNumber();
//////////////////////////////////////////////////////////////////
// Initial for timer and interrupt
//////////////////////////////////////////////////////////////////
void initialTMR0()
{
  // set timer0 control register
  T0CON.TMR0ON = 0;         // Disables timer0
  T0CON.T08BIT = 0;         // Timer0 is configured as a 16 bit
  T0CON.T0CS   = 1;         // Transition on T0CKI pin
  T0CON.T0SE   = 1;         // Increment on high-to-low transition on T0CKI pin
  T0CON.PSA    = 1;         // Timer0 prescaler in NOT assigned. Timer0 clock input
bypasses prescaler
  T0CON.T0PS2  = 1;
  T0CON.T0PS1  = 1;
  T0CON.T0PS0  = 1;
}
//////////////////////////////////////////////////////////////////////////////////////
void initialTMR1()
{
   // set timer 1 control register
   T1CON.RD16    = 1;
   T1CON.T1RUN   = 1;
   T1CON.T1CKPS1 = 0;
   T1CON.T1CKPS0 = 0;
```

```
  T1CON.T1OSCEN = 0;
  T1CON.T1SYNC  = 1;
  T1CON.TMR1CS  = 0;
  T1CON.TMR1ON  = 0;
}
////////////////////////////////////////////////////////////////
// Prints the current time, from bootup, to the display       //
////////////////////////////////////////////////////////////////
void PrintTimeToDisplay()
{
   // ByteToStr(seconds, txt);
    sprintf(txt, "%2.2d:%2.2d:%2.2d", hours, minutes, seconds);
   // Lcd_Out(1,1, "Geiger");
    Lcd_Out(2,1,txt);
}
////////////////////////////////////////////////////////////////////////////
// Handles microcontroller internal or external interrupts                //
////////////////////////////////////////////////////////////////////////////
void interrupt(void)
{
    if(INTCON.TMR0IF == 1)
    {
      HID_InterruptProc();       // Keep alive
      TMR0L = 100;               // Re-load TMR0L
      INTCON.TMR0IF = 0;         // Re-enable TMR0 interrupts
      // counter = counter +1;
      // INTCON.TMR0IF = 0;
    }
    else
```

```
{
// If the timer1 interrupt flag was set
if (PIR1.TMR1IF)
{
  // Prime the register so that the timer overflows (and thus the
  // interrupt) at the time desired.
  TMR1H = TIMER_START_VALUE_H;
  TMR1L = TIMER_START_VALUE_L;

  // Reset the TIMER1 interrupt flag. (This bit is not set back
  // by the PIC. We have to do this manually).
  PIR1.TMR1IF = 0;

  // when reload reaches 0, we have elapsed one second
  if(reload == 0)
  {
    // Note this timing will contain a margin of error.
    // My testing shows time times as recorded by the PIC
    // and by a Timex multipurpose chronograph on a digital watch.
    // PIC display    Timex
    // 00:03:00       00:03:00
    //
    reload = TIMER_RELOAD_VALUE + 1;

    seconds++;
    tempseconds = seconds;
    if (seconds == 60)
    {
      seconds = 0;
```

```
            minutes++;
            tempminutes = minutes;
            if (minutes == 60)
            {
              minutes = 0;
              hours++;
            }
          }

          // timer interrupt, so set the update display bit
          updateDisplay = 1;
        }
        reload--;
     }
   }
}
////////////////////////////////////////////////////////////////////////////////
// Count for radiation                                                        //
////////////////////////////////////////////////////////////////////////////////
void count()
{
        seconds = 0;
        minutes = 0;
          hours = 0;
         reload = 0;
  updateDisplay = 0;
    tempseconds = 0;
   time_to_stop = 1;
      stop_time = 0;
```

```
TMR0L = 0x00;
TMR0H = 0x00;
TMR1H = TIMER_START_VALUE_H;
TMR1L = TIMER_START_VALUE_L;
reload = TIMER_RELOAD_VALUE;
T1CON.TMR1ON = 1;
T0CON.TMR0ON = 1;
Lcd_Cmd(LCD_CLEAR);
//Lcd_Out(2, 3, "CPM");
GM_signal = 1;
PrintTimeToDisplay();
while (time_to_stop != stop_time)
{
  if(timeflag == 1)
  {
    time_to_stop = tempseconds;
    stop_time = 60;
  }
  else
  {
    if(timeflag == 2)
    {
      time_to_stop = tempminutes;
      stop_time = 5;
    }
    else
    {
      if(timeflag == 3)
      {
```

```
            time_to_stop = 1;
            stop_time = 0;
          }
          else
          {
            goto exit_count;
          }
        }
      }
      Timer_0 = TMR0H;
      Timer_0 = Timer_0 << 8;
      Timer_0 = Timer_0 | TMR0L;
      WordToStr(Timer_0,Timer_txt);
      Lcd_Out(1,3,Timer_txt);          // show number of count to the display
      if (updateDisplay)
      {
        // reset the updateDisplay flag
        updateDisplay = 0;
        // Put the time to the display
        PrintTimeToDisplay();
      }
      //if(ok_key == pressed)
      //{
      //   stop_count_flag = 1;
      //   goto exit_count;
      //}
    }
    exit_count:
    T1CON.TMR1ON = 0;
```

```
  T0CON.TMR0ON = 0;
  //TMR0L = 0x00;
  //TMR0H = 0x00;
  //Lcd_Out(2, 1, "End 1 M ");
}
////////////////////////////////////////////////////////////////////////////
void mRem_Count()
{
   while(ok_key == pressed)
   {
      Lcd_Out(1, 2, "Count");
      Lcd_Out(2, 1, show_mode);
   }
   Lcd_Cmd(LCD_CLEAR);
   while(ok_key == unpressed)
   {
        seconds = 0;
        minutes = 0;
          hours = 0;
         reload = 0;
  updateDisplay = 0;
    tempseconds = 0;
   time_to_stop = 1;
      stop_time = 0;
 Roentgen_per_Hour = 0;
      Rem_per_Hour = 0;
  Rem_per_Hour_Txt[13] = 0;
         Timer_0 = 0;
  TMR0L = 0x00;
```

```
  TMR0H = 0x00;
  TMR1H = TIMER_START_VALUE_H;
  TMR1L = TIMER_START_VALUE_L;
  reload = TIMER_RELOAD_VALUE;
  //Lcd_Cmd(LCD_CLEAR);
  GM_signal = 1;
  T1CON.TMR1ON = 1;     //Enable Timer 1
  T0CON.TMR0ON = 1;     //Enable Timer 0
  while(tempseconds != 1)
  {
     Timer_0 = TMR0H;
     Timer_0 = Timer_0 << 8;
     Timer_0 = Timer_0 | TMR0L;
     WordToStr(Timer_0,Timer_txt);
     //Lcd_Out(1,3,Timer_txt);
  }
  T1CON.TMR1ON = 0;
  T0CON.TMR0ON = 0;
  //Lcd_Out(2, 4, Timer_txt);
  COUNTtoREM();
  }
}
////////////////////////////////////////////////////////////////////////////
// Convert Count per Second to Rem
////////////////////////////////////////////////////////////////////////////
void COUNTtoREM()
{
   CPS_count = Timer_0;
   //CountRate = Timer_0;
```

```
  //CPS_count = CountRate/60;
  Roentgen_per_Hour = (CPS_count*0.002)/30;
  Rem_per_Hour = (Roentgen_per_Hour*11.933174224)/1;
  FloatToStr(Rem_per_Hour,Rem_per_Hour_Txt);
  Convert_for_LCD();
}
////////////////////////////////////////////////////////////////////////////
//  Convert for LCD
////////////////////////////////////////////////////////////////////////////
void Convert_for_LCD()
{
  i = 0;
  ////////////////////////////////////////////////////////
  //for(i=0; i<13; i++)
  //{
  //   ByteToStr(i,i_txt);
  //   Lcd_Out(2, 1, i_txt);
  //   Lcd_Chr(1, 2, Rem_per_Hour_Txt[i]);
  //   delay_ms(1000);
  //}
  ////////////////////////////////////////////////////////
  if(Rem_per_Hour_Txt[0] == '0')
  {
    Lcd_Out(1, 1, "0.0000");
    Lcd_Out(2, 2, "mrem/hr");
  }
  else
  {
  while(Rem_per_Hour_Txt[i] != 'e')
```

```
{
  i = i + 1;
}
i = i + 1;
if(Rem_per_Hour_Txt[i] == '+')
{
  goto exit_from_Convert_for_LCD;
}
else
{
  if(Rem_per_Hour_Txt[i] == '-')
  {
   i = i + 1;
  ////////////////////////////////////
  //ByteToStr(i,i_txt);
  //Lcd_Out(2, 1, i_txt);
  //Lcd_Chr(1, 2, Rem_per_Hour_Txt[i]);
  //delay_ms(2000);
  ////////////////////////////////////
   switch (Rem_per_Hour_Txt[i])
   {
      case '1': Lcd_Chr(1, 1, Rem_per_Hour_Txt[0]);
                Lcd_Chr(1, 2, Rem_per_Hour_Txt[2]);
                Lcd_Chr(1, 3, Rem_per_Hour_Txt[3]);
                Lcd_Chr(1, 4, Rem_per_Hour_Txt[1]);
                Lcd_Chr(1, 5, Rem_per_Hour_Txt[4]);
                Lcd_Chr(1, 6, Rem_per_Hour_Txt[5]);
                Lcd_Out(2, 2, "mrem/hr");          break;
```

```
case '2': Lcd_Chr(1, 1, Rem_per_Hour_Txt[0]);
        Lcd_Chr(1, 2, Rem_per_Hour_Txt[2]);
        Lcd_Chr(1, 3, Rem_per_Hour_Txt[1]);
        Lcd_Chr(1, 4, Rem_per_Hour_Txt[3]);
        Lcd_Chr(1, 5, Rem_per_Hour_Txt[4]);
        Lcd_Chr(1, 6, Rem_per_Hour_Txt[5]);
        Lcd_Out(2, 2, "mrem/hr");          break;


case '3': Lcd_Chr(1, 1, Rem_per_Hour_Txt[0]);
        Lcd_Chr(1, 2, Rem_per_Hour_Txt[1]);
        Lcd_Chr(1, 3, Rem_per_Hour_Txt[2]);
        Lcd_Chr(1, 4, Rem_per_Hour_Txt[3]);
        Lcd_Chr(1, 5, Rem_per_Hour_Txt[4]);
        Lcd_Chr(1, 6, Rem_per_Hour_Txt[5]);
        Lcd_Out(2, 2, "mrem/hr");          break;


case '4': Lcd_Chr(1, 1, '0');
        Lcd_Chr(1, 2, Rem_per_Hour_Txt[1]);
        Lcd_Chr(1, 3, Rem_per_Hour_Txt[0]);
        Lcd_Chr(1, 4, Rem_per_Hour_Txt[2]);
        Lcd_Chr(1, 5, Rem_per_Hour_Txt[3]);
        Lcd_Chr(1, 6, Rem_per_Hour_Txt[4]);
        Lcd_Out(2, 2, "mrem/hr");          break;

case '5': Lcd_Chr(1, 1, '0');
        Lcd_Chr(1, 2, Rem_per_Hour_Txt[1]);
        Lcd_Chr(1, 3, '0');
```

```
                Lcd_Chr(1, 4, Rem_per_Hour_Txt[0]);
                Lcd_Chr(1, 5, Rem_per_Hour_Txt[2]);
                Lcd_Chr(1, 6, Rem_per_Hour_Txt[3]);
                Lcd_Out(2, 2, "mrem/hr");          break;
      }
     }
     else
     {
       goto exit_from_Convert_for_LCD;
     }
  }
  exit_from_Convert_for_LCD:
  i = 0;
  }
}
////////////////////////////////////////////////////////////////////////////
// Change mode between Count per minute and mRem
//
////////////////////////////////////////////////////////////////////////////

void set_mode()
{
  Loop_start_mode:
  Lcd_Cmd(Lcd_CLEAR);
  while(mode_key == pressed)
  {
    Lcd_Out(1, 3, "MODE");
    Lcd_Out(2, 1, "CPM");
  }
```

```
LoopCPM:
if(ok_key == unpressed)
{
  if(mode_key == unpressed)
  {
    goto LoopCPM;
  }
  else
  {
    Lcd_Cmd(LCD_CLEAR);
    while(mode_key == pressed)
    {
      Lcd_Out(1, 3, "MODE");
      Lcd_Out(2, 1, "mRem");
    }
    LoopmRem:
    if(ok_key == unpressed)
    {
      if(mode_key == unpressed)
      {
        goto LoopmRem;
      }
      else
      {
        goto Loop_start_mode;
      }
    }
    else
    {
```

```
            modeflag = 2;
            show_mode = "mRem";
            goto exit_set_mode;
          }
        }
      }
      else
      {
        modeflag = 1;
        show_mode = "CPM";
      }
      exit_set_mode:
      mode_key = 1;
}
////////////////////////////////////////////////////////////////////////////
// Set time for count                                                     //
////////////////////////////////////////////////////////////////////////////
void set_time()
{
  start_set_time:
  Lcd_Cmd(Lcd_CLEAR);
  while(time_key == pressed)
  {
    Lcd_Out(1, 1, "Set time");
    Lcd_Out(2, 6, "1 M");
  }
  loop1M:
  if(ok_key == unpressed)
  {
```

```
if(time_key == unpressed)
{
  goto loop1M;
}
else
{
  Lcd_Cmd(LCD_CLEAR);
  while(time_key == pressed)
  {
    Lcd_Out(1, 1, "Set time");
    Lcd_Out(2, 6, "5 M");
  }
  loop5M:
  if(ok_key == unpressed)
  {
    if(time_key == unpressed)
    {
      goto loop5M;
    }
    else
    {
      Lcd_Cmd(LCD_CLEAR);
      while(time_key == pressed)
      {
        Lcd_Out(1, 1, "Set time");
        Lcd_Out(2, 6, "inf");
      }
      loopinf:
      if(ok_key == unpressed)
```

```
            {
              if(time_key == unpressed)
              {
                goto loopinf;
              }
              else
              {
                goto start_set_time;
              }
            }
            else
            {
              timeflag = 3;
              stop_time = 0;
              show_time = "inf";
              goto exit_set_time;
            }
          }
        }
        else
        {
          timeflag = 2;
          stop_time = 5;
          show_time = "5 M";
          goto exit_set_time;
        }
      }
    }
    else
```

```
  {
    timeflag = 1;
    stop_time = 60;
    show_time = "1 M";
    //Lcd_Out(2, 1, "Set 1M");              // exit 1 M
  }
  exit_set_time:
  time_key = 1;
}
////////////////////////////////////////////////////////////////////////////
// Select mode and time for count
////////////////////////////////////////////////////////////////////////////
void set_menu_count()
{
  start_menu_count:
  Lcd_Cmd(LCD_CLEAR);
  while(time_key == pressed || ok_key == pressed)
  {
    if(modeflag == 1)
    {
        switch (timeflag)
        {
          case 1: Lcd_Out(1, 2, "Count");
                  Lcd_Out(2, 1, "CPM");
                  Lcd_Out(2, 6, "1 M");  break;

          case 2: Lcd_Out(1, 2, "Count");
                  Lcd_Out(2, 1, "CPM");
                  Lcd_Out(2, 6, "5 M");  break;
```

```
            case 3: Lcd_Out(1, 2, "Count");
                    Lcd_Out(2, 1, "CPM");
                    Lcd_Out(2, 6, "Inf");  break;
          }
     //  Lcd_Out(1, 2, "Count");
     //  Lcd_Out(2, 1, show_mode);
     //  Lcd_Out(2, 6, show_time);
    }
    else
    {
      Lcd_Out(1, 2, "Count");
      Lcd_Out(2, 1, "mRem");
      //Lcd_Out(2, 6, show_time);
    }
}
loop_menu_count:
if(modeflag == 1)        // Mode CPM
{
   switch (timeflag)
        {
          case 1: Lcd_Out(1, 2, "Count");
                  Lcd_Out(2, 1, "CPM");
                  Lcd_Out(2, 6, "1 M");  break;

          case 2: Lcd_Out(1, 2, "Count");
                  Lcd_Out(2, 1, "CPM");
                  Lcd_Out(2, 6, "5 M");  break;
```

```
          case 3: Lcd_Out(1, 2, "Count");
                  Lcd_Out(2, 1, "CPM");
                  Lcd_Out(2, 6, "Inf");  break;
        }
  //Lcd_Out(1, 2, "Count");
  //Lcd_Out(2, 1, show_mode);
  //Lcd_Out(2, 6, show_time);
  if(ok_key == pressed)
  {
     count();
     Lcd_Out(2, 1, "Save  No");
     LoopSave:
     if(mode_key == pressed)
     {
////////////////////////////////////////////////////////
//    TEST
////////////////////////////////////////////////////////
//shortToStr(address,address_txt);
ShortToStr(timeflag,timeflag_txt);
Lcd_Out(2, 1, timeflag_txt);
delay_ms(2000);
////////////////////////////////////////////////////////
        if(timeflag == 1)
        {
           save_time = 1;
           save();
        }
        else
        {
```

```
      if(timeflag == 2)
      {
        save_time = 5;
        save();
      }
      else
      {
        Lcd_Cmd(LCD_CLEAR);
        while(ok_key == unpressed)
        {
          Lcd_Out(1, 3, Timer_txt);
          Lcd_Out(2, 1, "End");
          Lcd_Out(2, 6, show_time);
        }
        if(stop_count_flag == 1)
        {
          while(ok_key == pressed)
          {
            stop_count_flag = 0;
          }
        }
      goto exit_count;
      }
      exit_count:
      stop_count_flag = 0;
    }
}
else
{
```

```
      if(time_key == pressed)
      {
        goto exit_count_loop;
      }
      else
      {
        goto loopSave;
      }
    }
     exit_count_loop:
     Lcd_Cmd(LCD_CLEAR);
}
else
{
   if(mode_key == pressed)
   {
     set_mode();
     goto start_menu_count;
   }
   else
   {
     if(time_key == pressed)
     {
       set_time();
       goto start_menu_count;
     }
     else
     {
       goto loop_menu_count;
```

```
          }
        }
      }
  }
  else
  {
    Lcd_Out(1, 2, "Count");      // Mode mrem
    Lcd_Out(2, 1, "mrem");
    if(ok_key == pressed)
    {
      mRem_Count();
    }
    else
    {
      if(mode_key == pressed)
      {
        set_mode();
        goto start_menu_count;
      }
      else
      {
        goto loop_menu_count;
      }
    }
  }
}
////////////////////////////////////////////////////////////////////////////
void Memory_Read()
{
```

```
address = 1;
      i = 1;
while(ok_key == pressed)
{
   last_address = EEprom_Read(0x00);
}
Lcd_Cmd(LCD_CLEAR);
if(last_address == 0)
{
  while(ok_key == unpressed)
  {
  Lcd_Out(1, 2, "EEprom");
  Lcd_Out(2, 1, "NO DATA");
  }
}
else
{
   start_read:
   if(address != last_address)
   {
      Lcd_Out(1, 1, "DATA");
      ShortToStr(i, i_txt);
      Lcd_Out(1, 5, i_txt);

      data_read = EEprom_read(address);        // load Time
      ShortToStr(data_read,data_read_txt);
      Lcd_Chr(2, 7, data_read_txt[3]);
      Lcd_Chr(2, 8, 'M');
```

```
address = address + 1;
data_read = EEprom_read(address);         // load data low byte
TMR0L = data_read;
address = address + 1;
data_read = EEprom_read(address);
TMR0H = data_read;
Timer_0 = TMR0H;
Timer_0 = Timer_0 << 8;
Timer_0 = Timer_0 | TMR0L;
WordToStr(Timer_0,Timer_txt);
Lcd_Out(2,1,Timer_txt);
//delay_ms(3000);
while(ok_key == pressed)
{
   Lcd_Out(2,1,Timer_txt);
}
loop_Read:
if(ok_key == pressed)
{
  address = address + 1;
  i = i + 1;
  goto start_read;
}
else
{
   if(mode_key == pressed)
   {
      goto exit_read;
   }
```

```
          else
          {
            if(time_key == pressed)
            {
              goto exit_read;
            }
            else
            {
              goto loop_Read;
            }
          }
        }
      }
      else
      {
        while(ok_key == unpressed)
        {
        Lcd_Out(2, 1, "END DATA");
        }
      }
      exit_read:
      TMR0L = 0;
      TMR0H = 0;
    }
//  while(address != 255)
//  {
//    data_read = EEprom_Read(address);
//    ShortToStr(data_read,data_read_txt);
//    IntToStr(address,address_txt);
```

```
//    Lcd_Out(1, 1, address_txt);
//    Lcd_Out(2, 1, data_read_txt);
//    Delay_ms(1000);
//    address = address + 1;
//  }
}
////////////////////////////////////////////////////////////////////////////////
void Memory_Erase()
{
   Lcd_Cmd(LCD_CLEAR);
   while(ok_key == pressed)
   {
      Lcd_Out(1, 2, "MEMORY");
   }
   data = 0;
   address = 0;
   while(address != 255)
   {
      EEprom_Write(address, data);
      Lcd_Out(2, 1, "Erase.  ");
      delay_ms(5);
      Lcd_Chr(2, 7, '.');
      delay_ms(5);
      Lcd_Chr(2, 8, '.');
      address = address + 1;
   }
   IntToStr(address,address_txt);
   Lcd_Cmd(LCD_CLEAR);
   while(ok_key == unpressed)
```

```
  {
    Lcd_Out(1, 2, "EEprom");
    Lcd_Chr(2, 6, address_txt[3]);
    Lcd_Chr(2, 7, address_txt[4]);
    Lcd_Chr(2, 8, address_txt[5]);
    Lcd_Out(2, 1, "Free");
  }
}
////////////////////////////////////////////////////////////////////////////////
void save()
{
 if(timeflag = 1)
 {
   save_time = 1;
   address = EEprom_Read(0x00);      // Load last address
   ////////////////////////////////////////////////////
   //    TEST
   ////////////////////////////////////////////////////
   //shortToStr(address,address_txt);
   //Lcd_Out(2, 1, address_txt);
   //delay_ms(2000);
   ////////////////////////////////////////////////////
   if(address == 0)
   {
     address = address + 1;
   }
   EEprom_write(address,1);
   address = address + 1;
   EEprom_write(address,TMR0L);
```

```
  address = address + 1;
  EEprom_write(address,TMR0H);
  address = address + 1;
  EEprom_write(0x00,address);          // Save last address

}
else
{
   if(timeflag = 2)
   {
      save_time = 5;
      address = EEprom_Read(0x00);
  ////////////////////////////////////////////////
      if(address == 0)
      {
          address = address + 1;
      }
      EEprom_write(address,5);
      address = address + 1;
      EEprom_write(address,TMR0L);
      address = address + 1;
      EEprom_write(address,TMR0H);
      address = address + 1;
      EEprom_write(0x00,address);
   }
   else
   {
      goto exit_save;
   }
```

```
  }
  exit_save:
  Lcd_Cmd(LCD_CLEAR);
//  address = EEprom_Read(0);
//  data = TMR0L;
//  EEprom_Write(address,data);
//  address = address +1;
//  data = TMR0H;
//  EEprom_Write(address,data);
}
////////////////////////////////////////////////////////////////////////////////
void Memory_Check()
{
   Lcd_Cmd(LCD_CLEAR);
   while(ok_key == pressed)
   {
      Lcd_Out(1, 2, "MEMORY");
      Lcd_Out(2, 1, "Chk Mem");
   }
   data_read = EEprom_Read(0x00);
   empty = (255 - data_read);
   ShortToStr(data_read,data_read_txt);
   ShortToStr(empty,empty_txt);
   Lcd_Cmd(LCD_CLEAR);
   while(ok_key == unpressed)
   {
      Lcd_Out(1, 1, "DATA:");
      Lcd_Chr(1, 6, data_read_txt[1]);
      Lcd_Chr(1, 7, data_read_txt[2]);
```

```
      Lcd_Chr(1, 8, data_read_txt[3]);
      //Lcd_Out(1, 1, data_read_txt);
      Lcd_Out(2, 1, "Free:");
      Lcd_Out(2, 6, empty_txt);
    }
}
////////////////////////////////////////////////////////////////////////////
void set_menu_Memory()
{
   start_set_menu_memory:
   Lcd_Cmd(LCD_CLEAR);
   while(time_key == pressed || mode_key == pressed)
   {
      Lcd_Out(1, 2, "MEMORY");
      Lcd_Out(2, 1, "Read");
   }
   loop_read:
   if(mode_key == pressed)
   {
      Lcd_Cmd(LCD_CLEAR);
      while(mode_key == pressed)
      {
      Lcd_Out(1, 2, "MEMORY");
      Lcd_Out(2, 1, "Erase");
      }
      loop_erase:
      if(mode_key == pressed)
      {
         Lcd_Cmd(LCD_CLEAR);
```

```
  while(mode_key == pressed)
  {
    Lcd_Out(1, 2, "MEMORY");
    Lcd_Out(2, 1, "Chk Mem");
  }
  loop_check_mem:
  if(mode_key == pressed)
  {
    goto start_set_menu_memory;
  }
  else
  {
    if(ok_key == pressed)
    {
      Memory_Check();
    }
    else
    {
      if(time_key == pressed)
      {
        goto exit_from_memory;
      }
      else
      {
        goto loop_check_mem;
      }
    }
  }
}
```

```
  else
  {
    if(ok_key == pressed)
    {
      Memory_Erase();
    }
    else
    {
      if(time_key == pressed)
      {
        goto exit_from_memory;
      }
      else
      {
        goto loop_erase;
      }
    }
  }
}
else
{
  if(ok_key == pressed)
  {
    Memory_Read();
  }
  else
  {
    if(time_key == pressed)
    {
```

```
            goto exit_from_memory;
          }
          else
          {
            goto loop_read;
          }
        }
      }
      exit_from_memory:
      Lcd_Cmd(LCD_CLEAR);
}
////////////////////////////////////////////////////////////////////////////////
void SendAddress()
{
   write_buffer[0] = EEprom_read(0x00);
   write_buffer[1] = 0;
   write_buffer[2] = 0;
   write_buffer[3] = 0;
   Hid_Write(&Write_buffer,4);
}
////////////////////////////////////////////////////////////////////////////////
void MemoryReadDataNumber()
{
   number = Read_buffer[1];
   address = 1+(3*(number - 1));

      //last_address = EEprom_read(0x00);
      Write_buffer[0] = number;
```

```
      data_read = EEprom_read(address);   // Time
      Write_buffer[1] = data_read;

      address = address + 1;              // Data low byte
      data_read = EEprom_read(address);
      Write_buffer[2] = data_read;
      address = address + 1;              // Data high byte
      data_read = EEprom_read(address);
      Write_buffer[3] = data_read;
      Hid_Write(&Write_buffer,4);         // Send packet of data 4 byte
}
////////////////////////////////////////////////////////////////////////////
//  USB
////////////////////////////////////////////////////////////////////////////
void USB()
{
//   ADCON1 = 0xFF; // Set PORTB to digital I/O
//   TRISB = 0; // Set PORTB to outputs
//   PORTB = 0; // Clear all outputs
//   Lcd_Init(&PORTB);         // Initialize LCD connected to PORTB
   Lcd_Cmd(Lcd_CLEAR);       // Clear display
   Lcd_Out(1, 3, "USB");
//   Lcd_Cmd(Lcd_CURSOR_OFF);  // Turn cursor off
   //
   // Set interrupt registers to power-on defaults
   // Disable all interrupts
   //
   INTCON=0;
   INTCON2=0xF5;
```

```
  INTCON3=0xC0;
  RCON.IPEN=0;
  PIE1=0;
  PIE2=0;
  PIR1=0;
  PIR2=0;
//
// Configure TIMER 0 for 3.3ms interrupts. Set prescaler to 256
// and load TMR0L to 100 so that the time interval for timer
// interrupts at 48MHz is 256.(256-100).0.083 = 3.3ms
//
// The timer is in 8-bit mode by default
//
  T0CON = 0x47; // Prescaler = 256
  TMR0L = 100; // Timer count is 256-156 = 100
  INTCON.TMR0IE = 1; // Enable T0IE
  T0CON.TMR0ON = 1; // Turn Timer 0 ON
  INTCON = 0xE0; // Enable interrupts
//
// Enable USB port
//
  Hid_Enable(&Read_buffer, &Write_buffer);
  Delay_ms(1000);
  Delay_ms(1000);

//   WriteEEprom();
//
// Read from the USB port. Number of bytes read is in num
//
```

```
for(;;) // do forever
{
  num=0;
  while(num != 4) // Get 4 characters
  {
    num = Hid_Read();
  }
  Lcd_Out(2, 1, "Connect");
  switch(Read_buffer[0])
  {
    case 0x80:
          SendAddress();
          break;
    case 0x81:
          MemoryReadDataNumber();
          break;
     default:
          break;
  }
//   if(Read_buffer[0] == 'P' && Read_buffer[1] == '=' && Read_buffer[3] == 'T')
//   {
//     PORTB = Read_buffer[2];
//   }
}
  Hid_Disable();
}
////////////////////////////////////////////////////////////////////////////
// Change Menu
////////////////////////////////////////////////////////////////////////////
```

```
void set_menu()
{
  start_set_menu:
  Lcd_Cmd(LCD_CLEAR);
  while(ok_key == pressed)
  {
  Lcd_Out(1, 2, "COUNT");
  Lcd_Out(2, 1, "No    Ok");
  }
  loop_count:
  if(time_key == unpressed)
  {
    if(mode_key == unpressed)
    {
      if(ok_key == unpressed)
      {
        goto loop_count;
      }
      else
      {
        Lcd_Cmd(LCD_CLEAR);

        while(ok_key == pressed)
        {
          Lcd_Out(1, 2, "MEMORY");
          Lcd_Out(2, 1, "No    Ok");
        }
        loop_memory:
        if(time_key == unpressed)
```

```
{
  if(mode_key == unpressed)
  {
    if(ok_key == unpressed)
    {
      goto loop_memory;
    }
    else
    {
      Lcd_Cmd(LCD_CLEAR);   // USB menu
      while(ok_key == pressed)
      {
        Lcd_Out(1, 2, "USB");
        Lcd_Out(2, 1, "No    Ok");
      }
      loop_usb:
      if(time_key == unpressed)
      {
        if(mode_key == unpressed)
        {
          if(ok_key == unpressed)
          {
            goto loop_usb;
          }
          else
          {
            goto start_set_menu;
          }
        }
```

```
                else
                {
                  goto exit_set_menu;
                }
              }
              else
              {
                USB();
              }
            }
          }
          else
          {
            goto exit_set_menu;
          }
        }
        else
        {
          set_menu_Memory();
          //goto exit_set_menu;
        }
      }
    }
    else
    {
      goto exit_set_menu;
    }
  }
  else
```

```
   {
      set_menu_count();
   }
   exit_set_menu:
   mode_key = 1;
}
////////////////////////////////////////////////////////////////////////////
// Main Program                                                           //
////////////////////////////////////////////////////////////////////////////
void main()
{
  initialTMR0();
  initialTMR1();
  ADCON1 = 0x0D;
  TRISB = 0b00000011;               // PORTB is output
  TRISA = 0b00011100;
  Lcd_Init(&PORTB);         // Initialize LCD connected to PORTB
  Lcd_Cmd(Lcd_CLEAR);       // Clear display
  Lcd_Cmd(Lcd_CURSOR_OFF);  // Turn cursor off
  //Lcd_Out(1, 1, text);      // Print text to LCD, 1st row, 1st column
  PORTC = 0b11111111;
  TRISC = 0b00000000;
  PORTA.F2 = 1;
  PORTA.F3 = 1;
  PORTA.F4 = 1;
  PORTB.F1 = 1;
  TMR0L = 0x00;
  TMR0H = 0x00;
  countloop = 0;
```

```
  INTCON.GIE  = 1;
  INTCON.PEIE = 1;
  INTCON.TMR0IF = 0;
//  INTCON.TMR0IE = 1;
  PIR1.TMR1IF = 0;
  PIE1.TMR1IE = 1;
//  Hid_Enable(&Read_buffer, &Write_buffer);
//  Delay_ms(1000);
//  Delay_ms(1000);
  modeflag = 1;
  timeflag = 1;
  show_mode = "CPM";
  show_time = "1 M";
  while (1)
  {
    Lcd_Out(1, 2, text);
    if(ok_key == pressed)
    {
      set_menu();
      while(ok_key == pressed)
      {
        Lcd_Out(1, 2, text);
      }
      Lcd_Cmd(LCD_CLEAR);
    }
  }
}
```