Ex16-LCD-Ana

1

Generated by Doxygen 1.7.5

Thu Jun 21 2012 16:36:13

Contents

Chapter 1

File Index

1.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 2

File Documentation

2.1 00readme.c File Reference

Introduction.

2.1.1 Detailed Description

Introduction. This project toggles the LEDs on the timer and displays multiple messages on the first line of the LCD. The potentiometer on the Explorer 16 is read, and the value is displayed on the second line, in both a voltage and percentage.

When S3 is pressed, the top line of the display is blanked. Pressing S3 again restores the display. LED 8 follows S3 and LED 7 follows the top display status.

The application first sets the processor speed. In main.c, there are a number of configuration fuses set. By default, these work reasonably well on the Explorer 16, but it is preferable to be explicit about what they are doing.

The first configuration line:

```
_FOSCSEL( FNOSC_PRIPLL & IESO_OFF );
```

says to use the primary oscillator (i.e. the cystal), with the PLL system, and to start up with the user selected oscillator. An alternative is to start with a default internal RC oscillator, and then switch to the primary oscillator under program control.

The next line:

```
_FOSC( POSCMD_XT & FCKSM_CSECMD );
```

tells the dsPIC that the primary oscillator is an XT crystal. This basically affects the amount of power delivered to the crystal. EC is for very low power crystals, typically

watch crystals, XT is for "normal" crystals, and HS for high speed, typically >10MHz, crystals. It also says that it is permissible to switch clocks under program control, but should the selected oscillator fail, do not automatically switch to the fallback oscillator.

The third configuration line

```
_FWDT( FWDTEN_OFF );
```

disables the watchdog timer. If this were not done, the program would periodically reset, unless the program constantly resets the watchdog timer.

The next:

```
_FPOR( FPWRT_PWR64 );
```

holds off processor reset for 64 milliseconds after power has been applied. The idea is to give external circuitry an opportunity to stabilize before the program starts.

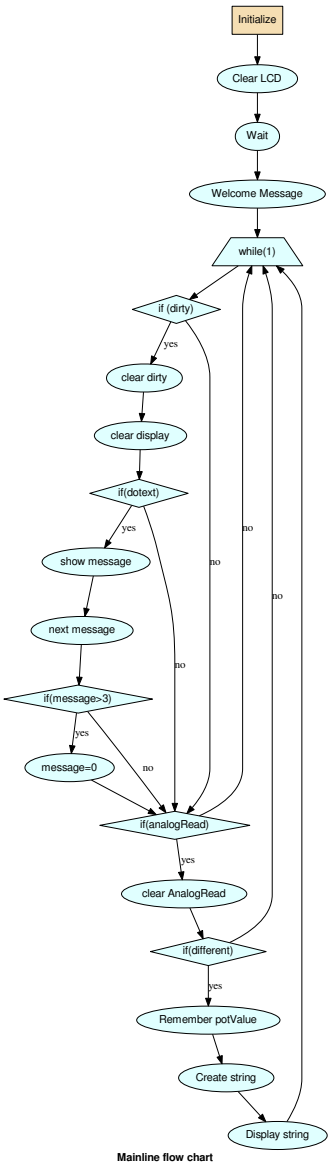
The final configuration line

```
_FICD( ICS_PGD1 & JTAGEN_OFF );
```

turns off the JTAG interface, and establishes PGD1/PGC1 as the pins for debug communication. There are three sets of programming pins on the dsPIC33FJ256GP701, so the developer may select a pair of pins that does not interfere with peripheral use for the selected circuit.

In Initialize(), two registers are set which determine how the PLL is configured. The CLKDIV register sets the pre- and post- PLL dividers which divide the clock before and after the PLL clock multiplier. PLLFBD sets the PLL feedback divisor which has the effect of multiplying the clock.

CLKDIV has a number of fields which allow the peripheral clock to be set slower than the instruction clock in some situations. These fields are not used, and are set to zero which essentially disables this feature.



Mainline flow chart

Definition in file 00readme.c.

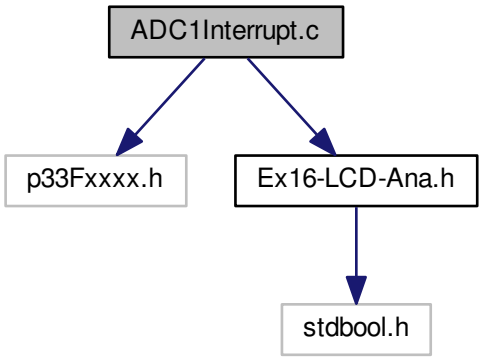
2.2 00readme.c

```
00001
```

2.3 ADC1Interrupt.c File Reference

Interrupt service routine for the Analog to Digital converter.

`#include <p33Fxxxx.h>` `#include "Ex16-LCD-Ana.h"` Include dependency graph for ADC1Interrupt.c:



Defines

- #define **EXTERN** extern

Functions

- void __attribute__ ((__interrupt__, auto_psv))

  ADC1 Interrupt Service Routine.

2.3.1 Detailed Description

Interrupt service routine for the Analog to Digital converter. This file provides the (very simple) ISR that is executed whenever an analog conversion has completed.

Definition in file ADC1Interrupt.c.

2.3.2 Function Documentation

2.3.2.1 void __attribute__ ((__interrupt__, auto_psv))

ADC1 Interrupt Service Routine.

Pseudocode:

```
    Clear the interrupt flag
    Grab the analog value and store it in potValue
    increment analogRead
```

Definition at line 22 of file ADC1Interrupt.c.

```
{
  IFS0bits.AD1IF = 0;       // Clear A/D interrupt flag
  potValue = ADC1BUF0;      // Save the potentiometer value
  analogRead++;             // Remember it has been read
}
```

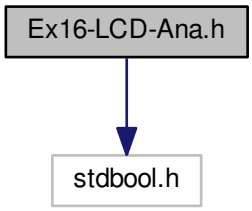
2.4 ADC1Interrupt.c

```
00001
00008 #include <p33Fxxxx.h>
00009
00010 #define EXTERN extern
00011 #include "Ex16-LCD-Ana.h"
00012
00014
00022 void __attribute__((__interrupt__, auto_psv)) _ADC1Interrupt( void )
00023 {
00024   IFS0bits.AD1IF = 0;       // Clear A/D interrupt flag
00025   potValue = ADC1BUF0;      // Save the potentiometer value
00026   analogRead++;             // Remember it has been read
00027 }
```

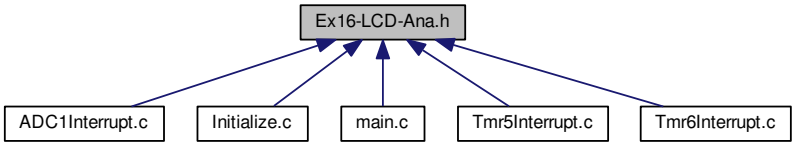
2.5 Ex16-LCD-Ana.h File Reference

Global declarations for Ex16-LCD-Ana.

`#include <stdbool.h>` Include dependency graph for Ex16-LCD-Ana.h:



This graph shows which files directly or indirectly include this file:



Defines

- #define LED7 LATAbits.LATA6

  Next to left LED latch.
- #define LED8 LATAbits.LATA7

  Leftmost LED latch.
- #define PB3 PORTDbits.RD6

  Leftmost pushbutton.

Functions

- void Initialize (void)

  Initialization for Ex16-LCD-Ana.

Variables

- EXTERN unsigned int analogRead

  Remember whether analog value has been read.
- EXTERN int dirty

  Dirty flag - if non-zero display is updated.
- EXTERN bool doText

  Indicate whether to display text message.
- EXTERN int message

  Current message number to display.
- EXTERN unsigned int potValue

  Value from the A/D converter.

2.5.1 Detailed Description

Global declarations for Ex16-LCD-Ana. File: Ex16-LCD-Ana.h Author: jjmcd

Created on June 19, 2012, 9:28 AM

Definition in file Ex16-LCD-Ana.h.

2.5.2 Define Documentation

2.5.2.1 #define LED7 LATAbits.LATA6

Next to left LED latch.

Definition at line 35 of file Ex16-LCD-Ana.h.

2.5.2.2 #define LED8 LATAbits.LATA7

Leftmost LED latch.

Definition at line 33 of file Ex16-LCD-Ana.h.

2.5.2.3 #define PB3 PORTDbits.RD6

Leftmost pushbutton.

Definition at line 37 of file Ex16-LCD-Ana.h.

2.5.3 Function Documentation

2.5.3.1 void Initialize (void)

Initialization for Ex16-LCD-Ana.

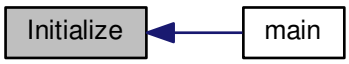
- Sets the processor clock to 40 MHz
- Initializes the ports
- Initializes timer 6
- Initializes timer 5
- Initialize the A/D converter
- Initializes the dirty flag and message number
- Initializes analogRead and doText
- Ensures the left two LEDs are off

Definition at line 43 of file Initialize.c.

```
{
    // Set the instruction clock speed
    //
    // Fcy 40 MIPS
    // DOZE = Fcy/8 = 011
    // DOZEN = 1
    // PLLPRE 2 = 00000
    // PLLDIV 40 = .38 = 0x26 = 0 0010 0110
    // PLLPOST 2 00
    //ROI   DOZE  DOZEN FRCDIV PLLPOST X   PLLPRE
    // 15 14 13 12  11  10 9 8   7 6   5  4 3 2 1 0
    //  0  0  0  0   0   0 0 0   0 0   0  0 0 0 0 0

    CLKDIV = 0x0000;
    PLLFBD = 0x0026;

    // Fcy 20 MIPS
    // PLLPRE 2 = 00000
    // PLLDIV 40 = .38 = 0x26 = 0 0010 0110
    // PLLPOST 4 01
    //ROI   DOZE  DOZEN FRCDIV PLLPOST X   PLLPRE
    // 15 14 13 12  11  10 9 8   7 6   5  4 3 2 1 0
    //  0  0  0  0   0   0 0 0   0 1   0  0 0 0 0 0
/*
    CLKDIV = 0x0008;
    PLLFBD = 0x0026;
*/

    TRISA = 0;              // All PORTA pins outputs
    LATA = 0x0001;          // Right LED on

    // Set timer 6 for right LED
    // Explanation ...
    //   Timer 6 will increment every 128 instruction cycles
    //   Once the count reaches 50,000, the timer 6 interrupt will fire
```

```
    //   and the count will be reset
    PR6 = 50000;            // Timer 6 counter to 50,000
    TMR6 = 0;               // Clear timer 6
    T6CON = 0x8030;         // 1:256 prescale, timer on, Clock Fcy
    IEC2bits.T6IE = 1;      // Enable Timer 6 interrupt

    // Set timer 5 for pushbutton monitor
    PR5 = 500;              // Timer 5 counter to 500
    TMR5 = 0;               // Clear timer 5
    T5CON = 0x8030;         // 1:256 prescale, timer on, Clock Fcy
    IEC1bits.T5IE = 1;      // Enable Timer 6 interrupt

    // Initialize the LCD
    LCDinit();

    // Initialize ADC
    /* set port configuration here */
    AD1PCFGLbits.PCFG4 = 0;         // ensure AN4/RB4 is analog (Temp Sensor)
    AD1PCFGLbits.PCFG5 = 0;         // ensure AN5/RB5 is analog (Analog Pot)
    /* set channel scanning here, auto sampling and convert,
       with default read-format mode */
    AD1CON1 = 0x00E4;
    /* select 12-bit, 1 channel ADC operation */
    AD1CON1bits.AD12B = 1;
    /* No channel scan for CH0+, Use MUX A,
       SMPI = 1 per interrupt, Vref = AVdd/AVss */
    AD1CON2 = 0x0000;
    /* Set Samples and bit conversion time */
    AD1CON3 = 0x032F;
    /* set channel scanning here for AN4 and AN5 */
    AD1CSSL = 0x0000;
    /* channel select AN5/RB5 */
    AD1CHS0 = 0x0005;
    /* reset ADC interrupt flag */
    IFS0bits.AD1IF = 0;
    /* enable ADC interrupts */
    IEC0bits.AD1IE = 1;
     /* turn on ADC module */
    AD1CON1bits.ADON = 1;

    // Initialize global variables
    dirty = 0;              // Message dirty flag
    message = 0;            // Current message number
    analogRead = 0;         // Set to A/D not read
    doText = true;          // Start with text display
    LED8 = LED7 = 0;

}
```

Here is the caller graph for this function:



2.5.4 Variable Documentation

2.5.4.1 EXTERN unsigned int analogRead

Remember whether analog value has been read.

Definition at line 27 of file Ex16-LCD-Ana.h.

2.5.4.2 EXTERN int dirty

Dirty flag - if non-zero display is updated.

Definition at line 21 of file Ex16-LCD-Ana.h.

2.5.4.3 EXTERN bool doText

Indicate whether to display text message.

Definition at line 29 of file Ex16-LCD-Ana.h.

2.5.4.4 EXTERN int message

Current message number to display.

Definition at line 23 of file Ex16-LCD-Ana.h.

2.5.4.5 EXTERN unsigned int potValue

Value from the A/D converter.

Definition at line 25 of file Ex16-LCD-Ana.h.

2.6 Ex16-LCD-Ana.h

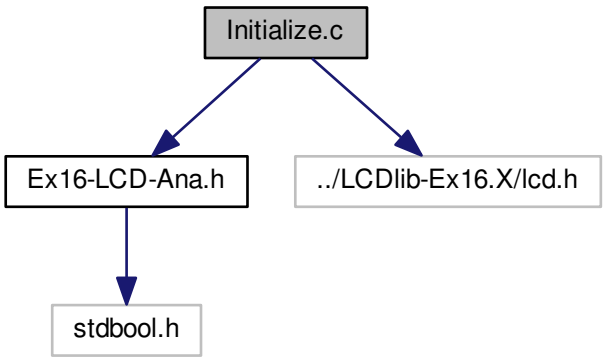
```
00001
00011 #ifndef EX16_LCD_ANA_H
00012 #define EX16_LCD_ANA_H
00013
00014 #ifdef  __cplusplus
00015 extern "C" {
00016 #endif
00017
00018 #include <stdbool.h>
00019
00021 EXTERN int dirty;
00023 EXTERN int message;
00025 EXTERN unsigned int potValue;
00027 EXTERN unsigned int analogRead;
```

```
00029 EXTERN bool doText;
00030
00031 // Macros for various devices
00033 #define LED8 LATAbits.LATA7
00034
00035 #define LED7 LATAbits.LATA6
00036
00037 #define PB3 PORTDbits.RD6
00038
00039
00041 void Initialize( void );
00042
00043
00044 #ifdef  __cplusplus
00045 }
00046 #endif
00047
00048 #endif  /* EX16_LCD_ANA_H */
00049
```

2.7 Initialize.c File Reference

Initialization for Ex16-LCD-Ana.

`#include "Ex16-LCD-Ana.h" #include "../LCDlib-Ex16.X/lcd.-h"` Include dependency graph for Initialize.c:



Defines

- #define **EXTERN** extern

Functions

- void Initialize (void)

  Initialization for Ex16-LCD-Ana.

2.7.1 Detailed Description

Initialization for Ex16-LCD-Ana.

Definition in file Initialize.c.

2.7.2 Function Documentation

2.7.2.1 void Initialize (void)

Initialization for Ex16-LCD-Ana.

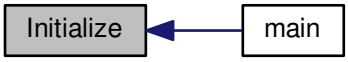
- Sets the processor clock to 40 MHz
- Initializes the ports
- Initializes timer 6
- Initializes timer 5
- Initialize the A/D converter
- Initializes the dirty flag and message number
- Initializes analogRead and doText
- Ensures the left two LEDs are off

Definition at line 43 of file Initialize.c.

```
{
    // Set the instruction clock speed
    //
    // Fcy 40 MIPS
    // DOZE = Fcy/8 = 011
    // DOZEN = 1
    // PLLPRE 2 = 00000
    // PLLDIV 40 = .38 = 0x26 = 0 0010 0110
    // PLLPOST 2 00
    //ROI   DOZE  DOZEN FRCDIV PLLPOST X   PLLPRE
    // 15 14 13 12  11  10 9 8   7 6   5  4 3 2 1 0
    //  0  0  0  0   0   0 0 0   0 0   0  0 0 0 0 0

    CLKDIV = 0x0000;
    PLLFBD = 0x0026;
```

```
    // Fcy 20 MIPS
    // PLLPRE 2 = 00000
    // PLLDIV 40 = .38 = 0x26 = 0 0010 0110
    // PLLPOST 4 01
    //ROI   DOZE  DOZEN FRCDIV PLLPOST X   PLLPRE
    // 15 14 13 12  11  10 9 8   7 6   5  4 3 2 1 0
    //  0  0  0  0   0   0 0 0   0 1   0  0 0 0 0 0
/*
    CLKDIV = 0x0008;
    PLLFBD = 0x0026;
*/

    TRISA = 0;              // All PORTA pins outputs
    LATA = 0x0001;          // Right LED on

    // Set timer 6 for right LED
    // Explanation ...
    //   Timer 6 will increment every 128 instruction cycles
    //   Once the count reaches 50,000, the timer 6 interrupt will fire
    //   and the count will be reset
    PR6 = 50000;            // Timer 6 counter to 50,000
    TMR6 = 0;               // Clear timer 6
    T6CON = 0x8030;         // 1:256 prescale, timer on, Clock Fcy
    IEC2bits.T6IE = 1;      // Enable Timer 6 interrupt

    // Set timer 5 for pushbutton monitor
    PR5 = 500;              // Timer 5 counter to 500
    TMR5 = 0;               // Clear timer 5
    T5CON = 0x8030;         // 1:256 prescale, timer on, Clock Fcy
    IEC1bits.T5IE = 1;      // Enable Timer 6 interrupt

    // Initialize the LCD
    LCDinit();

    // Initialize ADC
    /* set port configuration here */
    AD1PCFGLbits.PCFG4 = 0;         // ensure AN4/RB4 is analog (Temp Sensor)
    AD1PCFGLbits.PCFG5 = 0;         // ensure AN5/RB5 is analog (Analog Pot)
    /* set channel scanning here, auto sampling and convert,
       with default read-format mode */
    AD1CON1 = 0x00E4;
    /* select 12-bit, 1 channel ADC operation */
    AD1CON1bits.AD12B = 1;
    /* No channel scan for CH0+, Use MUX A,
       SMPI = 1 per interrupt, Vref = AVdd/AVss */
    AD1CON2 = 0x0000;
    /* Set Samples and bit conversion time */
    AD1CON3 = 0x032F;
    /* set channel scanning here for AN4 and AN5 */
    AD1CSSL = 0x0000;
    /* channel select AN5/RB5 */
    AD1CHS0 = 0x0005;
    /* reset ADC interrupt flag */
    IFS0bits.AD1IF = 0;
    /* enable ADC interrupts */
    IEC0bits.AD1IE = 1;
     /* turn on ADC module */
    AD1CON1bits.ADON = 1;

    // Initialize global variables
    dirty = 0;              // Message dirty flag
    message = 0;            // Current message number
    analogRead = 0;         // Set to A/D not read
    doText = true;          // Start with text display
    LED8 = LED7 = 0;

}
```

Here is the caller graph for this function:



2.8 Initialize.c

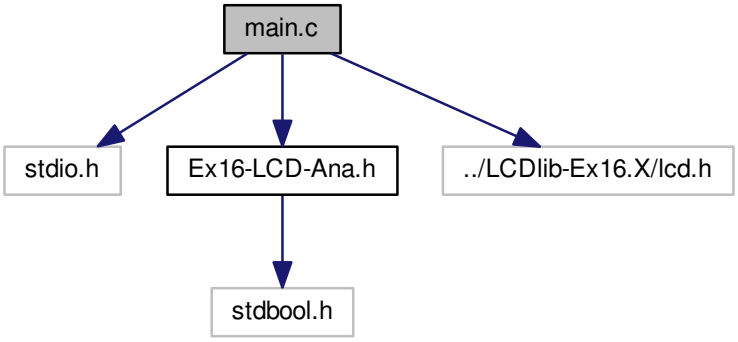
```
00001
00007 #if defined(__PIC24E__)
00008 #include <p24Exxxx.h>
00009
00010 #elif defined (__PIC24F__)
00011 #include <p24Fxxxx.h>
00012
00013 #elif defined(__PIC24H__)
00014 #include <p24Hxxxx.h>
00015
00016 #elif defined(__dsPIC30F__)
00017 #include <p30Fxxxx.h>
00018
00019 #elif defined (__dsPIC33E__)
00020 #include <p33Exxxx.h>
00021
00022 #elif defined(__dsPIC33F__)
00023 #include <p33Fxxxx.h>
00024
00025 #endif
00026
00027 #define EXTERN extern
00028 #include "Ex16-LCD-Ana.h"
00029
00030 #include "../LCDlib-Ex16.X/lcd.h"
00031
00033
00043 void Initialize( void )
00044 {
00045     // Set the instruction clock speed
00046     //
00047     // Fcy 40 MIPS
00048     // DOZE = Fcy/8 = 011
00049     // DOZEN = 1
00050     // PLLPRE 2 = 00000
00051     // PLLDIV 40 = .38 = 0x26 = 0 0010 0110
00052     // PLLPOST 2 00
00053     //ROI    DOZE  DOZEN FRCDIV PLLPOST X    PLLPRE
00054     // 15 14 13 12  11  10 9 8   7 6   5  4 3 2 1 0
00055     //  0  0  0  0   0   0 0 0   0 0   0  0 0 0 0 0
00056
00057     CLKDIV = 0x0000;
00058     PLLFBD = 0x0026;
00059
00060     // Fcy 20 MIPS
```

```
00061     // PLLPRE 2 = 00000
00062     // PLLDIV 40 = .38 = 0x26 = 0 0010 0110
00063     // PLLPOST 4 01
00064     //ROI    DOZE  DOZEN FRCDIV PLLPOST X   PLLPRE
00065     // 15 14 13 12  11  10 9 8   7 6   5  4 3 2 1 0
00066     //  0  0  0  0   0   0 0 0   0 1   0  0 0 0 0 0
00067 /*
00068     CLKDIV = 0x0008;
00069     PLLFBD = 0x0026;
00070 */
00071
00072     TRISA = 0;              // All PORTA pins outputs
00073     LATA = 0x0001;          // Right LED on
00074
00075     // Set timer 6 for right LED
00076     // Explanation ...
00077     //   Timer 6 will increment every 128 instruction cycles
00078     //   Once the count reaches 50,000, the timer 6 interrupt will fire
00079     //   and the count will be reset
00080     PR6 = 50000;            // Timer 6 counter to 50,000
00081     TMR6 = 0;               // Clear timer 6
00082     T6CON = 0x8030;         // 1:256 prescale, timer on, Clock Fcy
00083     IEC2bits.T6IE = 1;      // Enable Timer 6 interrupt
00084
00085     // Set timer 5 for pushbutton monitor
00086     PR5 = 500;              // Timer 5 counter to 500
00087     TMR5 = 0;               // Clear timer 5
00088     T5CON = 0x8030;         // 1:256 prescale, timer on, Clock Fcy
00089     IEC1bits.T5IE = 1;      // Enable Timer 6 interrupt
00090
00091     // Initialize the LCD
00092     LCDinit();
00093
00094     // Initialize ADC
00095     /* set port configuration here */
00096     AD1PCFGLbits.PCFG4 = 0;         // ensure AN4/RB4 is analog (Temp Sensor)
00097     AD1PCFGLbits.PCFG5 = 0;         // ensure AN5/RB5 is analog (Analog Pot)
00098     /* set channel scanning here, auto sampling and convert,
00099        with default read-format mode */
00100     AD1CON1 = 0x00E4;
00101     /* select 12-bit, 1 channel ADC operation */
00102     AD1CON1bits.AD12B = 1;
00103     /* No channel scan for CH0+, Use MUX A,
00104        SMPI = 1 per interrupt, Vref = AVdd/AVss */
00105     AD1CON2 = 0x0000;
00106     /* Set Samples and bit conversion time */
00107     AD1CON3 = 0x032F;
00108     /* set channel scanning here for AN4 and AN5 */
00109     AD1CSSL = 0x0000;
00110     /* channel select AN5/RB5 */
00111     AD1CHS0 = 0x0005;
00112     /* reset ADC interrupt flag */
00113     IFS0bits.AD1IF = 0;
00114     /* enable ADC interrupts */
00115     IEC0bits.AD1IE = 1;
00116      /* turn on ADC module */
00117     AD1CON1bits.ADON = 1;
00118
00119     // Initialize global variables
00120     dirty = 0;              // Message dirty flag
00121     message = 0;            // Current message number
00122     analogRead = 0;         // Set to A/D not read
00123     doText = true;          // Start with text display
00124     LED8 = LED7 = 0;
00125
00126 }
```

2.9 main.c File Reference

Mainline for Ex16-LCD-Ana.

`#include <stdio.h>  #include "Ex16-LCD-Ana.h"  #include "../LCDlib-Ex16.X/lcd.h"` Include dependency graph for main.c:



Functions

- _FICD (ICS_PGD1 &JTAGEN_OFF)

  Communicate on PGC1/EMUC1 and PGD1/EMUD1, JTAG is Disabled.
- _FOSC (POSCMD_XT &FCKSM_CSECMD)

  XT Oscillator Mode, Clock switching is enabled, Fail-Safe Clock Monitor is disabled.
- _FOSCSEL (FNOSC_PRIPLL &IESO_OFF)

  Primary Oscillator (XT, HS, EC) w/ PLL, Start up with user-selected oscillator.
- _FPOR (FPWRT_PWR64)

  Power-on reset timer 64 ms.
- _FWDT (FWDTEN_OFF)

  Watchdog timer enabled/disabled by user software.
- int main (void)

  Mainline for Ex16-LCD-Ana.

Variables

- char szMessage [9][17]

  Table of messages to be displayed.

2.9.1 Detailed Description

Mainline for Ex16-LCD-Ana. This application is intended to show use of the timer and the LCD. A flag is passed from the ISR to the mainline to indicate time to update the display.

A second line of the display shows the potentiometer position, in both voltage and percentage. The second display line is updated far faster than the top line, providing the value changes.

Pressing S3 toggles the first line of the display on and off.

File: main.c Author: jjmcd

Created on June 19, 2012, 9:27 AM

Definition in file main.c.

2.9.2 Function Documentation

2.9.2.1 _FICD (ICS_PGD1 & *JTAGEN_OFF*)

Communicate on PGC1/EMUC1 and PGD1/EMUD1, JTAG is Disabled.

2.9.2.2 _FOSC (POSCMD_XT & *FCKSM_CSECMD*)

XT Oscillator Mode, Clock switching is enabled, Fail-Safe Clock Monitor is disabled.

2.9.2.3 _FOSCSEL (FNOSC_PRIPLL & *IESO_OFF*)

Primary Oscillator (XT, HS, EC) w/ PLL, Start up with user-selected oscillator.

2.9.2.4 _FPOR (FPWRT_PWR64)

Power-on reset timer 64 ms.

2.9.2.5 _FWDT (FWDTEN_OFF)

Watchdog timer enabled/disabled by user software.

2.9.2.6 int main (void)

Mainline for Ex16-LCD-Ana.

Display a selected message and analog value on the LCD

Pseudocode:

```
Initialize()
Clear the LCD display
Delay one dirty flag cycle
Display a welcome message
Wait until ready to clear display
do forever
  if the dirty flag is set
    clear the dirty flag
    clear the display
    if doText is true
      display the current message
      increment the message number
      if we are at the end of messages
        point to the first message
    Set oldValue to impossible value
  if a new analog value is available
    remember we read the value
    if the value has changed enough to matter
      Set oldValue to potValue
      Create a string containing voltage and percentage
      display the string on the second line
```

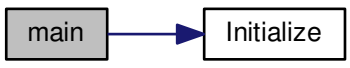
Remember previous analog value

Definition at line 122 of file main.c.

```
{
    int oldValue;

    // Initialize ports and variables
    Initialize();

    // Clear the screen
    LCDclear();

    // Wait a while to pretend like we are thinking hard
    dirty = 0;
    while ( !dirty )
        ;
    dirty = 0;

    // Display a friendly welcome mesage
    LCDputs(" To disable top  line press S3 ");

    //Hold off initial analog display until ready to clear welcome message
    while ( !dirty )
        ;
```

```
    while (1)
    {
        // If the message needs to be updated
        if ( dirty )
        {
            // Remember we did it
            dirty = 0;
            // Clear the display
            LCDclear();
            if ( doText )
            {
                // Display the current message
                LCDputs(szMessage[message]);
                // Point to the next message
                message++;
                // If we are at the end of the messages
                if ( message > 8 )
                    // point back to the firest message
                    message = 0;
            }
            // Force display of analog
            oldValue = 10000;
        }
        if ( analogRead )
        {
            // Work string for display
            char szValue[16];

            // Remember we read the analog
            analogRead = 0;

            // Check enough difference to display
            // (to prevent jitter in the last digit)
            if ( abs( oldValue-potValue ) > 10 )
            {
                // Remember current value
                oldValue = potValue;
                // Place the voltage and percentage into the string
                sprintf(szValue,"%5.3fV  %5.2f%%",
                        3.3*(float)potValue/4095.0,
                        100.0*(float)potValue/4095.0 );
                // Position to the second line and write string to LCD
                LCDposition( 0x40+1 );
                LCDputs(szValue);
            }
        }

    }
}
```

Here is the call graph for this function:



2.9.3 Variable Documentation

2.9.3.1 char szMessage[9][17]

Initial value:

```
{
    " Twas brillig,  ",
    " and the slithy ",
    "toves, did gyre ",
    " and gimble in  ",
    "   the wabe:    ",
    " All mimsy were ",
    " the borogoves, ",
    "  And the mome  ",
    "raths outgrabe. "
}
```

Table of messages to be displayed.

Definition at line 81 of file main.c.

2.10 main.c

```
00001
00021 /*********************************************************************************
00022  * Software License Agreement
00023  *
00024  * GPLV2+
00025  *
00026  *********************************************************************************/
00027
00028
00029 #if defined(__PIC24E__)
00030 #include <p24Exxxx.h>
00031
00032 #elif defined (__PIC24F__)
00033 #include <p24Fxxxx.h>
00034
00035 #elif defined(__PIC24H__)
00036 #include <p24Hxxxx.h>
00037
00038 #elif defined(__dsPIC30F__)
00039 #include <p30Fxxxx.h>
00040
00041 #elif defined (__dsPIC33E__)
00042 #include <p33Exxxx.h>
00043
00044 #elif defined(__dsPIC33F__)
00045 #include <p33Fxxxx.h>
00046
00047 #endif
00048
00049 #include <stdio.h>
00050
00051
00052 /*  This is cheating
00053  *
00054  * This is sort of a trick.  Global variables must be defined once,
```

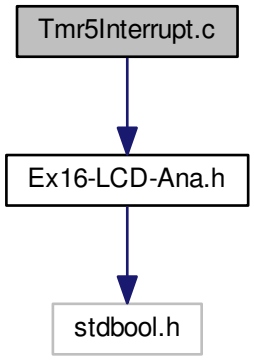
```
00055  * but anyplace they are used, they must be referenced as extern.  To
00056  * simplify keeping track, globals are declared in the header file
00057  * as EXTERN.  In the mainline, EXTERN is defined as nothing before
00058  * the header is included.  In all other files, EXTERN is declared
00059  * as extern.  This way all globals are created in the mainline but
00060  * are visible to all the other routines.
00061  */
00062 #define EXTERN
00063 #include "Ex16-LCD-Ana.h"
00064 // Notice that the LCD header file is provided by the LCD library project
00065 #include "../LCDlib-Ex16.X/lcd.h"
00066
00067 // Configuration fuses
00068 //
00070 _FOSCSEL( FNOSC_PRIPLL & IESO_OFF );
00072 _FOSC( POSCMD_XT & FCKSM_CSECMD );
00074 _FWDT( FWDTEN_OFF );
00076 _FPOR( FPWRT_PWR64 );
00078 _FICD( ICS_PGD1 & JTAGEN_OFF );
00079
00081 char szMessage[9][17] =
00082 {
00083     " Twas brillig,  ",
00084     " and the slithy ",
00085     "toves, did gyre ",
00086     " and gimble in  ",
00087     "   the wabe:    ",
00088     " All mimsy were ",
00089     " the borogoves, ",
00090     "  And the mome  ",
00091     "raths outgrabe. "
00092 };
00093
00095
00122 int main(void)
00123 {
00125     int oldValue;
00126
00127     // Initialize ports and variables
00128     Initialize();
00129
00130     // Clear the screen
00131     LCDclear();
00132
00133     // Wait a while to pretend like we are thinking hard
00134     dirty = 0;
00135     while ( !dirty )
00136         ;
00137     dirty = 0;
00138
00139     // Display a friendly welcome mesage
00140     LCDputs(" To disable top  line press S3 ");
00141
00142     //Hold off initial analog display until ready to clear welcome message
00143     while ( !dirty )
00144         ;
00145
00146     while (1)
00147     {
00148         // If the message needs to be updated
00149         if ( dirty )
00150         {
00151             // Remember we did it
00152             dirty = 0;
00153             // Clear the display
00154             LCDclear();
00155             if ( doText )
00156             {
```

```
00157                 // Display the current message
00158                 LCDputs(szMessage[message]);
00159                 // Point to the next message
00160                 message++;
00161                 // If we are at the end of the messages
00162                 if ( message > 8 )
00163                     // point back to the firest message
00164                     message = 0;
00165             }
00166             // Force display of analog
00167             oldValue = 10000;
00168         }
00169         if ( analogRead )
00170         {
00171             // Work string for display
00172             char szValue[16];
00173 
00174             // Remember we read the analog
00175             analogRead = 0;
00176 
00177             // Check enough difference to display
00178             // (to prevent jitter in the last digit)
00179             if ( abs( oldValue-potValue ) > 10 )
00180             {
00181                 // Remember current value
00182                 oldValue = potValue;
00183                 // Place the voltage and percentage into the string
00184                 sprintf(szValue,"%5.3fV  %5.2f%%",
00185                         3.3*(float)potValue/4095.0,
00186                         100.0*(float)potValue/4095.0 );
00187                 // Position to the second line and write string to LCD
00188                 LCDposition( 0x40+1 );
00189                 LCDputs(szValue);
00190             }
00191         }
00192 
00193     }
00194 }
```

2.11 Tmr5Interrupt.c File Reference

Timer 5 interrupt service routine.

`#include "Ex16-LCD-Ana.h"` Include dependency graph for Tmr5Interrupt.c:



Defines

- #define **EXTERN** extern

Functions

- void __attribute__ ((__interrupt__, auto_psv))

  Timer 5 Interrupt Service Routine.

Variables

- int last

  Counter used to delay toggling dirty flag.
- int offCount

  Number of interrupts PB3 has been released.
- int onCount

  Number of interrupts PB3 has been pressed.

2.11.1 Detailed Description

Timer 5 interrupt service routine. Whenever Timer 5 expires, this routine illuminates LED8 to follow the state of PB3. If PB3 has been pressed for a while, the state of doText is toggled. LED7 is illuminated it the doText flag is false.

The mainline uses doText to determine whether to display the top line of the LCD. LED7 is illuminated if the text is NOT displayed. Pressing and releasing PB3 changes the state.

Definition in file Tmr5Interrupt.c.

2.11.2 Function Documentation

2.11.2.1 void __attribute__ ((__interrupt__, auto_psv))

Timer 5 Interrupt Service Routine.

Gets executed whenever Timer 5 expires.

Causes LED8 to track PB3. If PB3 is released for a while, and it had previously been pressed for a while, the state of doText is toggled. If doText is true, LED7 is illuminated.

Pseudocode

```
Clear interrupt flag
Set LED8 to be complement of PB3
if PB3 is pressed
  Increment onCount
  Clear offCount
otherwise
  Increment offCount
  if PB3 has been released for a while
    if PB3 had been pressed for a while
      Complement doText
      Clear onCount
Set LED7 to complement of doText
```

Definition at line 70 of file Tmr5Interrupt.c.

```
{
    IFS1bits.T5IF = 0;              // Clear timer interrupt flag
                                    // This is always the first order of
                                    // business in an interrupt routine

    LED8 = !PB3;                    // LED8 follows PB3

    if ( !PB3 )                     // PB3 depressed
    {
        onCount++;                  // Count up time pressed
        offCount = 0;               // and reset un-pressed count
    }
    else                            // PB3 released
    {
        offCount++;                 // Increment released count
        if ( offCount > 5 )         // Released for a while
```

```
            if ( onCount > 5 )      // Was it actually pressed?
            {
                doText = !doText;   // Toggle text display
                onCount = 0;        // Reset pressed count
            }
    }
    LED7 = !doText;                 // LED7 follows text display state
}
```

2.11.3 Variable Documentation

2.11.3.1 int last

Counter used to delay toggling dirty flag.

Definition at line 44 of file Tmr5Interrupt.c.

2.11.3.2 int offCount

Number of interrupts PB3 has been released.

Definition at line 42 of file Tmr5Interrupt.c.

2.11.3.3 int onCount

Number of interrupts PB3 has been pressed.

Definition at line 40 of file Tmr5Interrupt.c.

2.12 Tmr5Interrupt.c

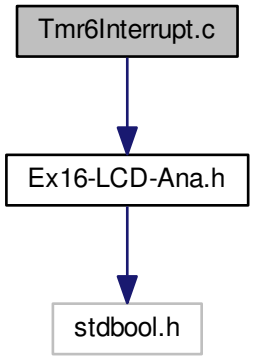
```
00001
00016 #if defined(__PIC24E__)
00017 #include <p24Exxxx.h>
00018
00019 #elif defined (__PIC24F__)
00020 #include <p24Fxxxx.h>
00021
00022 #elif defined(__PIC24H__)
00023 #include <p24Hxxxx.h>
00024
00025 #elif defined(__dsPIC30F__)
00026 #include <p30Fxxxx.h>
00027
00028 #elif defined (__dsPIC33E__)
00029 #include <p33Exxxx.h>
00030
00031 #elif defined(__dsPIC33F__)
00032 #include <p33Fxxxx.h>
00033
00034 #endif
00035
```

```
00036 #define EXTERN extern
00037 #include "Ex16-LCD-Ana.h"
00038
00040 int onCount;
00042 int offCount;
00044 int last;
00045
00047
00070 void __attribute__((__interrupt__, auto_psv)) _T5Interrupt( void )
00071 {
00072     IFS1bits.T5IF = 0;              // Clear timer interrupt flag
00073                                     // This is always the first order of
00074                                     // business in an interrupt routine
00075
00076     LED8 = !PB3;                    // LED8 follows PB3
00077
00078     if ( !PB3 )                     // PB3 depressed
00079     {
00080         onCount++;                  // Count up time pressed
00081         offCount = 0;               // and reset un-pressed count
00082     }
00083     else                            // PB3 released
00084     {
00085         offCount++;                 // Increment released count
00086         if ( offCount > 5 )         // Released for a while
00087             if ( onCount > 5 )      // Was it actually pressed?
00088             {
00089                 doText = !doText;   // Toggle text display
00090                 onCount = 0;        // Reset pressed count
00091             }
00092     }
00093     LED7 = !doText;                 // LED7 follows text display state
00094 }
```

2.13 Tmr6Interrupt.c File Reference

Timer 6 interrupt service routine.

`#include "Ex16-LCD-Ana.h"` Include dependency graph for Tmr6Interrupt.c:



Defines

- #define **EXTERN** extern

Functions

- void __attribute__ ((__interrupt__, auto_psv))

  Timer 6 Interrupt Service Routine.

Variables

- int delayCount

  Counter used to delay toggling dirty flag.

2.13.1 Detailed Description

Timer 6 interrupt service routine. Whenever Timer 6 expires, this routine toggles the rightmost 2 LEDs. After 5 interrupts, it sets the dirty flag causing the mainline to display a new message on the LCD.

Definition in file Tmr6Interrupt.c.

2.13.2 Function Documentation

2.13.2.1 void __attribute__ ((__interrupt__, auto_psv))

Timer 6 Interrupt Service Routine.

Gets executed whenever Timer 6 expires

Pseudocode:

```
Clear timer interrupt flag
Toggle right 2 LEDs (XOR LATA with 3)
increment delayCount
if  delayCount > 5
    Set dirty flag
    Reset delay count
```

Definition at line 50 of file Tmr6Interrupt.c.

```
{
    IFS2bits.T6IF = 0;      // Clear timer interrupt flag
                            // This is always the first order of
                            // business in an interrupt routine

    LATA ^= 0x0003;         // Toggle right 2 LEDs
    delayCount++;           // Increment delayCount
    if ( delayCount > 5 )   // Only update display every 5
    {                       // toggles of LEDs
        dirty = 1;          // Set the dirty flag
        delayCount = 0;     // Reset the delayCount
    }
}
```

2.13.3 Variable Documentation

2.13.3.1 int delayCount

Counter used to delay toggling dirty flag.

Definition at line 35 of file Tmr6Interrupt.c.

2.14 Tmr6Interrupt.c

```
00001
00011 #if defined(__PIC24E__)
00012 #include <p24Exxxx.h>
00013
00014 #elif defined (__PIC24F__)
00015 #include <p24Fxxxx.h>
00016
00017 #elif defined(__PIC24H__)
00018 #include <p24Hxxxx.h>
00019
00020 #elif defined(__dsPIC30F__)
```

```
00021 #include <p30Fxxxx.h>
00022
00023 #elif defined (__dsPIC33E__)
00024 #include <p33Exxxx.h>
00025
00026 #elif defined(__dsPIC33F__)
00027 #include <p33Fxxxx.h>
00028
00029 #endif
00030
00031 #define EXTERN extern
00032 #include "Ex16-LCD-Ana.h"
00033
00035 int delayCount;
00036
00038
00050 void __attribute__((__interrupt__, auto_psv)) _T6Interrupt( void )
00051 {
00052     IFS2bits.T6IF = 0;      // Clear timer interrupt flag
00053                             // This is always the first order of
00054                             // business in an interrupt routine
00055
00056     LATA ^= 0x0003;         // Toggle right 2 LEDs
00057     delayCount++;           // Increment delayCount
00058     if ( delayCount > 5 )   // Only update display every 5
00059     {                       // toggles of LEDs
00060         dirty = 1;          // Set the dirty flag
00061         delayCount = 0;     // Reset the delayCount
00062     }
00063 }
```