# LCDlib-Ex16

1

Generated by Doxygen 1.7.5

Thu Jun 21 2012 12:57:55

# Contents

# Chapter 1

# Todo List

**Global LCDcommand (char cmd)**

This routine delays 400us after sending the byte. Instead the LCD busy flag should be checked before sending the byte. All routines using delays, however, must follow this protocol

**Global LCDletter (char data)**

This routine delays 400us after sending the byte. Instead the LCD busy flag should be checked before sending the byte. All routines using delays, however, must follow this protocol

# Chapter 2

# File Index

## 2.1 File List

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

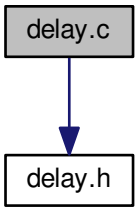
# Chapter 3

# File Documentation

## 3.1 delay.c File Reference

Routines used to provide delays for the LCD routines.

`#include "delay.h"` Include dependency graph for delay.c:




### Functions

- void Delay (unsigned int delay_count)

  *Delay for a specific count.*
- void Delay_Us (unsigned int delayUs_count)

  *Delay for a specified number of microseconds.*

### Variables

- unsigned int **temp_count**

### 3.1.1 Detailed Description

Routines used to provide delays for the LCD routines.

Definition in file delay.c.

### 3.1.2 Function Documentation

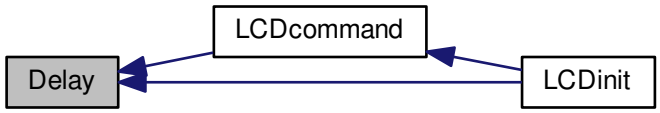
#### 3.1.2.1 void Delay ( unsigned int *delay_count* )

Delay for a specific count.

Definition at line 10 of file delay.c.

```
{
        temp_count = delay_count +1;
        asm volatile("outer: dec _temp_count");
        asm volatile("cp0 _temp_count");
        asm volatile("bra z, done");
        asm volatile("do #3200, inner" );
        asm volatile("nop");
        asm volatile("inner: nop");
        asm volatile("bra outer");
        asm volatile("done:");
}
```

Here is the caller graph for this function:



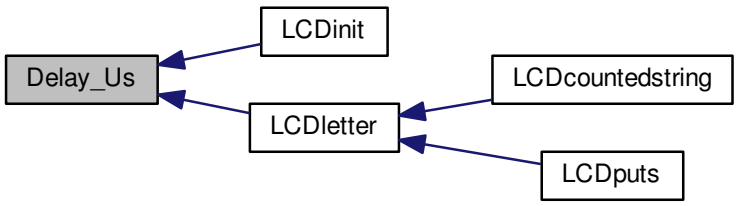

#### 3.1.2.2 void Delay_Us ( unsigned int *delayUs_count* )

Delay for a specified number of microseconds.

Definition at line 24 of file delay.c.

```
{
        temp_count = delayUs_count +1;
        asm volatile("outer1: dec _temp_count");
        asm volatile("cp0 _temp_count");
        asm volatile("bra z, done1");
        asm volatile("do #1500, inner1" );
        asm volatile("nop");
        asm volatile("inner1: nop");
        asm volatile("bra outer1");
        asm volatile("done1:");
}
```

Here is the caller graph for this function:




## 3.2 delay.c

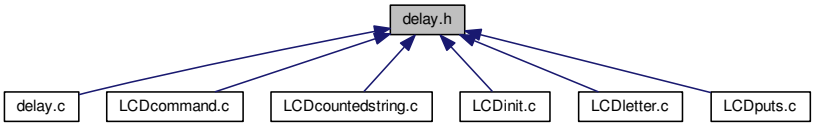
```
00001
00005 #include "delay.h"
00006
00007 unsigned int temp_count;
00008
00010 void Delay( unsigned int delay_count )
00011 {
00012         temp_count = delay_count +1;
00013         asm volatile("outer: dec _temp_count");
00014         asm volatile("cp0 _temp_count");
00015         asm volatile("bra z, done");
00016         asm volatile("do #3200, inner" );
00017         asm volatile("nop");
00018         asm volatile("inner: nop");
00019         asm volatile("bra outer");
00020         asm volatile("done:");
00021 }
00022
00024 void Delay_Us( unsigned int delayUs_count )
00025 {
00026         temp_count = delayUs_count +1;
00027         asm volatile("outer1: dec _temp_count");
```

```
00028         asm volatile("cp0 _temp_count");
00029         asm volatile("bra z, done1");
00030         asm volatile("do #1500, inner1" );
00031         asm volatile("nop");
00032         asm volatile("inner1: nop");
00033         asm volatile("bra outer1");
00034         asm volatile("done1:");
00035 }
00036
```

## 3.3 delay.h File Reference

Declarations for LCD delay routines.

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




### Defines

- #define Delay200uS_count (Fcy ∗ 0.0002) / 1080

  *Counts for a 200 us delay.*
- #define Delay_15mS_Cnt (Fcy ∗ 0.015) / 2950

  *Counts for a 15 ms delay.*
- #define Delay_1mS_Cnt (Fcy ∗ 0.001) / 2950

  *Counts for a 1 ms delay.*
- #define Delay_1S_Cnt (Fcy ∗ 1) / 2950

  *Counts for a 1 second delay.*
- #define Delay_2mS_Cnt (Fcy ∗ 0.002) / 2950

  *Counts for a 2 ms delay.*
- #define Delay_5mS_Cnt (Fcy ∗ 0.005) / 2950

  *Counts for a 5 ms delay.*
- #define Fcy 16000000

  *Instruction clock Hz.*

## Functions

- void Delay (unsigned int delay_count)
  *Delay for a specific count.*
- void Delay_Us (unsigned int delayUs_count)
  *Delay for a specified number of microseconds.*

### 3.3.1 Detailed Description

Declarations for LCD delay routines.

Definition in file delay.h.

### 3.3.2 Define Documentation

#### 3.3.2.1 #define Delay200uS_count (Fcy * 0.0002) / 1080

Counts for a 200 us delay.

Definition at line 15 of file delay.h.

#### 3.3.2.2 #define Delay_15mS_Cnt (Fcy * 0.015) / 2950

Counts for a 15 ms delay.

Definition at line 23 of file delay.h.

#### 3.3.2.3 #define Delay_1mS_Cnt (Fcy * 0.001) / 2950

Counts for a 1 ms delay.

Definition at line 17 of file delay.h.

#### 3.3.2.4 #define Delay_1S_Cnt (Fcy * 1) / 2950

Counts for a 1 second delay.

Definition at line 25 of file delay.h.

#### 3.3.2.5 #define Delay_2mS_Cnt (Fcy * 0.002) / 2950

Counts for a 2 ms delay.

Definition at line 19 of file delay.h.

#### 3.3.2.6 #define Delay_5mS_Cnt (Fcy * 0.005) / 2950

Counts for a 5 ms delay.

Definition at line 21 of file delay.h.

#### 3.3.2.7 #define Fcy 16000000

Instruction clock Hz.

Definition at line 7 of file delay.h.

### 3.3.3 Function Documentation

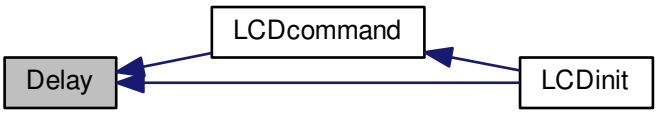
#### 3.3.3.1 void Delay ( unsigned int *delay_count* )

Delay for a specific count.

Definition at line 10 of file delay.c.

```
{
        temp_count = delay_count +1;
        asm volatile("outer: dec _temp_count");
        asm volatile("cp0 _temp_count");
        asm volatile("bra z, done");
        asm volatile("do #3200, inner" );
        asm volatile("nop");
        asm volatile("inner: nop");
        asm volatile("bra outer");
        asm volatile("done:");
}
```

Here is the caller graph for this function:



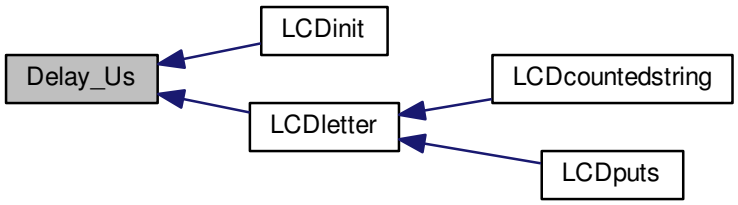
#### 3.3.3.2 void Delay_Us ( unsigned int *delayUs_count* )

Delay for a specified number of microseconds.

Definition at line 24 of file delay.c.

```
{
        temp_count = delayUs_count +1;
        asm volatile("outer1: dec _temp_count");
        asm volatile("cp0 _temp_count");
        asm volatile("bra z, done1");
        asm volatile("do #1500, inner1" );
        asm volatile("nop");
        asm volatile("inner1: nop");
        asm volatile("bra outer1");
        asm volatile("done1:");
}
```

Here is the caller graph for this function:




## 3.4 delay.h

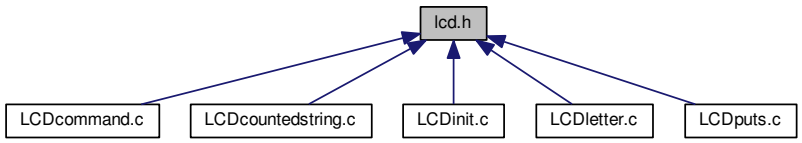
```
00001
00005 //#define Fcy  14754600
00007 #define Fcy  16000000
00008
00010 void Delay( unsigned int delay_count );
00012 void Delay_Us( unsigned int delayUs_count );
00013
00015 #define Delay200uS_count  (Fcy * 0.0002) / 1080
00016
00017 #define Delay_1mS_Cnt     (Fcy * 0.001) / 2950
00018
00019 #define Delay_2mS_Cnt     (Fcy * 0.002) / 2950
00020
00021 #define Delay_5mS_Cnt     (Fcy * 0.005) / 2950
00022
00023 #define Delay_15mS_Cnt    (Fcy * 0.015) / 2950
00024
```

```
00025 #define Delay_1S_Cnt      (Fcy * 1) / 2950
00026
```

## 3.5 lcd.h File Reference

LCD definitions.

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




### Defines

- #define LCDclear() LCDcommand( 0x01 )

  *Clear the LCD display and home cursor.*
- #define LCDhome() LCDcommand( 0x02 )

  *Set the LCD cursor to home.*
- #define LCDleft() LCDcommand( 0x10 )

  *Move the LCD cursor to the left.*
- #define LCDline2() LCDcommand( 0xC0 )

  *Position the LCD cursor to the second line.*
- #define LCDposition(a) LCDcommand( 0x80 + ( a & 0x7f) )

  *Set the LCD cursor position.*
- #define LCDright() LCDcommand( 0x14 )

  *Move the LCD cursor to the right.*
- #define LCDshift() LCDcommand( 0x1C )

  *Shift the LCD display.*

### Functions

- void LCDcommand (char cmd)

  *Send a command to the LCD.*

- void LCDcountedstring (unsigned char *data, unsigned char count)

  *Send a counted string to the LCD.*
- void LCDinit (void)

  *Initialize the LCD.*
- void LCDletter (char data)

  *Send a character to the LCD.*
- void LCDputs (char *)

  *Send a string to the LCD.*

### 3.5.1 Detailed Description

LCD definitions.

Definition in file lcd.h.

### 3.5.2 Define Documentation

#### 3.5.2.1 #define LCDclear( ) LCDcommand( 0x01 )

Clear the LCD display and home cursor.

Definition at line 27 of file lcd.h.

#### 3.5.2.2 #define LCDhome( ) LCDcommand( 0x02 )

Set the LCD cursor to home.

Definition at line 29 of file lcd.h.

#### 3.5.2.3 #define LCDleft( ) LCDcommand( 0x10 )

Move the LCD cursor to the left.

Definition at line 23 of file lcd.h.

#### 3.5.2.4 #define LCDline2( ) LCDcommand( 0xC0 )

Position the LCD cursor to the second line.

Definition at line 31 of file lcd.h.

#### 3.5.2.5 #define LCDposition( *a* ) LCDcommand( 0x80 + ( a & 0x7f) )

Set the LCD cursor position.

Definition at line 33 of file lcd.h.

#### 3.5.2.6 #define LCDright( ) LCDcommand( 0x14 )

Move the LCD cursor to the right.

Definition at line 21 of file lcd.h.

#### 3.5.2.7 #define LCDshift( ) LCDcommand( 0x1C )

Shift the LCD display.

Definition at line 25 of file lcd.h.

### 3.5.3 Function Documentation

#### 3.5.3.1 void LCDcommand ( char *cmd* )

Send a command to the LCD.

This routine simple sends a data byte to the LCD. The register select pin is set to 0 notifying the LCD that the byte is to be interpreted as a command.

**Parameters**

| | |
|---|---|
| *cmd* | char - Command byte to send to LCD |

**Returns**

none

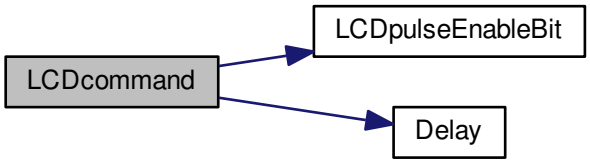
**Todo** This routine delays 400us after sending the byte. Instead the LCD busy flag should be checked before sending the byte. All routines using delays, however, must follow this protocol

Definition at line 43 of file LCDcommand.c.

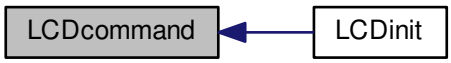
```
{
    LCD_DATA &= 0xFF00; // prepare RD0 - RD7
    LCD_DATA |= cmd; // command byte to lcd
    LCD_RW = 0; // ensure RW is 0
    LCD_RS = 0;
    LCDpulseEnableBit();
```

```
    Delay(Delay_5mS_Cnt); // 5ms delay
}
```

Here is the call graph for this function:




Here is the caller graph for this function:




#### 3.5.3.2 void LCDcountedstring ( unsigned char * *data,* unsigned char *count* )

Send a counted string to the LCD.

In C, strings are always terminated with a null character, so there is no need to count characters. However, it is possible to send something else to the LCD, thinking it is a string when in fact, it is not terminated. Therefore, in embedded applications, it is safer to count characters so the string display is guaranteed to complete no matter what the data.

**Parameters**

| | |
|---|---|
| *data* | unsigned char * - pointing to the string to be displayed |
| *count* | int - count of number of characters to send to LCD |

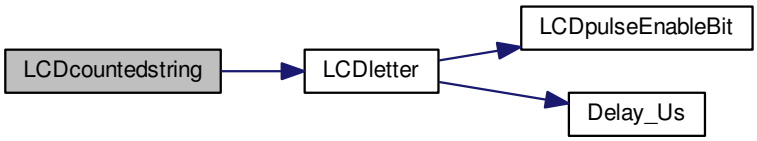
**Returns**

none

Definition at line 44 of file LCDcountedstring.c.

```
{
    while (count)
    {
        LCDletter(*data++);
        count--;
    }
}
```

Here is the call graph for this function:




#### 3.5.3.3 void LCDinit ( void )

Initialize the LCD.

LCDinit() first delays 15ms to allow the LCD internals to finish responding to power on. This is not always necessary, but typically only happens once in a program. The LCD initialization sequence is then sent, setting the LCD to bit data.

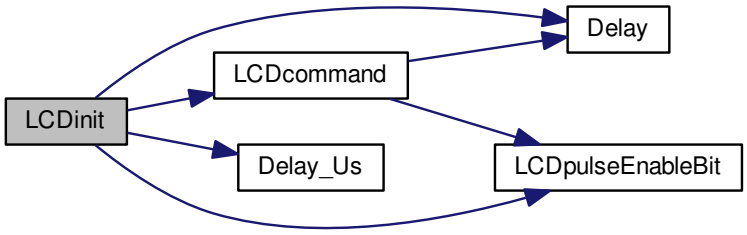
LCD options are then sent, LCD 2 line, 5x7 font, entry mode to not shift, display on, cursor off.

Definition at line 40 of file LCDinit.c.

```
                {
// 15mS delay after Vdd reaches nnVdc before proceeding with LCD
   initialization
// not always required and is based on system Vdd rise rate
Delay(Delay_15mS_Cnt); // 15ms delay

/* set initial states for the data and control pins */
LCD_DATA &= 0xFF00;
LCD_RW = 0; // R/W state set low
LCD_RS = 0; // RS state set low
```

```
    LCD_ENABLE = 0; // E state set low

    /* set data and control pins to outputs */
    LCD_DATATRIS &= 0xFF00;
    LCD_RW_TRIS = 0; // RW pin set as output
    LCD_RS_TRIS = 0; // RS pin set as output
    LCD_ENABLE_TRIS = 0; // E pin set as output

    /* 1st LCD initialization sequence */
    LCD_DATA &= 0xFF00;
    LCD_DATA |= 0x0038;
    LCDpulseEnableBit();
    Delay(Delay_5mS_Cnt); // 5ms delay

    /* 2nd LCD initialization sequence */
    LCD_DATA &= 0xFF00;
    LCD_DATA |= 0x0038;
    LCDpulseEnableBit();
    Delay_Us(Delay200uS_count); // 200uS delay

    /* 3rd LCD initialization sequence */
    LCD_DATA &= 0xFF00;
    LCD_DATA |= 0x0038;
    LCDpulseEnableBit();
    Delay_Us(Delay200uS_count); // 200uS delay

    // Establish the LCD options
    // LCD_FUN_SET | DL_8 | 2_LINE _ 5x7_FONT
    LCDcommand(0x38); // function set
    // LCD_DISPLAY | DISP_ON | CURS_OFF | BLINK_OFF
    LCDcommand(0x0C); // Display on/off control, cursor blink off (0x0C)
    // LCD_ENTRY_MODE | DIC_INCR | NO_SHIFT
    LCDcommand(0x06); // entry mode set (0x06)
}
```

Here is the call graph for this function:




#### 3.5.3.4 void LCDletter ( char *data* )

Send a character to the LCD.

This routine simply sends a data byte to the LCD. The register select pin is set to 1 notifying the LCD that the byte is to be used as a displayed character.

**Parameters**

| | |
|---|---|
| *data* | char - Character to send to the LCD |

**Returns**

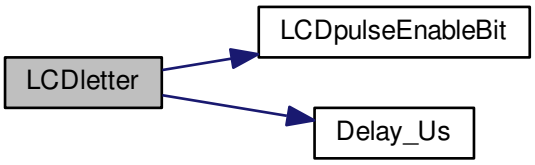
none

**Todo** This routine delays 400us after sending the byte. Instead the LCD busy flag should be checked before sending the byte. All routines using delays, however, must follow this protocol

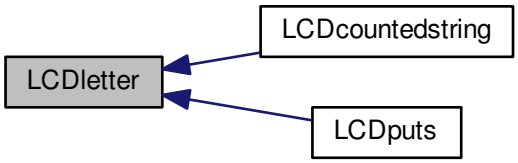
Definition at line 43 of file LCDletter.c.

```
{
    LCD_RW = 0; // ensure RW is 0
    LCD_RS = 1; // assert register select to 1
    LCD_DATA &= 0xFF00; // prepare RD0 - RD7
    LCD_DATA |= data; // data byte to lcd
    LCDpulseEnableBit();
    LCD_RS = 0; // negate register select to 0
    Delay_Us(Delay200uS_count); // 200uS delay
    Delay_Us(Delay200uS_count); // 200uS delay
}
```

Here is the call graph for this function:



Here is the caller graph for this function:




#### 3.5.3.5 void LCDputs ( char * p )

Send a string to the LCD.

Sends a null-terminated string to the LCD

**Parameters**

| | |
|---|---|
| *p* | char * - pointer to string to be displayed |

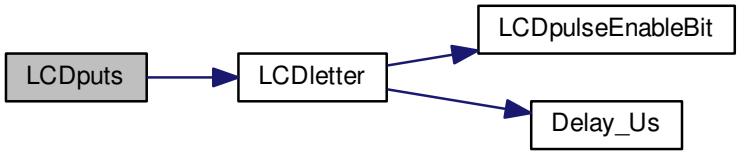
**Returns**

none

Definition at line 37 of file LCDputs.c.

```
{
    while (*p)
    {
        LCDletter(*p);
        p++;
    }
}
```

Here is the call graph for this function:



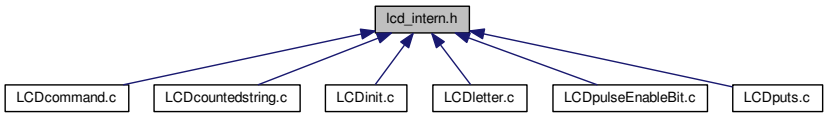

## 3.6 lcd.h

```
00001
00006 /****** LCD FUNCTION PROTOYPES ******/
00007
00009 void LCDinit( void );                    // initialize display
00011 void LCDcommand( char cmd );              // write command to lcd
00013 void LCDletter( char data );                 // write data to lcd
00015 void LCDcountedstring ( unsigned char *data, unsigned char count );
00017 void LCDputs( char * );
00018
00019 /*****  LCD COMMAND FUCNTION PROTOTYPES  *****/
00021 #define LCDright()      LCDcommand( 0x14 )
00022
00023 #define LCDleft()       LCDcommand( 0x10 )
00024
00025 #define LCDshift()      LCDcommand( 0x1C )
00026
00027 #define LCDclear()      LCDcommand( 0x01 )
00028
00029 #define LCDhome()       LCDcommand( 0x02 )
00030
00031 #define LCDline2()      LCDcommand( 0xC0 ) // (0xC0)
00032
00033 #define LCDposition(a)  LCDcommand( 0x80 + ( a & 0x7f) )
```

## 3.7 lcd_intern.h File Reference

Definitions used within LCD routines.

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




## Defines

- #define LCD_DATA LATE

  *LCD data port latch.*
- #define LCD_DATAPORT PORTE

  *LCD data port.*
- #define LCD_DATATRIS TRISE

  *LCD data port direction register.*
- #define LCD_ENABLE LATDbits.LATD4

  *LCD Enable pin.*
- #define LCD_ENABLE_TRIS TRISDbits.TRISD4

  *LCD Enable direction register bit.*
- #define LCD_RS LATBbits.LATB15

  *LCD Register select pin.*
- #define LCD_RS_TRIS TRISBbits.TRISB15

  *LCD Register select direction register bit.*
- #define LCD_RW LATDbits.LATD5

  *LCD Read/Write pin.*
- #define LCD_RW_TRIS TRISDbits.TRISD5

  *LCD Read/Write direction register bit.*

## Functions

- void LCDpulseEnableBit (void)

  *Toggle the LCD enable bit.*

### 3.7.1 Detailed Description

Definitions used within LCD routines. This file contains definitions of the various connections to the LCD on the Explorer 16 board. They are uninteresting outside the LCD routines.

Definition in file lcd_intern.h.

### 3.7.2 Define Documentation

#### 3.7.2.1 #define LCD_DATA LATE

LCD data port latch.

Definition at line 49 of file lcd_intern.h.

#### 3.7.2.2 #define LCD_DATAPORT PORTE

LCD data port.

Definition at line 51 of file lcd_intern.h.

#### 3.7.2.3 #define LCD_DATATRIS TRISE

LCD data port direction register.

Definition at line 53 of file lcd_intern.h.

#### 3.7.2.4 #define LCD_ENABLE LATDbits.LATD4

LCD Enable pin.

Definition at line 37 of file lcd_intern.h.

#### 3.7.2.5 #define LCD_ENABLE_TRIS TRISDbits.TRISD4

LCD Enable direction register bit.

Definition at line 45 of file lcd_intern.h.

#### 3.7.2.6 #define LCD_RS LATBbits.LATB15

LCD Register select pin.

Definition at line 35 of file lcd_intern.h.

#### 3.7.2.7 #define LCD_RS_TRIS TRISBbits.TRISB15

LCD Register select direction register bit.

Definition at line 43 of file lcd_intern.h.

#### 3.7.2.8 #define LCD_RW LATDbits.LATD5

LCD Read/Write pin.

Definition at line 33 of file lcd_intern.h.

#### 3.7.2.9 #define LCD_RW_TRIS TRISDbits.TRISD5

LCD Read/Write direction register bit.

Definition at line 41 of file lcd_intern.h.

### 3.7.3 Function Documentation

#### 3.7.3.1 void LCDpulseEnableBit ( void )

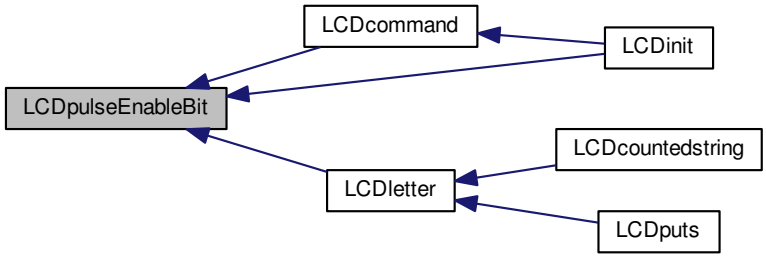
Toggle the LCD enable bit.

Each LCD command is strobed into the device by raising the enable bit for at least 40 microseconds. This routine provides this function to the other functions in the library.

Definition at line 35 of file LCDpulseEnableBit.c.

```
{
    LCD_ENABLE = 1;
    Nop();
    Nop();
    Nop();
    LCD_ENABLE = 0; // toggle E signal

}
```

Here is the caller graph for this function:




## 3.8 lcd_intern.h

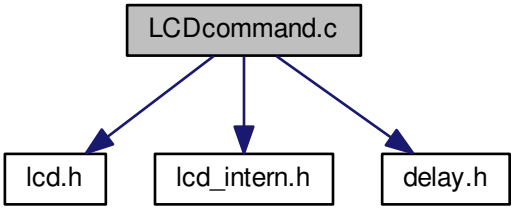
```
00001
00009 /*
00010  * File:   lcd_intern.h
00011  * Author: jjmcd
00012  *
00013  * Created on June 19, 2012, 12:57 PM
00014  */
00015
00016 #ifndef LCD_INTERN_H
00017 #define LCD_INTERN_H
00018
00019 #ifdef  __cplusplus
00020 extern "C" {
00021 #endif
00022
00023 /*
00024    For Explorer 16 board, here are the data and control signal definitions
00025    RS -> RB15
00026    E  -> RD4
00027    LCD_RW -> RD5
00028    DATA -> RE0 - RE7
00029 */
00030
00031 // Control signal data pins
00033 #define  LCD_RW         LATDbits.LATD5
00034
00035 #define  LCD_RS         LATBbits.LATB15
00036
00037 #define  LCD_ENABLE     LATDbits.LATD4
00038
00039 // Control signal pin direction
00041 #define  LCD_RW_TRIS        TRISDbits.TRISD5
00042
00043 #define  LCD_RS_TRIS        TRISBbits.TRISB15
00044
00045 #define  LCD_ENABLE_TRIS    TRISDbits.TRISD4
00046
```

```
00047 // Data signals and pin direction
00049 #define  LCD_DATA      LATE
00050
00051 #define  LCD_DATAPORT  PORTE
00052
00053 #define  LCD_DATATRIS  TRISE
00054
00056 void LCDpulseEnableBit( void );
00057
00058 #ifdef  __cplusplus
00059 }
00060 #endif
00061
00062 #endif  /* LCD_INTERN_H */
00063
```

## 3.9 LCDcommand.c File Reference

Send a command to the LCD.

`#include "lcd.h" #include "lcd_intern.h" #include "delay.-h"` Include dependency graph for LCDcommand.c:




### Functions

- void LCDcommand (char cmd)

  *Send a command to the LCD.*

### 3.9.1 Detailed Description

Send a command to the LCD.

Definition in file LCDcommand.c.

### 3.9.2 Function Documentation

#### 3.9.2.1 void LCDcommand ( char *cmd* )

Send a command to the LCD.

This routine simple sends a data byte to the LCD. The register select pin is set to 0 notifying the LCD that the byte is to be interpreted as a command.

**Parameters**

| | |
|---|---|
| *cmd* | char - Command byte to send to LCD |

**Returns**

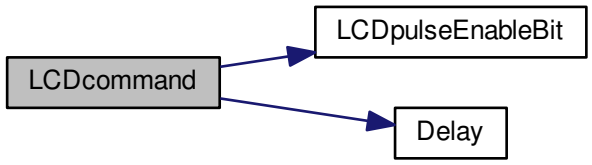
none

**Todo** This routine delays 400us after sending the byte. Instead the LCD busy flag should be checked before sending the byte. All routines using delays, however, must follow this protocol

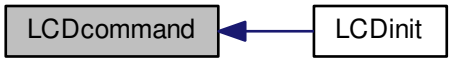
Definition at line 43 of file LCDcommand.c.

```
{
    LCD_DATA &= 0xFF00; // prepare RD0 - RD7
    LCD_DATA |= cmd; // command byte to lcd
    LCD_RW = 0; // ensure RW is 0
    LCD_RS = 0;
    LCDpulseEnableBit();
    Delay(Delay_5mS_Cnt); // 5ms delay
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



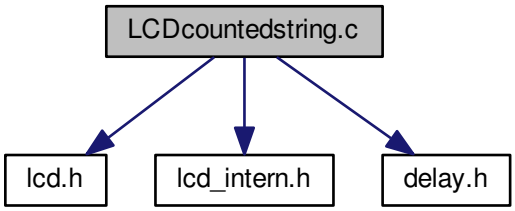

## 3.10 LCDcommand.c

```
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 #include "lcd.h"
00028 #include "lcd_intern.h"
00029 #include "delay.h"
00030
00032
00043 void LCDcommand( char cmd )
00044 {
00045     LCD_DATA &= 0xFF00; // prepare RD0 - RD7
00046     LCD_DATA |= cmd; // command byte to lcd
00047     LCD_RW = 0; // ensure RW is 0
00048     LCD_RS = 0;
00049     LCDpulseEnableBit();
00050     Delay(Delay_5mS_Cnt); // 5ms delay
00051 }
00052
```

## 3.11 LCDcountedstring.c File Reference

Send a specific number of characters to the LCD.

`#include "lcd.h" #include "lcd_intern.h" #include "delay.-h"` Include dependency graph for LCDcountedstring.c:




## Functions

- void LCDcountedstring (unsigned char *data, unsigned char count)

  *Send a counted string to the LCD.*

### 3.11.1 Detailed Description

Send a specific number of characters to the LCD.

Definition in file LCDcountedstring.c.

### 3.11.2 Function Documentation

#### 3.11.2.1 void LCDcountedstring ( unsigned char * *data*, unsigned char *count* )

Send a counted string to the LCD.

In C, strings are always terminated with a null character, so there is no need to count characters. However, it is possible to send something else to the LCD, thinking it is a string when in fact, it is not terminated. Therefore, in embedded applications, it is safer to count characters so the string display is guaranteed to complete no matter what the data.

**Parameters**

| | |
|---|---|
| *data* | unsigned char * - pointing to the string to be displayed |
| *count* | int - count of number of characters to send to LCD |

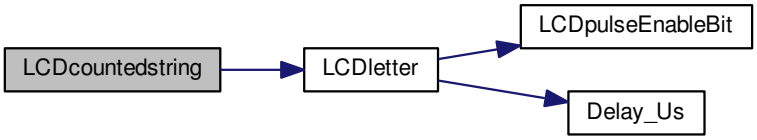
**Returns**

none

Definition at line 44 of file LCDcountedstring.c.

```
{
    while (count)
    {
        LCDletter(*data++);
        count--;
    }
}
```

Here is the call graph for this function:




## 3.12 LCDcountedstring.c

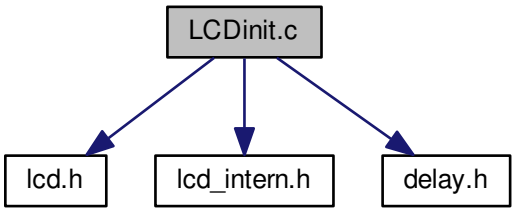
```
00001
00007 #if defined(__PIC24E__)
00008 #include <p24Exxxx.h>
00009
00010 #elif defined (__PIC24F__)
00011 #include <p24Fxxxx.h>
00012
00013 #elif defined(__PIC24H__)
00014 #include <p24Hxxxx.h>
00015
00016 #elif defined(__dsPIC30F__)
00017 #include <p30Fxxxx.h>
00018
00019 #elif defined (__dsPIC33E__)
00020 #include <p33Exxxx.h>
00021
00022 #elif defined(__dsPIC33F__)
00023 #include <p33Fxxxx.h>
00024
00025 #endif
00026
00027 #include "lcd.h"
00028 #include "lcd_intern.h"
00029 #include "delay.h"
00030
```

```
00031
00033
00044 void LCDcountedstring( unsigned char *data, unsigned char count)
00045 {
00046     while (count)
00047     {
00048         LCDletter(*data++);
00049         count--;
00050     }
00051 }
```

## 3.13 LCDinit.c File Reference

Initialize the LCD.

`#include "lcd.h" #include "lcd_intern.h" #include "delay.-h"` Include dependency graph for LCDinit.c:




### Functions

- void LCDinit (void)

  *Initialize the LCD.*

### 3.13.1 Detailed Description

Initialize the LCD.

Definition in file LCDinit.c.

### 3.13.2 Function Documentation

#### 3.13.2.1 void LCDinit ( void )

Initialize the LCD.

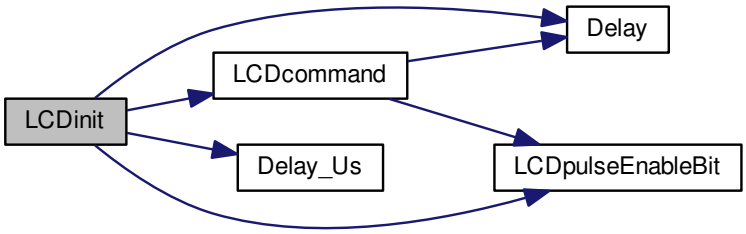
LCDinit() first delays 15ms to allow the LCD internals to finish responding to power on. This is not always necessary, but typically only happens once in a program. The LCD initialization sequence is then sent, setting the LCD to bit data.

LCD options are then sent, LCD 2 line, 5x7 font, entry mode to not shift, display on, cursor off.

Definition at line 40 of file LCDinit.c.

```
                    {
    // 15mS delay after Vdd reaches nnVdc before proceeding with LCD
       initialization
    // not always required and is based on system Vdd rise rate
    Delay(Delay_15mS_Cnt); // 15ms delay

    /* set initial states for the data and control pins */
    LCD_DATA &= 0xFF00;
    LCD_RW = 0; // R/W state set low
    LCD_RS = 0; // RS state set low
    LCD_ENABLE = 0; // E state set low

    /* set data and control pins to outputs */
    LCD_DATATRIS &= 0xFF00;
    LCD_RW_TRIS = 0; // RW pin set as output
    LCD_RS_TRIS = 0; // RS pin set as output
    LCD_ENABLE_TRIS = 0; // E pin set as output

    /* 1st LCD initialization sequence */
    LCD_DATA &= 0xFF00;
    LCD_DATA |= 0x0038;
    LCDpulseEnableBit();
    Delay(Delay_5mS_Cnt); // 5ms delay

    /* 2nd LCD initialization sequence */
    LCD_DATA &= 0xFF00;
    LCD_DATA |= 0x0038;
    LCDpulseEnableBit();
    Delay_Us(Delay200uS_count); // 200uS delay

    /* 3rd LCD initialization sequence */
    LCD_DATA &= 0xFF00;
    LCD_DATA |= 0x0038;
    LCDpulseEnableBit();
    Delay_Us(Delay200uS_count); // 200uS delay

    // Establish the LCD options
    // LCD_FUN_SET | DL_8 | 2_LINE _ 5x7_FONT
    LCDcommand(0x38); // function set
    // LCD_DISPLAY | DISP_ON | CURS_OFF | BLINK_OFF
    LCDcommand(0x0C); // Display on/off control, cursor blink off (0x0C)
    // LCD_ENTRY_MODE | DIC_INCR | NO_SHIFT
    LCDcommand(0x06); // entry mode set (0x06)
}
```

Here is the call graph for this function:




## 3.14 LCDinit.c

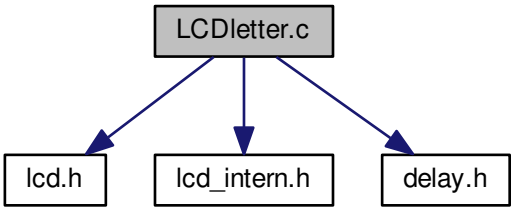
```
00001
00007 #if defined(__PIC24E__)
00008 #include <p24Exxxx.h>
00009
00010 #elif defined (__PIC24F__)
00011 #include <p24Fxxxx.h>
00012
00013 #elif defined(__PIC24H__)
00014 #include <p24Hxxxx.h>
00015
00016 #elif defined(__dsPIC30F__)
00017 #include <p30Fxxxx.h>
00018
00019 #elif defined (__dsPIC33E__)
00020 #include <p33Exxxx.h>
00021
00022 #elif defined(__dsPIC33F__)
00023 #include <p33Fxxxx.h>
00024
00025 #endif
00026
00027 #include "lcd.h"
00028 #include "lcd_intern.h"
00029 #include "delay.h"
00030
00032
00040 void LCDinit( void) {
00041     // 15mS delay after Vdd reaches nnVdc before proceeding with LCD
      initialization
00042     // not always required and is based on system Vdd rise rate
00043     Delay(Delay_15mS_Cnt); // 15ms delay
00044
00045     /* set initial states for the data and control pins */
00046     LCD_DATA &= 0xFF00;
00047     LCD_RW = 0; // R/W state set low
00048     LCD_RS = 0; // RS state set low
00049     LCD_ENABLE = 0; // E state set low
```

```
00050
00051     /* set data and control pins to outputs */
00052     LCD_DATATRIS &= 0xFF00;
00053     LCD_RW_TRIS = 0; // RW pin set as output
00054     LCD_RS_TRIS = 0; // RS pin set as output
00055     LCD_ENABLE_TRIS = 0; // E pin set as output
00056
00057     /* 1st LCD initialization sequence */
00058     LCD_DATA &= 0xFF00;
00059     LCD_DATA |= 0x0038;
00060     LCDpulseEnableBit();
00061     Delay(Delay_5mS_Cnt); // 5ms delay
00062
00063     /* 2nd LCD initialization sequence */
00064     LCD_DATA &= 0xFF00;
00065     LCD_DATA |= 0x0038;
00066     LCDpulseEnableBit();
00067     Delay_Us(Delay200uS_count); // 200uS delay
00068
00069     /* 3rd LCD initialization sequence */
00070     LCD_DATA &= 0xFF00;
00071     LCD_DATA |= 0x0038;
00072     LCDpulseEnableBit();
00073     Delay_Us(Delay200uS_count); // 200uS delay
00074
00075     // Establish the LCD options
00076     // LCD_FUN_SET | DL_8 | 2_LINE _ 5x7_FONT
00077     LCDcommand(0x38); // function set
00078     // LCD_DISPLAY | DISP_ON | CURS_OFF | BLINK_OFF
00079     LCDcommand(0x0C); // Display on/off control, cursor blink off (0x0C)
00080     // LCD_ENTRY_MODE | DIC_INCR | NO_SHIFT
00081     LCDcommand(0x06); // entry mode set (0x06)
00082 }
```

## 3.15 LCDletter.c File Reference

Send a character to the LCD.

`#include "lcd.h" #include "lcd_intern.h" #include "delay.-h"` Include dependency graph for LCDletter.c:



## Functions

- void LCDletter (char data)

  *Send a character to the LCD.*

### 3.15.1 Detailed Description

Send a character to the LCD.

Definition in file LCDletter.c.

### 3.15.2 Function Documentation

#### 3.15.2.1 void LCDletter ( char *data* )

Send a character to the LCD.

This routine simply sends a data byte to the LCD. The register select pin is set to 1 notifying the LCD that the byte is to be used as a displayed character.

**Parameters**

| | |
|---|---|
| *data* | char - Character to send to the LCD |

**Returns**

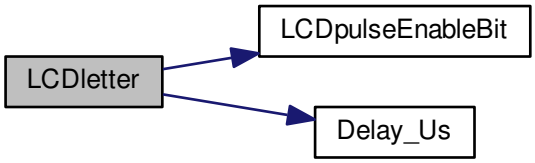
none

**Todo** This routine delays 400us after sending the byte. Instead the LCD busy flag should be checked before sending the byte. All routines using delays, however, must follow this protocol

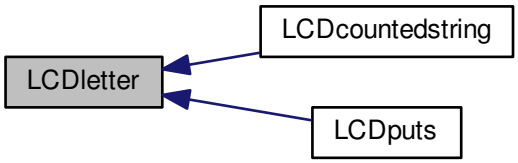
Definition at line 43 of file LCDletter.c.

```
{
    LCD_RW = 0; // ensure RW is 0
    LCD_RS = 1; // assert register select to 1
    LCD_DATA &= 0xFF00; // prepare RD0 - RD7
    LCD_DATA |= data; // data byte to lcd
    LCDpulseEnableBit();
    LCD_RS = 0; // negate register select to 0
    Delay_Us(Delay200uS_count); // 200uS delay
    Delay_Us(Delay200uS_count); // 200uS delay
}
```

Here is the call graph for this function:




Here is the caller graph for this function:




## 3.16 LCDletter.c

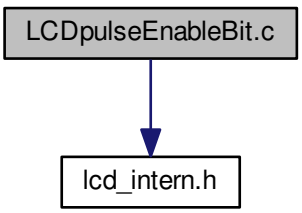
```
00001
00007 #if defined(__PIC24E__)
00008 #include <p24Exxxx.h>
00009
00010 #elif defined (__PIC24F__)
00011 #include <p24Fxxxx.h>
00012
00013 #elif defined(__PIC24H__)
00014 #include <p24Hxxxx.h>
00015
00016 #elif defined(__dsPIC30F__)
00017 #include <p30Fxxxx.h>
00018
00019 #elif defined (__dsPIC33E__)
00020 #include <p33Exxxx.h>
00021
```

```
00022 #elif defined(__dsPIC33F__)
00023 #include <p33Fxxxx.h>
00024
00025 #endif
00026
00027 #include "lcd.h"
00028 #include "lcd_intern.h"
00029 #include "delay.h"
00030
00031
00033
00043 void LCDletter( char data )
00044 {
00045     LCD_RW = 0; // ensure RW is 0
00046     LCD_RS = 1; // assert register select to 1
00047     LCD_DATA &= 0xFF00; // prepare RD0 - RD7
00048     LCD_DATA |= data; // data byte to lcd
00049     LCDpulseEnableBit();
00050     LCD_RS = 0; // negate register select to 0
00051     Delay_Us(Delay200uS_count); // 200uS delay
00052     Delay_Us(Delay200uS_count); // 200uS delay
00053 }
```

## 3.17 LCDpulseEnableBit.c File Reference

Pulse the LCD enable bit for long enough.

`#include "lcd_intern.h"` Include dependency graph for LCDpulseEnableBit.c:




### Functions

- void LCDpulseEnableBit (void)

  *Toggle the LCD enable bit.*

### 3.17.1 Detailed Description

Pulse the LCD enable bit for long enough.

Definition in file LCDpulseEnableBit.c.

### 3.17.2 Function Documentation

#### 3.17.2.1 void LCDpulseEnableBit ( void )

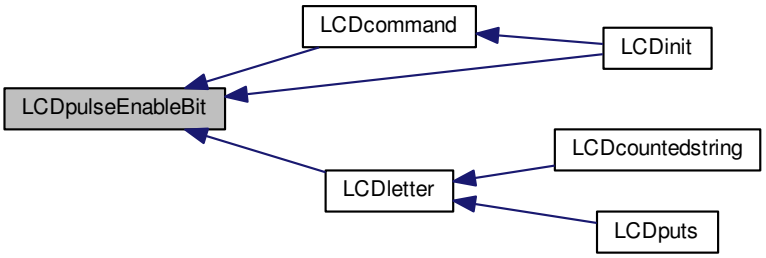
Toggle the LCD enable bit.

Each LCD command is strobed into the device by raising the enable bit for at least 40 microseconds. This routine provides this function to the other functions in the library.

Definition at line 35 of file LCDpulseEnableBit.c.

```
{
    LCD_ENABLE = 1;
    Nop();
    Nop();
    Nop();
    LCD_ENABLE = 0; // toggle E signal

}
```

Here is the caller graph for this function:




## 3.18 LCDpulseEnableBit.c

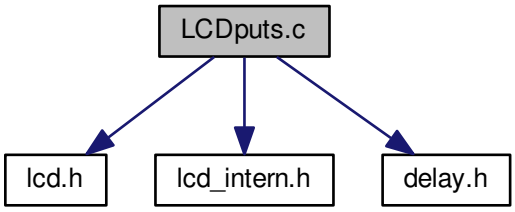
```
00001
00007 #if defined(__PIC24E__)
```

```
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 #include "lcd_intern.h"
00028
00030
00035 void LCDpulseEnableBit( void )
00036 {
00037     LCD_ENABLE = 1;
00038     Nop();
00039     Nop();
00040     Nop();
00041     LCD_ENABLE = 0; // toggle E signal
00042
00043 }
```

## 3.19 LCDputs.c File Reference

Put a string to the LCD.

`#include "lcd.h" #include "lcd_intern.h" #include "delay.-h"` Include dependency graph for LCDputs.c:



## Functions

- void LCDputs (char *p)

  *Send a string to the LCD.*

### 3.19.1 Detailed Description

Put a string to the LCD.

Definition in file LCDputs.c.

### 3.19.2 Function Documentation

#### 3.19.2.1 void LCDputs ( char * p )

Send a string to the LCD.

Sends a null-terminated string to the LCD

**Parameters**

| | |
|---|---|
| *p* | char * - pointer to string to be displayed |

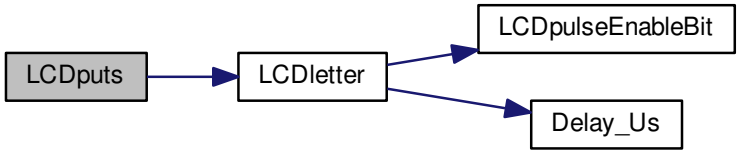
**Returns**

none

Definition at line 37 of file LCDputs.c.

```
{
    while (*p)
    {
        LCDletter(*p);
        p++;
    }
}
```

Here is the call graph for this function:




## 3.20 LCDputs.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 #include "lcd.h"
00028 #include "lcd_intern.h"
00029 #include "delay.h"
00030
00031
00033
00037 void LCDputs( char *p )
00038 {
00039     while (*p)
00040     {
00041         LCDletter(*p);
00042         p++;
00043     }
00044 }
```