

Green highlight is the necessary parts for handshaking

Yellow highlight is the section for error display

```
#include <main.h>
```

```
#include <2408.C>
```

```
#define LCD_ENABLE_PIN PIN_B2
```

```
#define LCD_RS_PIN PIN_B0
```

```
#define LCD_RW_PIN PIN_B1
```

```
#define LCD_DATA4 PIN_B4
```

```
#define LCD_DATA5 PIN_B5
```

```
#define LCD_DATA6 PIN_B6
```

```
#define LCD_DATA7 PIN_B7
```

```
#define HS_PIN_B3
```

```
#define mem1 0x00
```

```
#define mem2 0x05
```

```
// error checking set to 0 for normal ops and 1 to generate error
```

```
#define error 0
```

```
unsigned int read_buttons(void);
```

```
void write_to_our_mem(unsigned int,unsigned int);
```

```
unsigned int read_from_our_mem(unsigned int);
```

```
#include <lcd.c>
```

```
void main()
```

```
{
```

```
    setup_adc_ports(NO_ANALOGS, VSS_VDD);
```

```
    init_ext_eeprom();
```

```
    unsigned int value,localdata=0,old_localdata=0;
```

```
    unsigned int returned_data=0;
```

```
    lcd_init();
```

```
// initialise lcd
```

```
    while(TRUE)
```

```
// program main loop start
```

```
    {
```

```
        value=0;
```

```
        value=read_buttons();
```

```
        switch(value)
```

```
        {
```

```
            case 0x01: if(!(localdata & 0x01))localdata+=1;
```

```
// button = 1      add one to local data
```

```
                else localdata=localdata & 0xfe;
```

```
// if toggled off AND local data with 0xFE
```

```
                write_to_our_mem(mem1,localdata);
```

```
// write data to mem1 location
```

```
                printf(lcd_putc,"\f value = %x",localdata);
```

```
// display localdata on line 1
```

```
                break;
```

```

case 0x02: if(!localdata & 0x02))localdata+=2;           // button = 2      add two to local data
            else localdata=localdata & 0xfd;             // if toggled off AND local data with 0xfd
            write_to_our_mem(mem1,localdata);            // write data to mem1 location
            printf(lcd_putc,"\f value = %x",localdata);   // display localdata on line 1
            break;

case 0x04: if(!localdata & 0x04))localdata+=4;           // button = 4      add 4 to local data
            else localdata=localdata & 0xfb;             // if toggled off AND local data with 0xfb
            write_to_our_mem(mem1,localdata);            // write data to mem1 location
            printf(lcd_putc,"\f value = %x",localdata);   // display localdata on line 1
            break;

case 0x08: if(!localdata & 0x08))localdata+=8;           // button = 8      add 8 to local data
            else localdata=localdata & 0xf7;             // if toggled off AND local data with 0xf7
            write_to_our_mem(mem1,localdata);            // write data to mem1 location
            printf(lcd_putc,"\f value = %x",localdata);   // display localdata on line 1
            break;
    }

    if(old_localdata!=localdata)
    {
        delay_ms(200);                                   // wait 50mS
        returned_data=read_from_our_mem(mem2);            // get the data stored in mem2
    }

```

```

returned_data+=error;          //introduce error if defined
lcd_gotoxy(1,2);              // goto second line
printf(lcd_putc," reply = %x",returned_data); // print the reply data on line 2
if(returned_data!=localdata)  // if data in mem2 differs from localdata
{
    delay_ms(100);             // wait for 100ms to change the display
    lcd_gotoxy(1,2);           // cursor to second line
    printf(lcd_putc," reply = %x ERROR",returned_data); // print the reply data on line 2 with the word error
}

```

```

old_localdata=localdata;      // move localdata into old_localdata

```

```

}

```

```

}

```

```

// program main loop end

```

```

}

```

```

unsigned int read_buttons(void)

```

```

{

```

```

    unsigned int value=0;

```

```

    value=input_a();

```

```

// read button value (port A)

```

```

    if(value)

```

```

// if port A > 0

```

```

    {

```

```

        while(input_a());

```

```

// wait for button to be released

```

```

    }

```

```
return(value);                                     // return button value
}
```

```
void write_to_our_mem(unsigned int address,unsigned int data)
```

```
{
```

```
    while(!input(HS));                             // check that the HS line is high
```

```
    output_low(HS);                                 // pull HS line low (we control the memory)
```

```
    i2c_start();                                    // send i2c start pulse
```

```
    i2c_write(0xA0);                                // send chip type and identity
```

```
    i2c_write(address);                             // write chip internal address
```

```
    i2c_write(data);                                // write data to internal address
```

```
    i2c_stop();                                      // send stop pulse.
```

```
    delay_ms(20);
```

```
    input(HS);                                       // turn the hs line to input (high resistance)
```

```
}
```

```
unsigned int read_from_our_mem(unsigned int address)
{
    unsigned int val;

    while(!input(HS)); // check that the HS line is high
    output_low(HS); // pull HS line low (we control the memory)
    i2c_start(); // issue i2c start pulse
    i2c_write(0xA0); // issue chip type and number
    i2c_write(address); // send address to be read from
    i2c_start(); // issue another start pulse
    i2c_write(0xA1); // issue chip type and number read command
    val=i2c_read(); // read data from the chip
    i2c_stop();
    delay_ms(20);
    input(HS); // turn the hs line to input (high resistance)
    return(val);
}
```