



DesignNews

PIC Microcontroller Embedded Development Using the CCS PIC MCU C Compiler

Day 5:

Coding a Winbond SpiFlash Driver Using CCS C

Sponsored by



Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Attendee Chat’ by maximizing the chat widget in your dock.

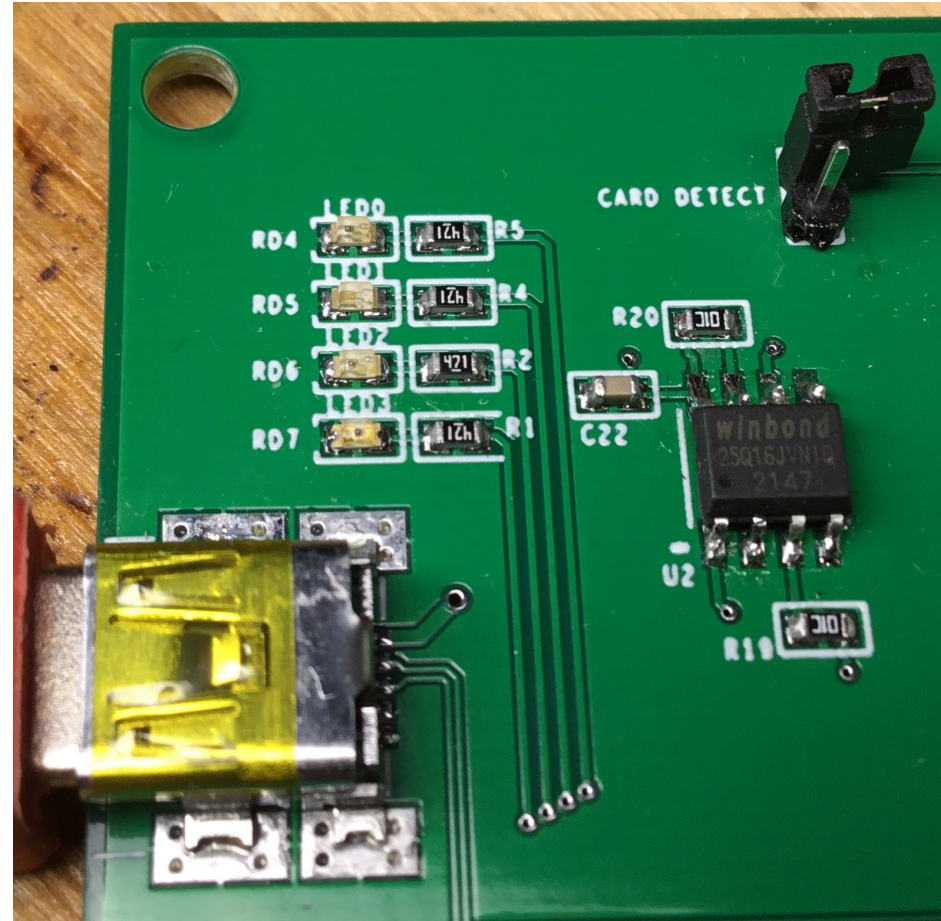


Fred Eady

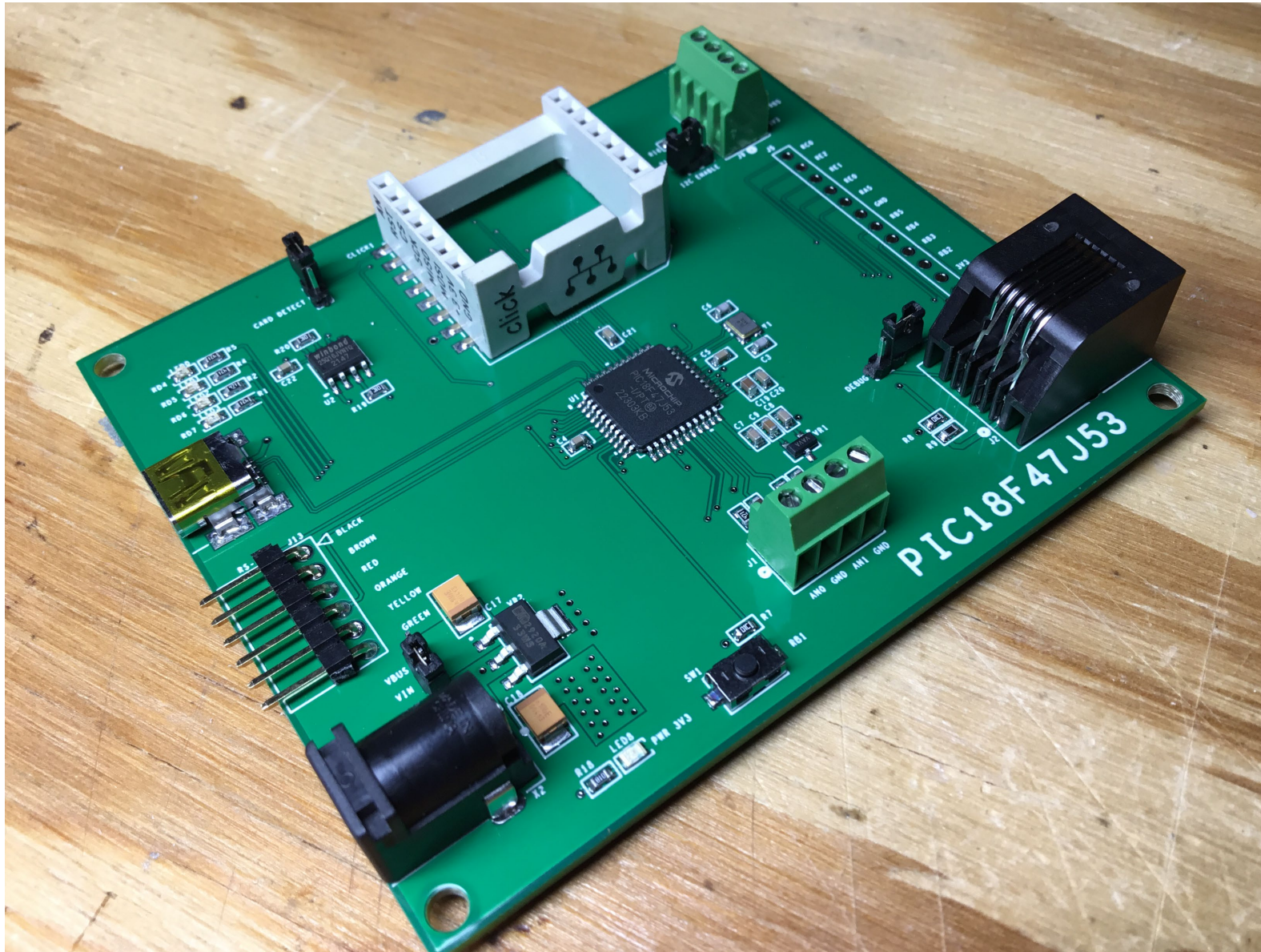
Visit 'Lecturer Profile' in your console for more details.

AGENDA

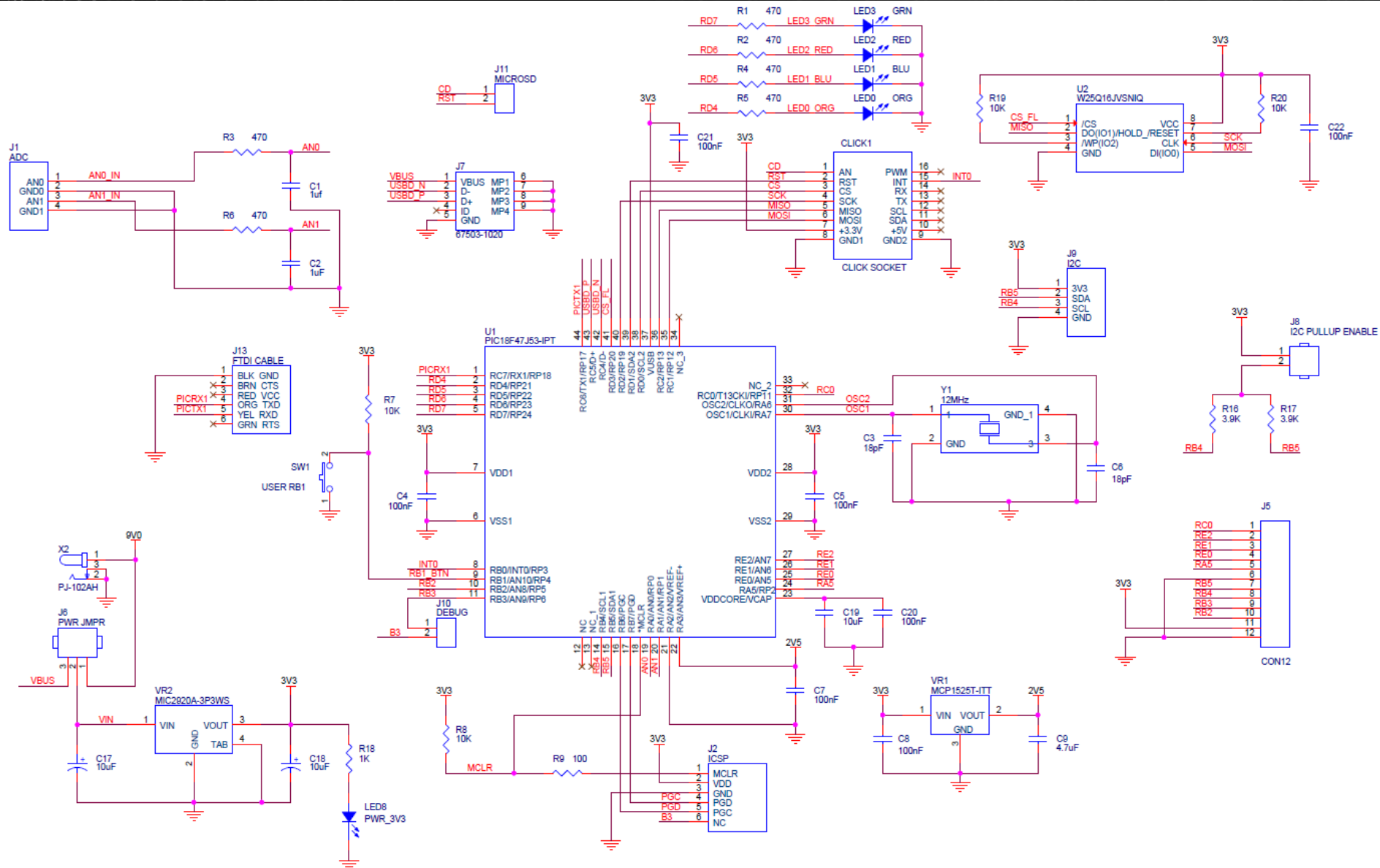
- **SpiFlash Hardware**
- **Code the W25Q Driver**
- **Test the W25Q Driver**



PIC18F47J53 Hardware



PIC18F47J53 Hardware



main.h

```
#include <18F47J53.h>
#device ADC=12

#FUSES NOWDT           //No Watch Dog Timer
#FUSES SOSC_HIGH       //High-power SOSC circuit is selected
#FUSES DSWDTOSC_INT    //DSWDT uses INTRC as reference clock
#FUSES RTCOSC_T1       //RTCC uses Secondary Oscillator as reference source
#FUSES IOL1WAY         //Allows only one reconfiguration of peripheral pins
#FUSES ADC12           //ADC is 12-bits
#FUSES MSSPMSK7        //MSSP uses 7 bit Masking mode
#FUSES WPFPP           //Write/Erase Protect Page Start/End Location, set to last page or use WPFPP=x to set page
#FUSES WPDIS           //All Flash memory may be erased or written
#FUSES WPEND           //Flash pages WPFPP to Configuration Words page are write/erase protected

#use delay(clock=48MHz,crystal=12MHz,USB_FULL)
#use FIXED_IO( D_outputs=PIN_D7,PIN_D6,PIN_D5,PIN_D4,PIN_D3,PIN_D2)

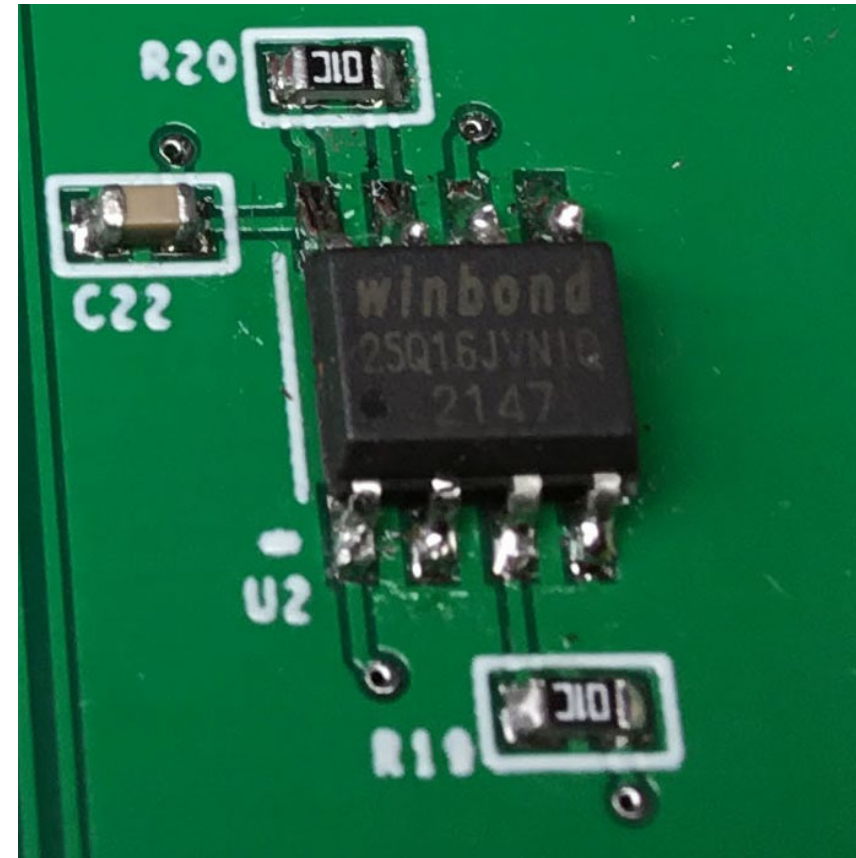
#define btnB1 PIN_B1
#define led0 PIN_D4
#define led1 PIN_D5
#define led2 PIN_D6
#define led3 PIN_D7
#define CS_FL PIN_D3

#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8,stream=EUSART1)

#define USB_CONFIG_BUS_POWER 500
#include <usb_cdc.h>
```

Function Prototypes

```
/**  
*****  
**/*  FUNCTION PROTOTYPES  
*****  
void readFlashByte(uint8_t hiAddr, uint8_t midAddr, uint8_t loAddr);  
void writeFlashByte(uint8_t hiAddr, uint8_t midAddr, uint8_t loAddr, uint8_t data);  
void readBlock(uint8_t hiAddr, uint8_t midAddr, uint8_t loAddr, uint8_t *buffer, uint16_t len);  
void writeBlock(uint8_t hiAddr, uint8_t midAddr, uint8_t loAddr, uint8_t *buffer, uint16_t len);  
void eraseBulk(void);  
void readStatusReg1(void);  
void readStatusReg2(void);  
void getJEDECID(void);  
uint32_t getAddress(uint16_t size);  
uint16_t sizeofStr(&inputStr);  
uint32_t getCapacity(void);  
uint32_t getMaxPage(void);  
  
typedef struct  
{  
    BOOLEAN supported;  
    BOOLEAN supportedMan;  
    BOOLEAN sfdpAvailable;  
    uint8_t manufacturerID;  
    uint8_t memoryTypeID;  
    uint8_t capacityID;  
    uint32_t capacity;  
    uint32_t eraseTime;  
}chipID;  
chipID chip;
```



SRAM Variables

```
//*****  
//*   SRAM VARIABLES  
//*****  
uint8_t  scratch8;  
uint16_t scratch16;  
  
uint8_t  tempVal;  
uint8_t  bufJEDEC[8];  
uint8_t  readbuf256[256];  
uint8_t  writebuf256[256];  
uint8_t  flashdata;  
uint8_t  eraseCmd;  
  
uint32_t  currentAddress;  
uint8_t   statusRegister1;  
uint8_t   statusRegister2;
```



SPI Definitions

Project Wizard - C:\Users\Public\cecCCS\day5_code\main.ccsproj

File Help

Options Code

SPI

Port Count: 1 PORT 1

Pin Assignments

Use hardware

SPI2

CLK pin: D2

DO pin: C1

DI pin: C2

Enable Pin: None

Diagnostics Pin: None

Load Pin: None

Timing

Baud: 0

High time(us): 0

Low time(us): 0

Enable delay(ms): 0

Data hold(ms): 0

Settings

Master

Mode: 0

Bits: 8

Load active: Low

Enable active: Low

First bit: MSB

Sample Count: 1

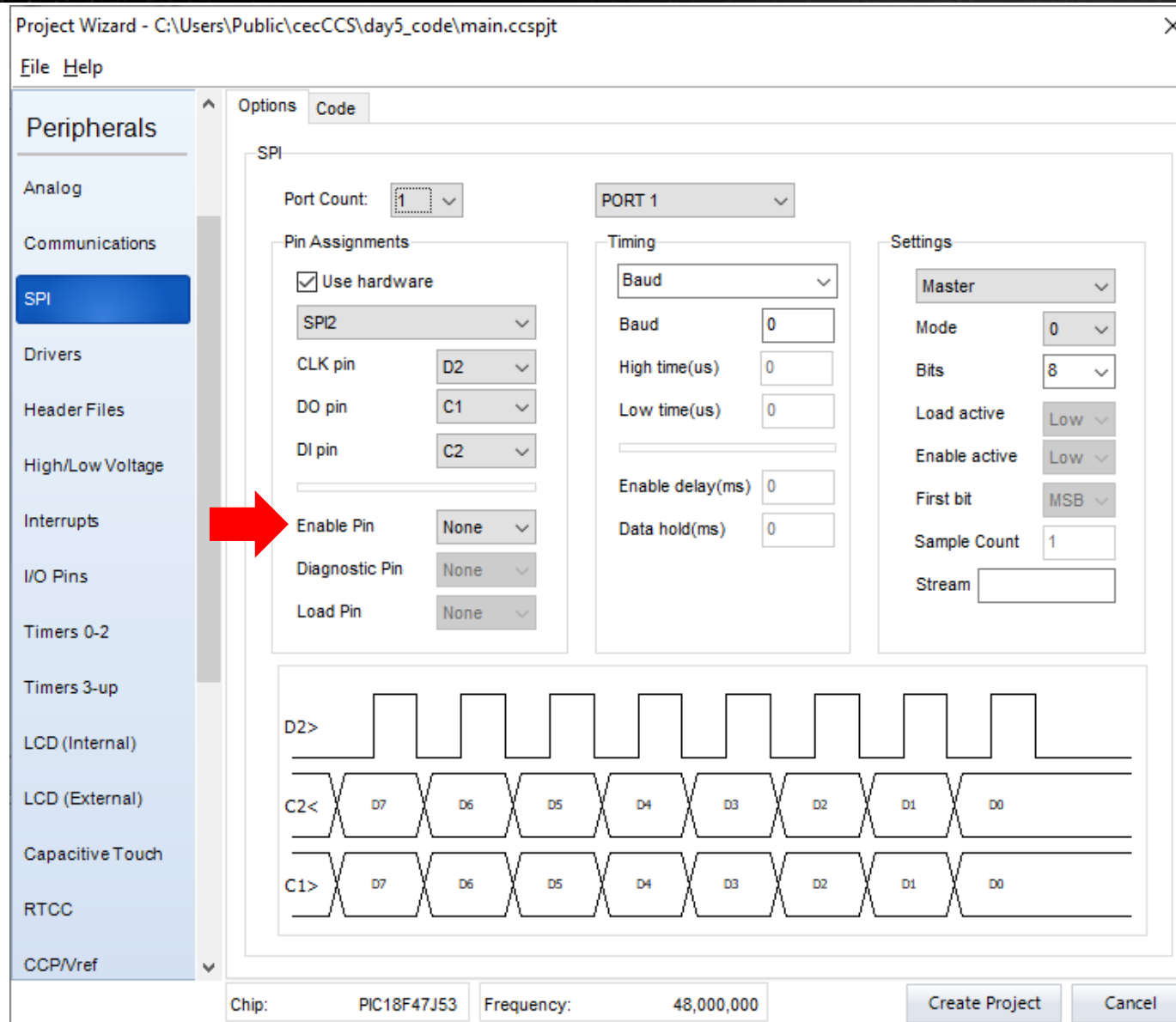
Stream:

D2>

C2< D7 D6 D5 D4 D3 D2 D1 D0

C1> D7 D6 D5 D4 D3 D2 D1 D0

Chip: PIC18F47J53 Frequency: 48,000,000 Create Project Cancel



SPI Definitions

Project Wizard - C:\Users\Public\cecCCS\day5_code\main.ccsproj

File Help

Options Code

Peripherals

- Analog
- Communications
- SPI
- Drivers
- Header Files
- High/Low Voltage
- Interrupts
- I/O Pins**
- Timers 0-2
- Timers 3-up
- LCD (Internal)
- LCD (External)
- Capacitive Touch
- RTCC
- CCP/Vref

I/O Pins

Pull-up Resistors

Port C

PIN_C0	None	Name:	
PIN_C1	None	Name:	
PIN_C2	None	Name:	
PIN_C4	None	Name:	
PIN_C5	None	Name:	
PIN_C6	None	Name:	
PIN_C7	None	Name:	

Port D

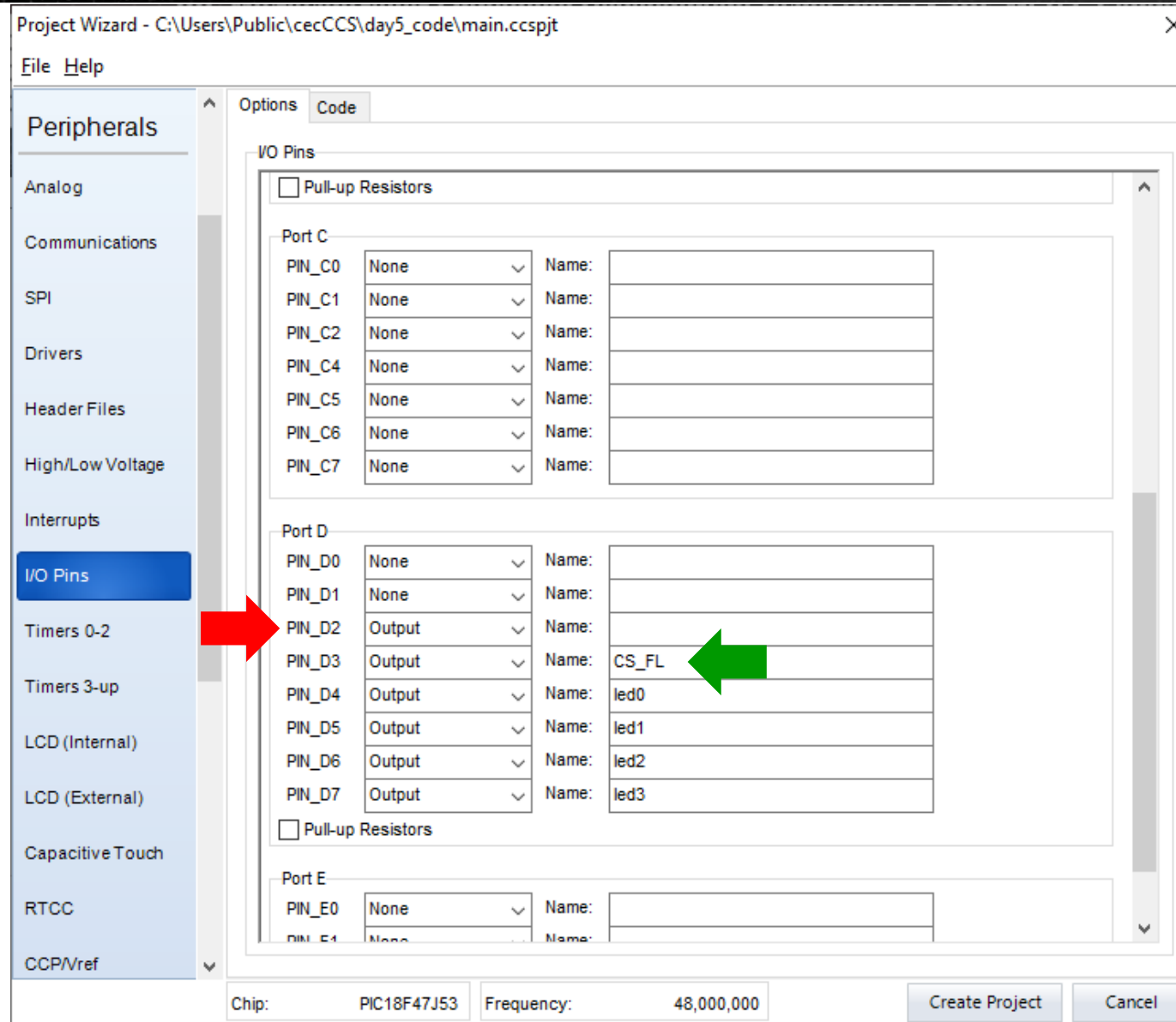
PIN_D0	None	Name:	
PIN_D1	None	Name:	
PIN_D2	Output	Name:	
PIN_D3	Output	Name:	CS_FL
PIN_D4	Output	Name:	led0
PIN_D5	Output	Name:	led1
PIN_D6	Output	Name:	led2
PIN_D7	Output	Name:	led3

Pull-up Resistors

Port E

PIN_E0	None	Name:	
PIN_E1	None	Name:	

Chip: PIC18F47J53 Frequency: 48,000,000 Create Project Cancel



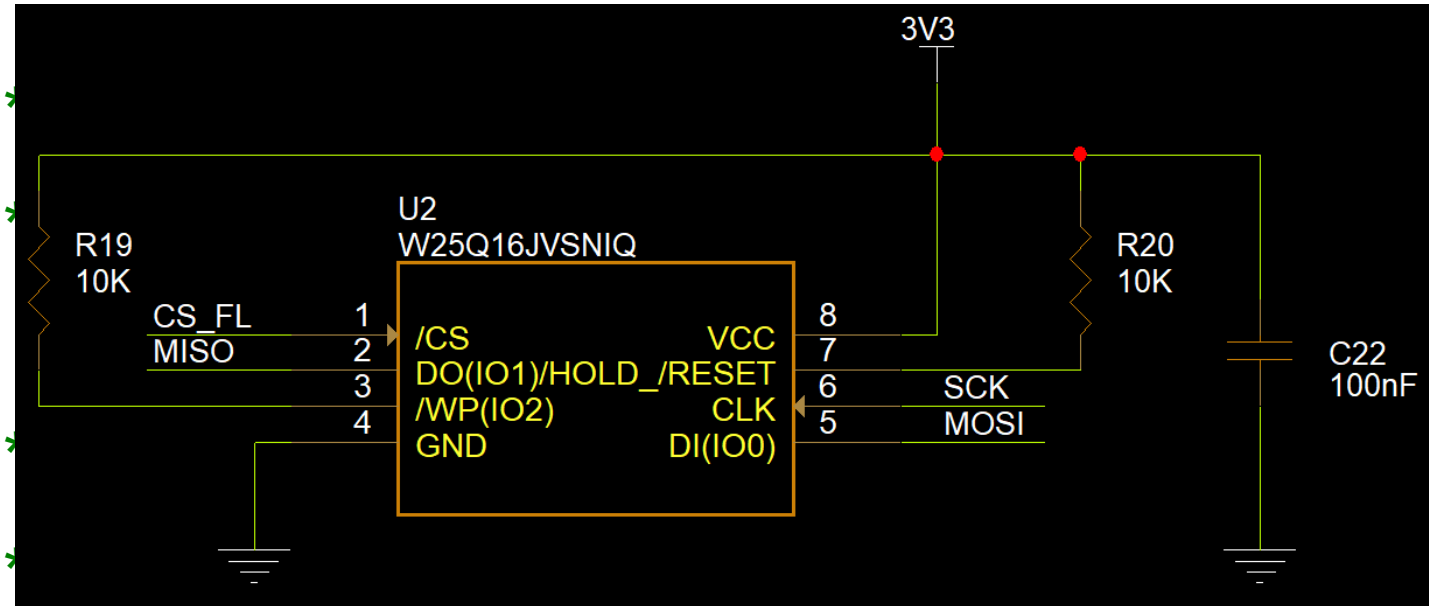
SPI Definitions

```

//*****
//*   SPI PIN SELECT DEFINITIONS
//*****
#pin_select SCK2=PIN_D2
#pin_select SDI2=PIN_C2
#pin_select SD02=PIN_C1
//*****
//*   I/O MACROS AND ALIASES
//*****
#define selectFlash output_low(CS_FL)
#define deselectFlash output_high(CS_FL)

#use spi (MASTER, SPI2, MODE=0, BITS=8, STREAM=spiflash)

```



Commands

```

//*****
//*  COMMANDS
//*****
#define MANID          0x90
#define PAGEPROG      0x02
#define READDATA      0x03
#define FASTREAD      0x0B
#define WRITEDISABLE  0x04
#define READSTAT1     0x05
#define READSTAT2     0x35
#define READSTAT3     0x15
#define WRITESTATEN   0x50
#define WRITESTAT1    0x01
#define WRITESTAT2    0x31
#define WRITESTAT3    0x11
#define WRITEENABLE   0x06
#define ADDR4BYTE_EN  0xB7
#define ADDR4BYTE_DIS 0xE9
#define SECTORERASE   0x20
#define BLOCK32ERASE  0x52
#define BLOCK64ERASE  0xD8
#define CHIPERASE     0x60
#define ALT_CHIPERASE 0xC7 // Some flash chips use a different chip erase command
#define SUSPEND       0x75
#define ID            0x90
#define RESUME        0x7A
#define JEDECID       0x9F
#define POWERDOWN     0xB9
#define RELEASE       0xAB
#define READSFDP      0x5A
#define UNIQUEID      0x4B
#define FRAMSERNO     0xC3

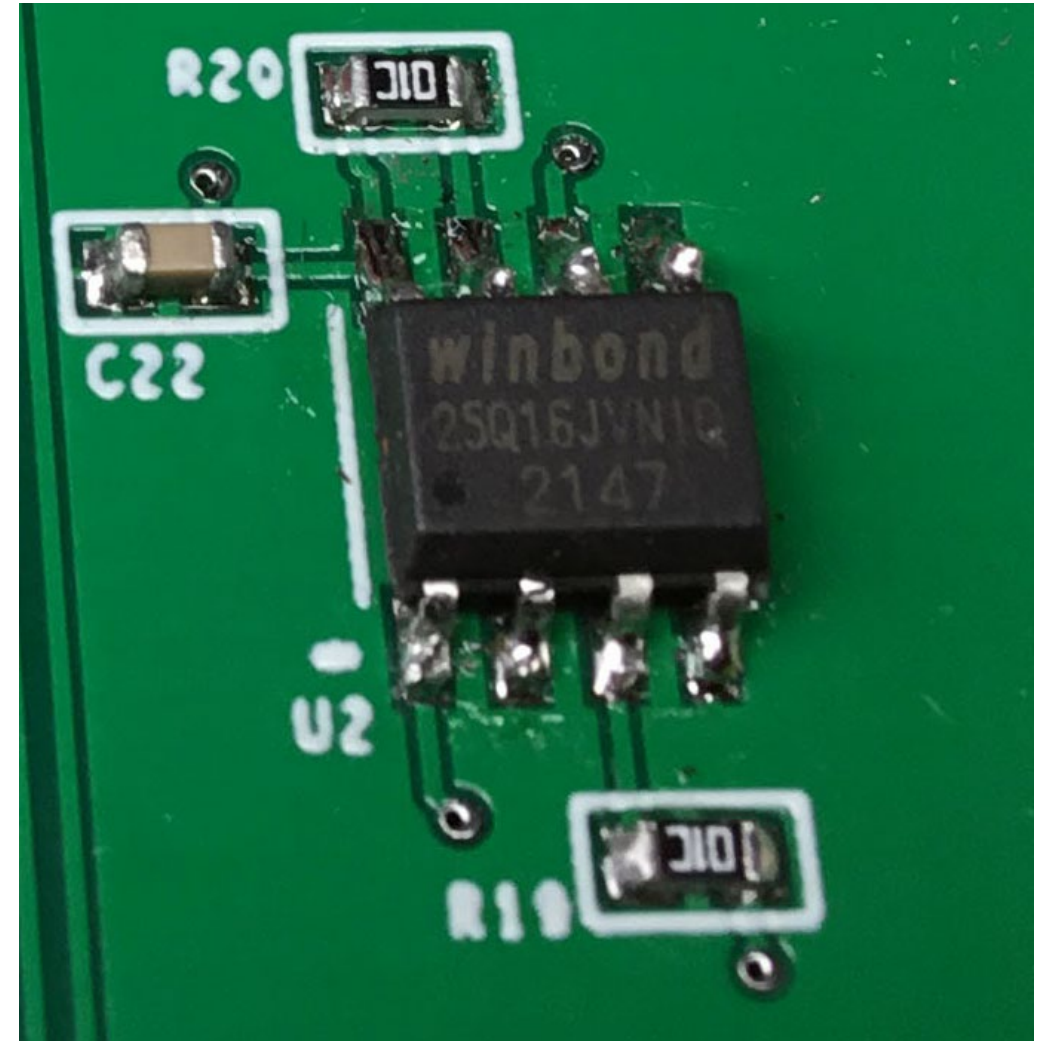
#define NOTBUSY       0x00
#define BUSY          0x01
#define WRTEN         0x02

```



Manufacturer IDs

```
// MANUFACTURER IDs  
#define microchipID 0xBF  
#define winbondID 0xEF  
#define cypressID 0x01
```



Read Status Register 1

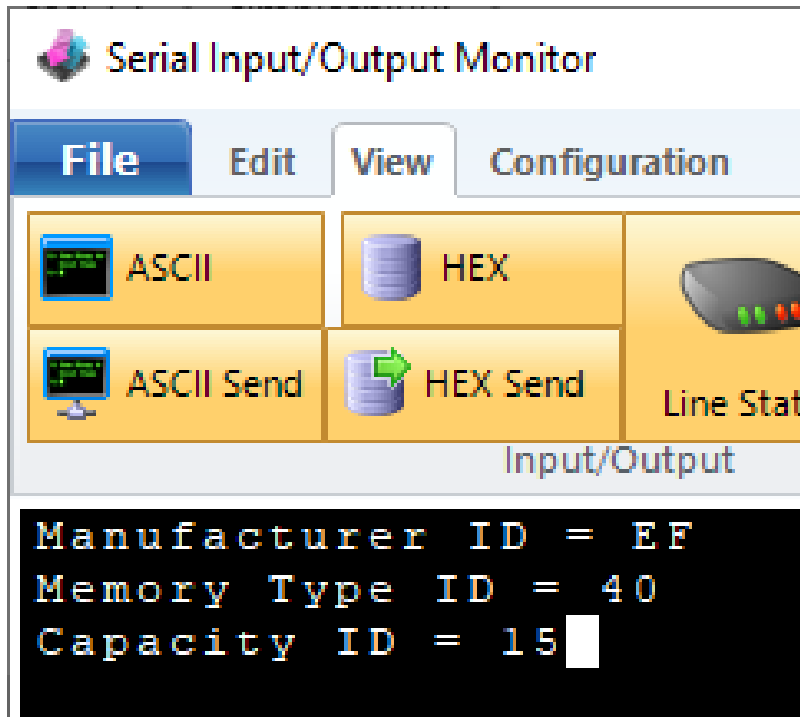
```
/**
 * READ STATUS REGISTER 1
 * Returns contents of Status Register 1 in statusRegister1
 */
void readStatusReg1(void)
{
    selectFlash;
    spi_xfer(READSTAT1);
    statusRegister1 = spi_xfer(0);
    deselectFlash;
}
```

Read W25Q16JV-IQ JEDEC ID

```
//*****  
//*   READ FLASH JEDEC ID  
//*****  
void getJEDECID(void)  
{  
    uint8_t status;  
  
    do{  
        readStatusReg1();  
        status = statusRegister1 & (BUSY | WRTEN);  
    }while(status != NOTBUSY);  
  
    selectFlash;  
    spi_xfer(JEDECID);  
    chip.manufacturerID = spi_xfer(0);    // manufacturer id  
    chip.memoryTypeID = spi_xfer(0);    // memory type  
    chip.capacityID = spi_xfer(0);    // capacity  
    deselectFlash;  
}
```


Read W25Q16V-IQ JEDEC ID

```
void main()
{
    deselectFlash();
    getJEDECID();
    printf("Manufacturer ID = %02X\r\nMemory Type ID = %02X\r\nCapacity ID = %02X", chip.manufacturerID, chip.memoryTypeID, chip.capacityID);
    while(1);
}
```



9.1 Device ID and Instruction Set Tables

9.1.1 Manufacturer and Device Identification

MANUFACTURER ID	(MF7 - MF0)	
Winbond Serial Flash	EFh	
Device ID	(ID7 - ID0)	(ID15 - ID0)
Instruction	ABh, 90h, 92h, 94h	9Fh
W25Q16JV-IQ/JQ	14h	4015h
W25Q16JV-IM*/JM*	14h	7015h

Bulk Erase Function

```

//*****
/*  BULK ERASE ENTIRE FLASH ARRAY
/*  THIS TAKES A BIT OF TIME...
//*****
void eraseBulk(void)
{
    uint8_t status;

    fprintf(EUSART1, "\r\nBulk Erase Started \r\n");
    delay_ms(100);

    selectFlash;
    spi_xfer(WRITEENABLE);
    deselectFlash;

    selectFlash;
    switch(chip.manufacturerID)
    {
        case microchipID:
            spi_xfer(0xC7);
            break;
        case winbondID:
            spi_xfer(0x60);
            break;
        case cypressID:
            spi_xfer(0x60);
            break;
    }
    deselectFlash;

    do{
        readStatusReg1();
        status = statusRegister1 & (BUSY | WRTEN);
    }while(status != NOTBUSY);

    fprintf(EUSART1, "Bulk Erase Complete \r\n");
    delay_ms(100);
}

```



```

//*****
/*  COMMANDS
//*****
#define MANID          0x90
#define PAGEPROG      0x02
#define READDATA      0x03
#define FASTREAD      0x0B
#define WRITEDISABLE  0x04
#define READSTAT1     0x05
#define READSTAT2     0x35
#define READSTAT3     0x15
#define WRITESTATEN   0x50
#define WRITESTAT1    0x01
#define WRITESTAT2    0x31
#define WRITESTAT3    0x11
#define WRITEENABLE   0x06
#define ADDR4BYTE_EN  0xB7
#define ADDR4BYTE_DIS 0xE9
#define SECTORERASE   0x20
#define BLOCK32ERASE  0x52
#define BLOCK64ERASE  0xD8
#define CHIPERASE     0x60
#define ALT_CHIPERASE 0xC7 // Some flash chips use a different chip erase command

```

Write Block Function

```

//*****
/**  WRITE A BLOCK OF DATA TO FLASH
/**  flashAddr = 32-bit flash address
/**  len = number of bytes to write
/**  *buffer = array of data to write to flash
//*****
void writeBlock(uint32_t flashAddr, uint8_t *buffer, uint16_t len)
{
    uint16_t indx = 0;
    uint8_t status;

    selectFlash;
    spi_xfer(WRITEENABLE);
    deselectFlash;

    do{
        readStatusReg1();
        status = statusRegister1 & WRTEN;
    }while(status != WRTEN);

    selectFlash;
    spi_xfer(PAGEPROG);
    spi_xfer(High(flashAddr));
    spi_xfer(Mid(flashAddr));
    spi_xfer(Low(flashAddr));
    do{
        spi_xfer(buffer[indx++]);
    }while(indx < len);
    deselectFlash;
}

```

```

#define Low(param) ((char *)&param)[0] //0x000y
#define Mid(param) ((char *)&param)[1] //0x00y0
#define High(param) ((char *)&param)[2] //0x0y00
#define Highest(param) ((char *)&param)[3] //0xy000
#define Loww(param) ((int *)&param)[0] //0x00yy
#define Top(param) ((int *)&param)[1] //0xyy00

```

Read Block Function

```

//*****
/**  READ A BLOCK OF DATA FROM FLASH
/**  flashAddr = 32-bit flash address
/**  len = number of bytes to read
/**  *buffer = array to store data read from flash
//*****
void readBlock(uint32_t flashAddr, uint8_t *buffer, uint16_t len)
{
    uint8_t status;
    uint16_t indx = 0;

    do{
        readStatusReg1();
        status = statusRegister1 & (BUSY | WRTEN);
    }while(status != NOTBUSY);

    selectFlash;
    spi_xfer(READDATA);
    spi_xfer(High(flashAddr));
    spi_xfer(Mid(flashAddr));
    spi_xfer(Low(flashAddr));
    do{
        buffer[indx++] = spi_xfer(0);
    }while(indx < len);
    deselectFlash;
}

```

```

#define Low(param) ((char *)&param)[0] //0x000y
#define Mid(param) ((char *)&param)[1] //0x00y0
#define High(param) ((char *)&param)[2] //0x0y00
#define Highest(param) ((char *)&param)[3] //0xy000
#define Loww(param) ((int *)&param)[0] //0x00yy
#define Top(param) ((int *)&param)[1] //0xyy00

```

Initialize Function

```
/**
 * INITIALIZE FUNCTION
 */
void init(void)
{
    deselectFlash;
    delay_ms(100);

    setup_adc_ports(sAN1 | sAN0, VREF_VREF);
    setup_adc(ADC_CLOCK_INTERNAL | ADC_TAD_MUL_20);
    setup_spi(SPI_MASTER | SPI_XMIT_L_TO_H | SPI_CLK_DIV_64);

    //LOAD writebuf256 for testing
    scratch8 = 0x00;

    for(scratch16=0;scratch16<0x0100;scratch16++)
    {
        writebuf256[scratch16] = scratch8++;
    }

    usb_init();
    getJEDECID();
}
```



main Function

```
void main()
{
    uint16_t readbufindx, outerloop;
    init();
    eraseBulk();
    writeBlock(0x00000000, writebuf256, 0x0100);
    readBlock(0x00000000, readbuf256, 0x0100);
    readbufindx = 0x00;
    do{
        for(outerloop=0; outerloop<16; outerloop++)
        {
            for(scratch8=0; scratch8<16; scratch8++)
            {
                fprintf(EUSART1, "%2X ", readbuf256[readbufindx++]);
            }
            fprintf(EUSART1, "\r\n");
        }
        fprintf(EUSART1, "\r\n");
        delay_ms(500);
        readbufindx = 0x00;
    }while(1);

    while(TRUE)
    {
        //TODO: User Code
    }
}
```



Build and Run

Serial Input/Output Monitor

File Edit View Configuration

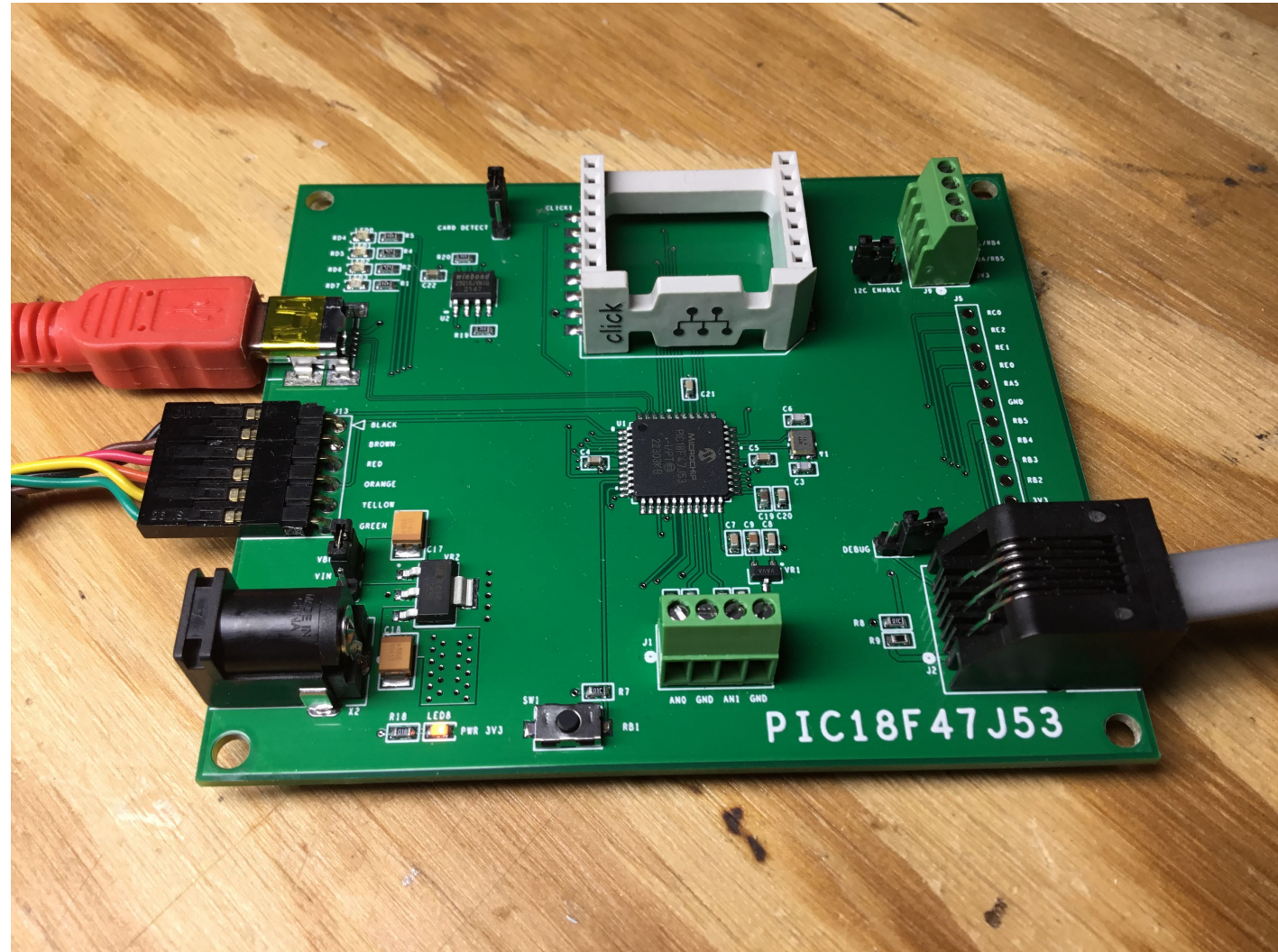
ASCII HEX ASCII Send HEX Send Line Status Clear Terminal Columns Display

Input/Output Viewing Options

```

Bulk Erase Started
Bulk Erase Complete
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F
70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF
B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF
E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF
F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF

00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F
70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF
B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF
    
```

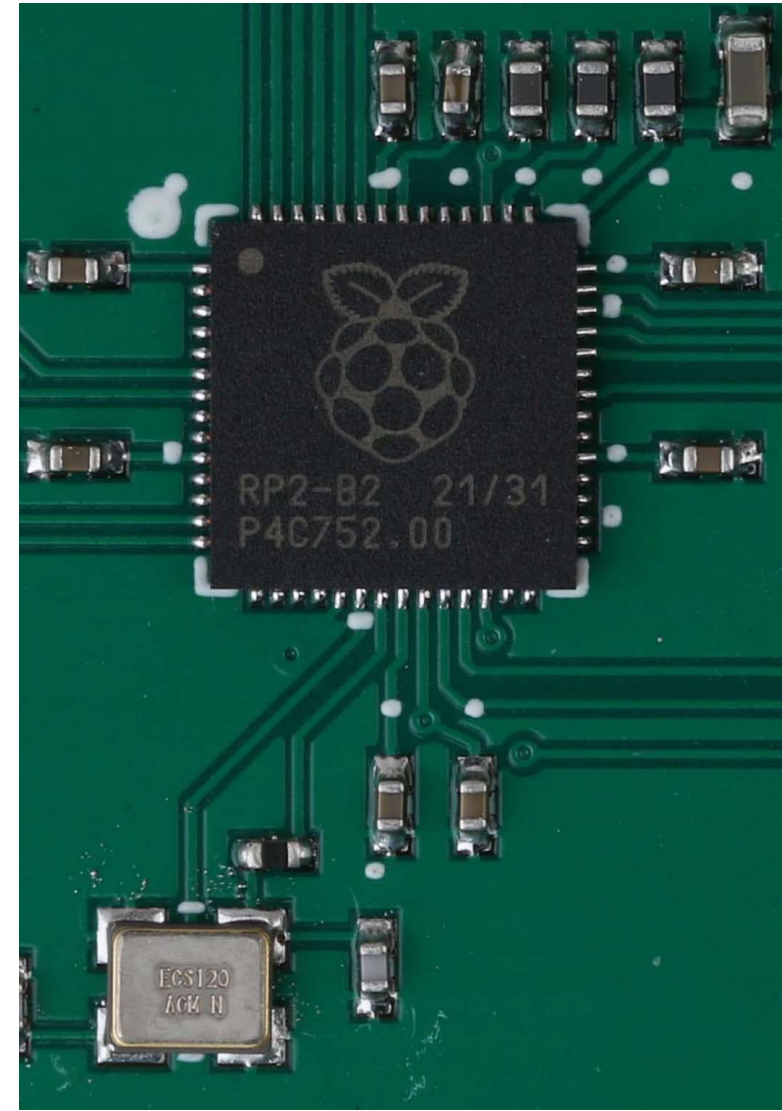


Thank you for attending!!!

Please consider the resources below:

- [ccsinfo.com](https://www.ccsinfo.com)
- **CCS C Compiler Manual**
- **Master and Command C for PIC MCU (PDF)**

MORE TO COME..





DesignNews

Thank You

Sponsored by

