Sparkfun 18F2553 USB Bit Wacker (UBW)
Quickstart Guide

Overview

This Quickstart Guide is meant to provide the user of the PIC18F2553 UBW a condensed guide on starting to use the UBW as well as a reference guide to inside details of the hardware and firmware.

Contents

Table I: Sparkfun PIC18F2553 USB Bit Wacker (UBW) Specifications

Feature	Specification
Microcontroller	Microchip PIC18F2553 microcontroller SOIC-28W Microchip PIC 18F2455/2550/4455/4550 Datasheet Microchip PIC18F2458/2553/4458/4553 Datasheet
FLASH/RAM/EEPROM	32K / 2K / 256 bytes
Clock	24 MHz Crystal Oscillator
USB	Full speed USB 2.0
Supply Voltage	4.2V-5.5V
Connectors	USB Mini-B Locations for two 0.1" headers (12-pin and 9-pin spaced apart 0.9") 5-pin Mini-ICSP port (1.5mm spacing) on back side
PCB Dimensions	1.0"x1.6"
Firmware	Microchip bootloader V1.00 UBW FW D Version 1.4.3
Switches	Two momentary switches for Reset and general input (also used to enter program mode during a reset)
Indicators	Power (Green), USB status (Yellow), Programming state (Red)
Serial communication	USART(TX/RX), SPI, I2C
Hardware Math support	8x8 Single cycle hardware multiplier
I/O	16 general I/O Configurable up to 16 digital inputs or outputs (25mA sink/source capability) Configurable up to 10 Analog channels Hardware digital Capture, Compare, and PWM output
A/D	12 bits, up to 10 channels
Timers	4

I. Steps to evaluate pre-programmed user firmware capabilities.

1. Download and install Microchip's "MCHPFSUSB Framework" – Google for latest release.
2. Connect the UBW USB to your computer.
    a. The **Power** LED and LED1 (**S1**) should be lit.
3. Point the device driver to the Microchip directory for the CDC RS-232 Emulator (driver identical for both releases)
    a. V2.3 release→C:\Microchip Solutions\USB Device – CDC – Basic Demo\inf
    b. V1.3 release→C:\MCHPFSUSB\fw\Cdc\inf\win2k_xp_vista32_64
    c. The LED1 (**S1**) should be blinking after installation of the driver.
4. Start up a terminal emulator
    a. For XP, Microsoft includes HyperTerminal under All Programs→Accessories→Communications.
    b. Verify the assigned COM port using the device manager.
    c. Configure
        i. Select assigned COM port
        ii. Set no hardware handshaking
        iii. Echo typed characters (the UBW does not echo)
        iv. Linefeed????
5. Verify firmware version
    a. Type "V" or "v" and <Enter>
    b. If purchased recently, "UBW FW D Version 1.4.3" will be returned.
6. Read the status of all the ports
    a. Type "I" or "I" and <Enter>
    b. The UWB will return "I,n,n,n" where n is a decimal representation of the port inputs.
7. Configure the ports
    a. Type "C,0,0,5" and <Enter>
    b. This will turn off the S1 LED since the assigned pin is now configured for input.
8. Continuously read back the port values
    a. Type "T,200,0" and <Enter>
    b. This will return all port values every 200ms or 5 times per second.
    c. Press the PGM button to see the port value change.
9. Reset the UBW
    a. Type "R" or "r" and <Enter>
    b. Repeat until the UBW is reset
    c. LED1 should be blinking
10. Read a memory location in RAM
    a. Type "MR,1234"
    b. Returned is the value at address 1234.
11. More details on the V1.4.3 firmware→ http://www.schmalzhaus.com/UBW/Doc/FirmwareDDocumentation_v143.html

II. Startup the bootloader if you are ready to load compiled code onto the UBW.

1. Put the UBW into bootloader mode
    a. Press the **PRG** button while plugging in the UBW into the USB, OR
    b. Press **Reset** and **PRG** buttons and then release the **Reset** button while holding down the **PRG** button.
    c. The yellow LED1 will blink.
2. Point the device driver loader to the Microchip \MCHPUSB Driver \Release directory for the bootloader driver (driver identical for both releases).
    a. V2.3 release →C:\Microchip Solutions\USB Tools\MCHPUSB Custom Driver\MCHPUSB Driver\Release
    b. V1.3 release →C:\MCHPFSUSB\Pc\MCHPUSB Driver\Release
    c. The yellow LED1 and red LED2 will alternatingly blink.
3. Launch the Microchip PICDEM FS USB Bootloader on the PC
    a. V2.3 release →C:\Microchip Solutions\USB Tools\Pdfsusb.exe
        i. Note: USB HID Bootloader firmware is currently not loaded.
    b. V1.3 release →C:\MCHPFSUSB\Pc\Pdfsusb.exe
4. Read Device
    a. Select “PICDEM FS USB 0 (Boot)” from pull-down menu.
    b. Press “Read Device” or “R” to read program memory from UBW .
5. Save UBW firmware to hex file
    a. Press “Save To HEX File” or “S” to save HEX file.
    b. For information on HEX file format see → http://en.wikipedia.org/wiki/.hex
    c. Review the HEX file output.
6. Run user firmware
    a. Press “Execute” or “E” to run user firmware (Same as Reset).
7. More details on the bootloader →http://www.schmalzhaus.com/UBW/Doc/FirmwareBDocumentation.html
8. New user firmware can be loaded (For advanced users only – do at your own risk).
    a. Put UBW into bootloader mode (Step 1).
    b. Press “Load HEX File” or “L” to load HEX file.
    c. Press “Program Device” or “P” to upload the HEX file to the UBW.
    d. Press “Execute” or “E” to run the new user firmware (Same as Reset).

III. Compiling new user-ware

1. Load Compiler
    a. Download, unzip, and install the Microchip C18 Compiler (Student edition available for free)
    b. Download, unzip, and install the Microchip MPLAB Integrated Development Environment (free)
    c. Download, unzip, and install any additional updates.
    d. Download available User Guides for future quick reference or create bookmarks.
2. Load User-ware source code project
    a. Download, unzip, and install the user firmware source code from http://schmalzhaus.com/UBW/FW/D_143/D_143.zip
3. Startup **MPLAB IDE**.
    a. Open up the FW_D_2455 workspace from File>Open Workspace…
    b. Read *MPLAB-C18-Getting-Started*, additional information available thorough Help>WebLinks
    c. To prevent a 62 charter length limitation with the COFF to COD file converter, check “Suppress COD file generation” box through menu Project>Build Options...>Project under the MPLINK Linker tab.
    d. Build and compile project through Project>Build All or from Toolbar.
4. The output “.HEX” file can be loaded into the UBW following step 8 of the previous section.
5. Modify the user.c source code for your own project needs.
    a. Realize that if you overwrite the bootloader, you will have to reload the UBW bootloader using a PIC programmer. Usually this will not be an issue unless you modify the linker file or explicitly specify an address below 0x800, or modify configuration bits which prevent the USB from functioning.

IV. Inside details on the firmware.

Bootloader lies at 0x000 to 0x7FF
Pre-loaded firmware HEX file→http://schmalzhaus.com/UBW/FW/D_143/_2550output/FW_D_2455.hex
Pre-loaded Bootloader firmware HEX file→ http://schmalzhaus.com/UBW/FW/2455/B/_output/FW_B_20.hex
User code lies from 0x800 to 0x460D (memory ends at 0x5FFF). Nearly 5K available for future feature expansion.

Pre-loaded firmware source files → http://schmalzhaus.com/UBW/FW/D_143/D_143.zip
Pre-loaded Bootloader firmware source files → http://schmalzhaus.com/UBW/FW/2455/B/B.zip
Changes to bootloader include changing io_cfg.h and adding line 51 `#pragma udata access fast_vars` to boot.c.

Modified user firmware files :

\UBW\FW\D_143\user\user.c \\ User code on version 1.4.3 Rev. D of the firmware.
UserInit(); Initialization of UBW configuration
TMR0 Timer
TMR2 1ms timer for PWM
ProcessIO(); \\ USB I\O processing
SwitchIsPressed (); Status of PRG button
BlinkUSBStatus(); Blink S1 LED with USB status
parse_packet (); Distributes to individual command parsers
low_ISR(); \\ Low priority interrupt routine
high_ISR(); \\ High priority interrupt routine (original, not modified)

\UBW\FW\D_143\io_cfg.h \\ I\O configuration for header file for UBW (UBW specific)
LED1 = RC0
LED2 = RC1
SW(PRG) = RC2
SDA = RB1
SCL = RB0

\UBW\FW\B\main.c \\ Bootloader configuration bits modified
`#pragma udata access fast_vars` \\ Added to reduce allow code to fit below 0x800
\UBW\FW\B\io_cfg.h \\ I/O port definitions (see user FW definitions above)

User firmware project files original from Microchip (not modified):

user.h
cdc.c, cdc.h
main.h
typedefs.h
usb9.c
usbcrtltrf.c
usbdrv.c
usbdsc.c, usbdsc.h
usbmmap.c, usbmmap.h
usb.h
usbcfg.h
p18f2455.h
rm18f2455.lkr

Bootloader firmware project files original from Microchip (not modified):

boot.c, boot.h
main.c, main.h
usb9.c
usbcrtltrf.c
usbdrv.c
usbdsc.c, usbdsc.h
usbmmap.c, usbmmap.h
usb.h
usbcfg.h
rm18f2455.lkr

Used for I2C configuration (future)- need to modify for 18F2553.

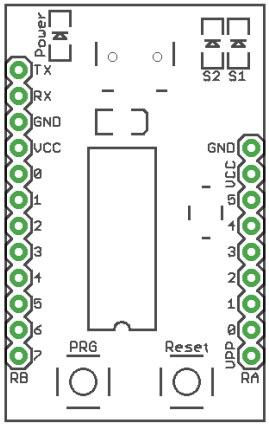
I2C Line	Macros	Default Value	Use
DATA Pin	DATA_PIN	PORTBbits.RB4	Pin used for the DATA line.
	DATA_LAT	LATBbits.RB4	Latch associated with DATA pin.
	DATA_LOW	TRISBbits.TRISB4 = 0;	Statement to configure the DATA pin as an output.
	DATA_HI	TRISBbits.TRISB4 = 1;	Statement to configure the DATA pin as an input.
CLOCK Pin	SCLK_PIN	PORTBbits.RB3	Pin used for the CLOCK line.
	SCLK_LAT	LATBbits.LATB3	Latch associated with the CLOCK pin.
	CLOCK_LOW	TRISBbits.TRISB3 = 0;	Satement to configure the CLOCK pin as an output.
	CLOCK_HI	TRISBbits.TRISB3 = 1;	Statement to configure the CLOCK pin as an input.

RB1=SCK
RB0=SDA

Appendix A: Technical Reference

Table II: Sparkfun PIC18F2553 UBW Pinout

Alternate Assignments
RC6/CK
RC7/DT/SDO
AN12/INT0/FLT0/SDI/SDA
AN10/INT1/SCK/SCL
AN8/INT2/VMO
AN9/CCP2/VPO
AN11/KBI0
KBI1/PGM
KBI2/PGC
KBI3/PGD



Alternate Assignments
AN4//HLVDIN/C2OUT
T0CKI/C1OUT/RCV
AN3/VREF+
AN2/VREF-/CVREF
AN1
AN0
/RE3

Table III: PIC18F2553 Pin Assignments

UBW Primary assignment	Pin	Pin Name	Pin Name	Pin	UBW Primary assignment
VPP	1	/VPP/RE3	RB7/KBI3/PGD	28	RB7
RA0	2	RA0/AN0	RB6/KBI2/PGC	27	RB6
RA1	3	RA1/AN1	RB5/KBI1/PGM	26	RB5
RA2	4	RA2/AN2/VREF-/CVREF	RB4/AN11/KBI0	25	RB4
RA3	5	RA3/AN3/VREF+	RB3/AN9/CCP2(1)/VPO	24	RB3
RA4	6	RA4/T0CKI/C1OUT/RCV	RB2/AN8/INT2/VMO	23	RB2
RA5	8	RA5/AN4//HLVDIN/C2OUT	RB1/AN10/INT1/SCK/SCL	22	RB1
GND	7	VSS	RB0/AN12/INT0/FLT0/SDI/SDA	21	RB0
OSC1	9	OSC1/CLKI	VDD	20	VCC
OSC2	10	OSC2/CLKO/RA6	VSS	19	GND
LED1	11	RC0/T1OSO/T13CKI	RC7/RX/DT/SDO	18	RX
LED2	12	RC1/T1OSI/CCP2(1)/	RC6/TX/CK	17	TX
PRG-switch	13	RC2/CCP1	RC5/D+/VP	16	D+
100nF Cap	14	VUSB	RC4/D-/VM	15	D-

Note 1: RB3 is the alternate pin for CCP2 multiplexing (For this UBW, this assignment is required since RC1 and RC2 are connected to LED2 and the PRG switch) .

Table IV: PIC18F2553 Pin Name Descriptions

Pin Name	Description
RAx, RBx, RCx, REx	Pin-by-pin configurable digital I/Os
ANx	Analog inputs
INTx, KBIx	External interrupts, Interrupt on change inputs
CCPx	Capture Compare
CxOUT	Comparator Output
D+, D-	USB data plus and minus
VP, VM, VPO, VMO	External USB transceiver lines
	SPI Slave Select
SDI, SDO, SCK	SPI Data In, SPI Data Out, SPI ClocK
SDA, SCL	I2C Data I/O, I2C Clock
TX, RX	USART (RS-232) Transmit/Receive
TxCKI	Timer ClocK Inputs
T1OSI, T1OSO	Timer1 oscillator input and output
VPP, PGC, PGD, PGM	ICSP programming
HLVDIN	High/Low-Voltage Detect Input (for brownout detection)
CLKI, CLKO	External digital clock input and output
VREF+, VREF-	ADC reference voltages

Note: See PIC18F2550/3 datasheets for additional details.

Table V: Version 1.4.3 Command set

Command	Description	Full command sequence followed by <CR>
R	**Reset**	R
C	**Configure I\O and analog pins**	C,<DirA>,<DirB>,<DirC>,<AnalogEnableCount> <DirA>,<DirB>,<DirC> = [0..255]; Output(0), Input(1) for each bit. <AnalogEnableCount>=[0..12]; Number of I\Os to enable for analog input.
O	**Output to digital ports** Writes to all three ports with values.	O,<PortA>,<PortB>,<PortC> <PortX>=[0..255]; Output value to each port
I	**Input from digital ports** Returns decimal values for PortA, PortB, and PortC separated by commas.	I
V	**Returns firmware version**	V
A	**Analog input request** Returns decimal values for all enabled analog inputs separted by commas.	A
T	**Timed I\O (digital and analog)** Periodically returns “I” and “A” packets	T,<Time>,<Mode> <Time>=[1..30000]; Time in units of ms. <Mode>=[0,1]; Digital “I” packets (0), Analog “A” packets (1)
PI	**Pin Input read** Returns decimal value of Port AND’ed with 2^^<Pin>	PI,<Port>,<Pin> <Port> = [A,B,C] <Pin>=[0..7]
PO	**Pin Output write** Writes to bit <Pin>	PO,<Port>,<Pin>,<Value> <Port> = [A,B,C] <Pin>=[0..7] <Value>=[0..255]
PD	**Pin Direction set**	PD,<Port>,<Pin>,<Direction> <Port> = [A,B,C] <Pin>=[0..7] <Direction>=[0,1] ; Output(0), Input(1)
MR	**Memory Read**	MR,<Address> <Address>=[0..4095]; RAM address
MW	**Memory Write**	MW,<Address>,<Value> <Address>=[0..4095]; RAM address <Value>=[0..255]
CU	**Configure UBW** Setting parameter 1 to a value of 0 turns off the “OK” response.	CU,<Parameter>,<Value> <Parameter>=[0..255]; <Value>=[0..255]
RC	**RC servo pulse output**	RC,<Port>,<Pin>,<Value> <Port> = [A,B,C] <Pin>=[0..7] <Value>=[0..11890]
BO	**Bulk digital Output****	BO,<ASCII_HEX_Bytes> <ASCII_HEX_Bytes>=[00..FF]xN; Each byte in Hex is concatenated.
BC	**Bulk digital Configure****	BC,<Init>,<WaitMask>,<WaitDelay>,<StrobeMask>,<StrobeDelay> <Init>=[0..255]; <Init> is initial value written to PortA. <WaitMask>=[0..255]; “Busy” bit mask used if <WaitDelay> > 0. <WaitDelay>= [0..255]; Units of 400ns. <StrobeMask>=[0..255]; XOR’ed with <Init> for strobe signal. <StrobeDelay>=[0..255]; Units of 830ns.
BS	**Binary Send to parallel output****	BS,<ByteCount>,<BinaryStreamOfBytes> <ByteCount>=[1..56]; Number of bytes to be written <BinaryStreamOfBytes>= Sequence of 8 bit characters to be written

** PortB is assigned as the parallel output port, PortA is the busy input and strobe output defined by <WaitMask> and <StobeMask>. <Init> is initial value written to PortA.
<WaitDelay>: The <WaitDelay> is the maximum amount of time to wait for the busy bit to become asserted, and then to become de-asserted. If <WaitDelay> expires, then the next byte is just sent out. After <StrobeDelay>, PortA=<Init>
<StrobeDelay>: The length of time that the strobe bits (from <StrobeMask>) are inverted from their initial values.

Table VI: Future Commands

Command	Description	Full command sequence followed by <CR>
CX	Configure serial port	**Not Implemented**
TX	Transmit serial data	**Not Implemented** RX,<ByteCount>,<BinaryStreamOfBytes> <ByteCount>=[1..56]; Number of bytes to be written <BinaryStreamOfBytes>= Sequence of 8 bit characters to be written
RX	Receive serial data	**Not Implemented** RX,<LengthRequest> <LengthRequest>=[0..??]
SS	Send SPI	**Not Implemented**
RS	Receive SPI	**Not Implemented**
CS	Configure SPI	**Not Implemented**
SI	Send I2C	**Not Implemented**
RI	Receive I2C	**Not Implemented**
CI	Configure I2C	**Not Implemented**
TP	Toggle Pin	**Implemented in version 1.5.0 Beta** TP,<Port>,<Pin> <Port>=[A,B,C,D,E] <Pin>=[0..7]
SL	Set Latch	**Implemented in version 1.5.0 Beta** SL,<PortA>,[PortB],[PortC],[PortD],[PortE]
TO	porT Output	**Implemented in version 1.5.0 Beta** TO,<Port>,<Value> <Port>=[A,B,C,D,E] <Value>=[0..255]
TI	porT Input	**Implemented in version 1.5.0 Beta** TI,<Port> <Port>=[A,B,C,D,E]
PW	PWM output	**Implemented in version 1.5.0 Beta** PW,<Port>,<Pin>,<Value> <Port>=[A,B,C,D,E] <Pin>=[0..7] <Value>=[0..255]
SC	Serial Configure	**Implemented in version 1.5.0 Beta** SC,<DataPort>,<DataPin>,<DataState>,<ClockPort>,<ClockPin>,<ClockState>,<LatchPort>,<LatchPin>,<LatchState>
SO	Serial Output	**Implemented in version 1.5.0 Beta** SO,<Latch>,<Data1>,[Data2],[Data3]...
EC	stEpper Configure	**Implemented in version 1.5.0 Beta** EC,???
ES	stEpper Step	**Implemented in version 1.5.0 Beta** ES,???

Table VII: Linker Memory Map

Bank	NAME	START	END	PROTECTED	Description	Linker specific
CODEPAGE	vectors	0x0	0x1D	PROTECTED	Vectors for Reset, Interrupts, and Traps	Bootloader FW
CODEPAGE	page	0x1E	0x7FF		Bootloader code	Bootloader FW
CODEPAGE	boot	0x0	0x7FF	PROTECTED	Bootloader region	User FW
CODEPAGE	vectors	0x800	0x829	PROTECTED	Vectors for Reset, Interrupts, and Traps	User FW
CODEPAGE	page	0x82A	0x5FFF		User code	User FW
CODEPAGE	idlocs	0x200000	0x200007	PROTECTED	IDs	
CODEPAGE	config	0x300000	0x30000D	PROTECTED	Configuration Bits **SECTION NAME=CONFIG ROM=config**	
CODEPAGE	devid	0x3FFFFE	0x3FFFFF	PROTECTED	Device ID	
CODEPAGE	eedata	0xF00000	0xF000FF	PROTECTED	EEPROM location	
ACCESSBANK	accessram	0x0	0x5F		Fast vars location	
DATABANK	gpr0	0x60	0xFF		Bank0	
DATABANK	gpr1	0x100	0x1FF		Bank1	
DATABANK	gpr2	0x200	0x2FF		Bank2	
DATABANK	gpr3	0x300	0x3FF		Bank3, Stack location **STACK SIZE=0x100 RAM=gpr3**	
DATABANK	usb4	0x400	0x4FF	PROTECTED	Bank4	
DATABANK	usb5	0x500	0x5FF	PROTECTED	Bank5	
DATABANK	usb6	0x600	0x6FF	PROTECTED	Bank6	
DATABANK	usb7	0x700	0x7FF	PROTECTED	Bank7	
ACCESSBANK	accesssfr	0xF60	0xFFF	PROTECTED	Special Function Registers	

Table VIII: Vector addresses

Vector	Address	Description
RESET	0x000000	Reset Vector (bootloader)
LOW_INT	0x000008	Low-priority interrupt vector (bootloader)
HIGH_INT	0x000018	High-priority interrupt vector (bootloader)
TRAP	0x000028	Trap vector (bootloader)
USER_RESET	0x000800	User Reset Vector
USER_LOW_INT	0x000808	User Low-priority interrupt vector
USER_HIGH_INT	0x000818	User High-priority interrupt vector
USER_TRAP	0x000828	User Trap vector (i.e. math overflow)

Table IX: PIC Configuration Bits (As viewed in MPLAB and associated #pragma config parameters)

Address	Value	Category	Setting	#pragma config
300000	25	96MHz PLL Prescaler	Divide by 6 (24MHz input)	* PLLDIV = 6
		CPU System Clock Postscaler	[OSC1/OSC2 Src: /1][96MHz PLL Src: /2]	CPUDIV = OSC1_PLL2
		Full-Speed USB Clock Source Selection	Clock src from 96MHz PLL/2	USBDIV = 2
300001	0E	Oscillator	HS: HS+PLL, USB-HS	* FOSC = HSPLL_HS
		Fail-Safe Clock Monitor Enable	Disabled	FCMEN = OFF
		Internal External Switch Over	ModeDisabled	IESO = OFF
300002	3A	Power Up Timer	Enabled	PWRT = ON
		Brown Out Detect	Controlled with SBOREN bit	BOR = SOFT
		Brown Out Voltage	2.0V	BORV = 3
		USB Voltage Regulator	Enabled	VREGEN = ON
300003	1E	Watchdog Timer	Disabled-Controlled by SWDTEN bit	WDT = OFF
		Watchdog Postscaler	1:32768	WDTPS = 32768
300005	81	CCP2 Mux	RC1	** CCP2MX = ON
		PortB A/D Enable	PORTB<4:0> configured as digital I/O on RESET	PBADEN = OFF
		Low Power Timer1 Osc enable	Disabled	LPT1OSC = OFF
		Master Clear Enable	MCLR Enabled,RE3 Disabled	MCLRE = ON
300006	81	Stack Overflow Reset	Enabled	STVREN = ON
		Low Voltage Program	Disabled	LVP = OFF
		Extended Instruction Set Enable	bitDisabled	XINST = OFF
300008	0F	Code Protect 00800-01FFF	Disabled	CP0 = OFF
		Code Protect 02000-03FFF	Disabled	CP1 = OFF
		Code Protect 04000-05FFF	Disabled	CP2 = OFF
		Code Protect 06000-07FFF	Disabled	CP3 = OFF
300009	80	Code Protect Boot	**Enabled**	**CPB = ON**
		Data EE Read Protect	Disabled	CPD = OFF
30000A	0F	Table Write Protect 00800-01FFF	Disabled	WRT0 = OFF
		Table Write Protect 02000-03FFF	Disabled	WRT1 = OFF
		Table Write Protect 04000-05FFF	Disabled	WRT2 = OFF
		Table Write Protect 06000-07FFF	Disabled	WRT3 = OFF
30000B	A0	Config. Write Protect	Disabled	WRTC = OFF
		Table Write Protect Boot	**Enabled**	**WRTB = ON**
		Data EE Write Protect	Disabled	WRTD = OFF
30000C	0F	Table Read Protect 00800-01FFF	Disabled	EBTR0 = OFF
		Table Read Protect 02000-03FFF	Disabled	EBTR1 = OFF
		Table Read Protect 04000-05FFF	Disabled	EBTR2 = OFF
		Table Read Protect 06000-07FFF	Disabled	EBTR3 = OFF
30000D	40	Table Read Protect Boot	Disabled	EBTRB = OFF

Notes: *PLLDIV=6 and FOSC=HSPLL_HS are specific to the SFE 18F2553 UBW using a 24MHz crystal.
** For the SFE 18F2553 UBW, for CCP2 to be used CCP2MX=OFF. ← Note to Brian in future updates of the firmware.

Table X: UBW User Firmware Configurations

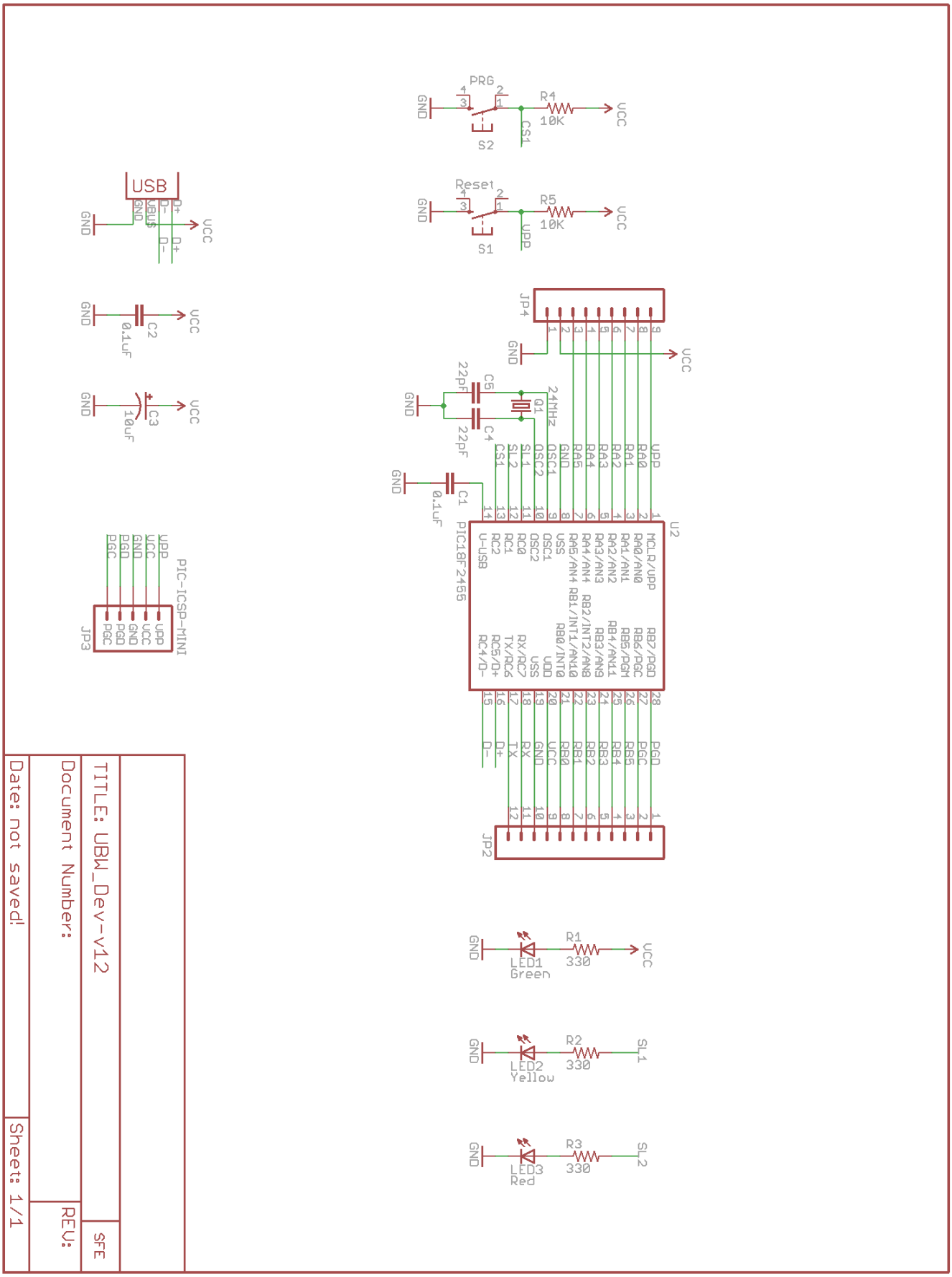
File	Define/Routine	Config parameter	Description
Main.c bootloader		ADCON1 |= 0x0F;	Set last 4 bits of ADCON1 to 1.
io_cfg.h	#define mInitSwitch()	TRISCbits.TRISC2 = 1;	Initalize switch as an input
io_cfg.h	#define UserSW	PORTCbits.RC2	Define UserSW as the **PRG** bit
io_cfg.h	#define mInitAllLEDs()	LATC &= 0xFC; TRISC &= 0xFC;	Initalize LED I/Os to outputs
io_cfg.h	#define mLED_1	LATCbits.LATC0	Define mLED_1 as the LED1 (**S1**) control bit
io_cfg.h	#define mLED_2	LATCbits.LATC1	Define mLED_2 as the LED2 (**S2**) control bit
user.c	UserInit()	LATA = 0x00;TRISA = 0xFF;	Assign all bits of PORTA inputs
user.c	UserInit()	ADCON1 = 0x0F;	Turn all analog inputs into digital inputs
user.c	UserInit()	ADCON0bits.ADON = 0;	Turn off the ADC
user.c	UserInit()	CMCON = 0x07;	Set comparators as digital inputs
user.c	UserInit()	LATB = 0x00; TRISB = 0xFF;	Assign all bits of PORTB inputs
user.c	UserInit()	LATC = 0x00; TRISC = 0xFF;	Assign all bits of PORTC inputs
user.c	UserInit()	mInitAllLEDs();	Initalize LED I/Os to outputs
user.c	UserInit()	mInitSwitch();	Initalize switch as an input
user.c	UserInit()	stdout = _H_USER;	Redirect stdout to USB
user.c	UserInit()	T2CONbits.T2CKPS1 = 1; T2CONbits.T2CKPS0 = 1;	The prescaler will be at 16
user.c	UserInit()	T2CONbits.T2OUTPS3 = 0; T2CONbits.T2OUTPS2 = 0; T2CONbits.T2OUTPS1 = 1; T2CONbits.T2OUTPS0 = 0;	We want the TMR2 post scaler to be a 3
user.c	UserInit()	 T0CONbits.PSA = 1; T0CONbits.T0CS = 0; T0CONbits.T08BIT = 0; INTCONbits.TMR0IF = 0; INTCONbits.TMR0IE = 0 ; INTCON2bits.TMR0IP = 0;	Enable TMR0 for our RC timing operation Do NOT use the prescaler Use internal clock 16 bit timer Clear the interrupt flag And clear the interrupt enable Low priority
user.c	UserInit()	RCONbits.IPEN = 1; T2CONbits.TMR2ON = 0; PIE1bits.TMR2IE = 1; IPR1bits.TMR2IP = 0;	Enable interrupt priorities
user.c	UserInit()	INTCONbits.GIEH = 1;	Global Interrupts High priority Enable on
user.c	UserInit()	INTCONbits.GIEL = 1;	Global Interrupts Low priority Enable on
user.c	UserInit()	T2CONbits.TMR2ON = 1;	Turn on Timer2

Table XI: ISR related configurations

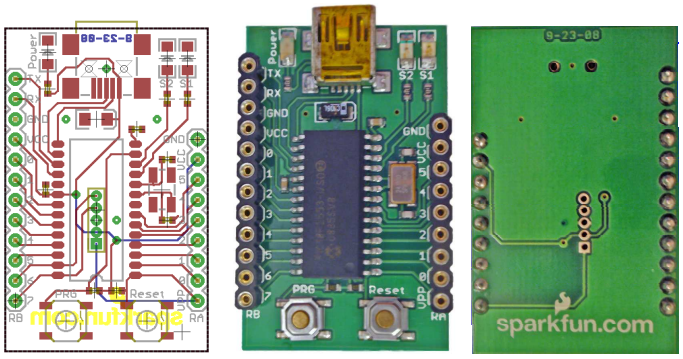
File	Define	Config parameter	Description
user.c	low_ISR()	if (PIR1bits.TMR2IF)	Do we have a Timer2 interrupt? (1ms rate)
		PIR1bits.TMR2IF = 0;	Clear the interrupt
		TMR0H, TMR0L	Set timer value
		INTCONbits.TMR0IE = 1;	Then make sure the timer's interrupt enable is set
		INTCONbits.TMR0IF = 0;	And be sure to clear the flag too
		T0CONbits.TMR0ON = 1;	Turn on Timer0
		if (AnalogEnable > 0)	See if it's time to start analog conversions
		ADCON0 =	Set the channel to zero to start off with
		PIR1bits.ADIF = 0;	Clear the interrupt
		IPR1bits.ADIP = 0;	And make sure to always use low priority.
		PIE1bits.ADIE = 1;	Set the interrupt enable
		ADCON0bits.ADON = 1;	Make sure it's on!
		ADCON0bits.GO_DONE = 1;	And tell the A/D to GO!
		if (PIR1bits.ADIF)	Do we have an analog interrupt?
		PIR1bits.ADIF = 0;	Clear the interrupt
		ADCON0 =	Update the channel number
		ADCON0bits.GO_DONE = 1;	And start the next conversion
		if (INTCONbits.TMR0IF)	Do we have a TMR0 interrupt? (RC command)
		T0CONbits.TMR0ON = 0;	Turn off Timer0
		INTCONbits.TMR0IF = 0;	Clear the interrupt
		INTCONbits.TMR0IE = 0;	And disable it
	parse_C_packet()	if (0 == AA)	If we are turning off Analog inputs
		ADCON1 = 0x0F;	Turn all analog inputs into digital inputs
		ADCON0bits.ADON = 0;	Turn off the ADC
		AnalogEnable = 0;	Turn off our own idea of how many analog channels to convert
		else	
		AnalogEnable = 0;	
		ADCON0 = 0;	Start by selecting channel zero
		ADCON1 = 15 - AA;	Then enabling the proper number of channels
		ADCON2 = 0b10111110;	Tad = Fosc/64
		ADCON0bits.ADON = 1;	Turn on the ADC
		AnalogEnable = AA;	Tell ourselves how many channels to convert,
		T2CONbits.TMR2ON = 1;	and turn on ISR conversions

Appendix B: Schematics and board layout

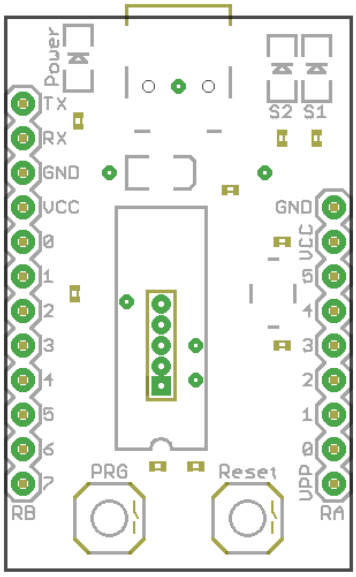
18F2553 UBW Schematic Version 12



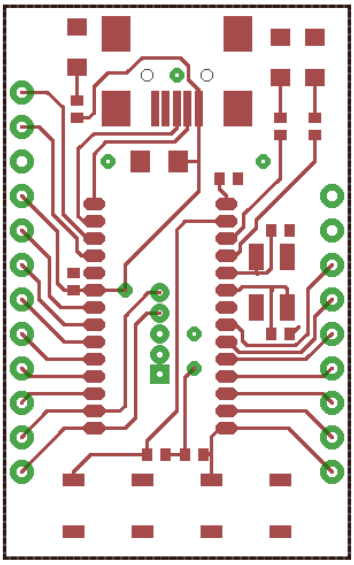
Sparkfun 18F2553 UBW board layouts (actual size, female headers ordered separately)



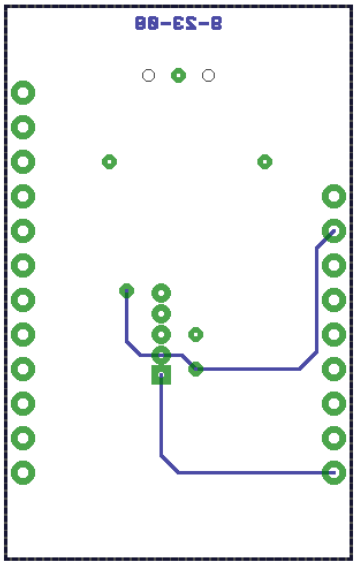
Eagle Board Layouts (Version 12)



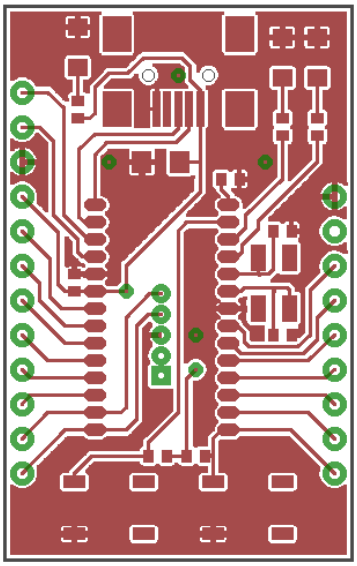
Silkscreen (gray)



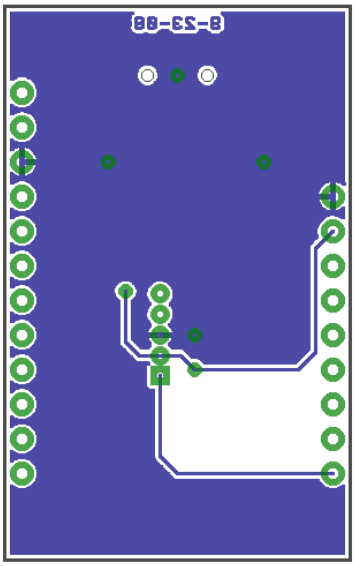
Top layer without fill



Bottom layer without fill



Top layer with fill



Bottom layer with fill