Wisard Version2 / 3 Reference

See Also Wisard_Message_Format, PartsList.pdf Wisard_Boards V1.pdf, Xbee9XRadioNotes.docx

Table Of Contents

Overview

This document describes the Version2.0 ISFS Wisard Prototype SensorNode Board based around the PIC24FJ64GB004 16-bit processor. This document describes how those boards and PIC processor.

SensorNodeBoard: This board is intended to sample and forward sensor data. Its serial output can be routed to either an Xbee2 radio and/or a direct RS232 connection. The Xbee radio can be linked to the RepeaterBoard or to a directly connected Xbee base radio attached to an ADAM. It includes 5 miniDin-6 connectors for i/o to I2C based PIC sensors. The RS232 is carried on a miniDin-6 connector also used for power input as with the Version1 boards. A DC-DC converter:……operating voltage of 3.3. A 2-pin connector allows a 'AA' battery pack to be used for either stand-alone and/or a quasi-ups operation.

Processor Features & Specifications:

Differences between PIC24F and PIC24H Processors

PIC24F: Lower Power, 16MIPS, Mid-Range Performance
PIC24H: 16-bit MCU at 40MIPS, Higher Performance
Family: Single-Cycle bit manipulation, instructions, hardware multiply, 5 cycle Intr.Response.

	PIC24F	PIC24H
ADC	10-bit	12-bit
Uarts	2-4	1-2
I2C	2-3	1-2
eeprom	0-512B (PIC24FK)	
ram	4-96k	1-16k
		Higher power / consumpt.
130degC		PIC24HJ, PIC33FJ
Special	DSWDT, DSBOR, USBOTG	DMA

For Wisard we will use the PIC24F family to save power

Comparison of PIC Processor Specifications:

Spec	***FJ64GB004***	***FJ256GB206 (206,210)***	***PIC18LF2520***
Operating Voltage Range	2.0 – 3.6	2.2 – 3.6	2.0 – 5.5
I/O Port Sink/Source current	18mA / 18mA Current Limit Resistance: Rmin = 3.3V.018 ~= 183ohms	18mA / 18mA Current Limit Resistance: Rmin = 3.3V.018 ~= 183ohms	25mA / 25mA Current Limit Ohms Rmin = 3.3/.025 ~=132ohms
	Note: Using 470ohm was too much and prevents multiple Wisard sensors from working. 200 did work in a bench test. 25mA absolute max implies at least 132ohm min.		

MicroChip Support:

Local Representatives: **Lange Sales** 1500 West Canal Court Bldg. A, Suite 100 Littleton,CO 80120 USA **Phone:** (303) 795-3600 **Fax:** (303) 795-0373; Judy Miller.

On-Line Support: FIRST: enter Tech.Question Ticket on-line If the email response does not solve the problem, contact the NORTH AMERICAN Support department by calling (480) 792-7627. Enter the Ticket ID shown in the subject line of this message for verification. Enter the Ticket ID which is 1-133803 substituting the '-' with the '*' key. Press # on your telephone when finished.

Version2 Board Notes

The PIC-24FJ64GB004 is a 16-bit class processor.

Program Memory: 64-kByte program flash (22,016 instructions)

Data Memory: 8192-bytes data SRAM
No internal EEPROM, Simulation software permits use of flash to simulate it

10bit ADC, 25 I/O pins (sink/source up to 25mA), 2-I2C, 2-of-SPI/USART, 5-16bit timers, USB-OTG

Internal Osc. 31k to 8mhz, Watchdog Timer

2nd Internal Osc Low-Power Internal RC Osc fixed at 31kHz (used in sleep/idle modes)

External Osc. upto 32MHz

Instruction Rate/Cycle:

Osc/2 = Fosc 1 Cycle to fetch (prefetch used) + 1 Cycle to execute (except instructions that change the program flow)

2.0 – 3.6V operation

PIC Hardware Serial Port:

UART1 and UART2 are on Reprogrammable Pins (PPS)

Board:		
SensorNode	PIC Tx to Max3323 RS232	Pin4, RP24
	PIC Rx from Max3323 RS232	Pin5, RP25
	PIC Tx to Xbee	Pin20, RP6
	PIC Rx from Xbee	Pin19, RP5

BaudRate Generation: See specific chip data sheet.

The 16-bit BRGenerator supports either sync or async modes. It controls a free-running timer which samples the i/o in either a rate of FOSC/32 (Standard) or FOSC/8 (High).

The Standard Calculation for the UxBRG register is: (FCY/16*BaudRate)-1. With bit BRGH=0 in the UxMODE register.

The High Mode has UxBRG = (FCY/4*BaudRate)-1. With BRGH=0

NOTE: **rates 57600, and 115200 are unusable** due to the high clock error from 'ideal'

BaudRate Values - Standard, SYSCLK=16000000, FCY=8000000

9600: BaudRateReg=51.08/ 51 Error=-0.16/0 %
19200: BaudRateReg=25.04/ 25 Error=-0.16/0 %
38400: BaudRateReg=12.02/ 12 Error=-0.16/0 %
57600: BaudRateReg= 7.68/ 7 Error=-8.51/-8 %
115200: BaudRateReg= 3.34/ 3 Error=-8.51/-8 %

BaudRate Values - High

9600: BaudRateReg=207.33/207 Error=-0.16/0 %
19200: BaudRateReg=103.17/103 Error=-0.16/0 %
38400: BaudRateReg=51.08/ 51 Error=-0.16/0 %
57600: BaudRateReg=33.72/ 33 Error=-2.12/-2 %
115200: BaudRateReg=16.36/ 16 Error=-2.12/-2 %

In these calculations the integer value gets programmed into the UxBRG register

PIC I2C:

The p24FJ64GB004 has 2 hardware I2C ports. These are on specific pins as opposed to many of the reprogrammable pin functions of the device. I2C1= 1(SDA1/RB9), 44(SCL1/RB8) and I2C2=23(SDA2/RB2), 24(SCL2/RB3). I2C2 is used to talk with the Wisard sensors

Board:	*Hardware I2C2:* *Clock RB3(Pin24), Data RB2(Pin23)*
SensorNodeV2.0	To Sensors (MiniDin's J1.1-5 only)
Sensors:	MiniDin J1-5 Pin-5 pulled to ground to force sensors into 'i2c mode'
	MiniDin J1-5 Pin6 NOTE: Jumperable on 'left-two' for +12vdc, Available for 'right-three' as gpio

I2C Addressing: 7-bit, although 10-bit is available. Valid Range = 0x8-0x7F Addresses 0-7 are reserved. The 'known' address is shifted 1 byte left for i/o calls (<<1), thus 0x49 becomes 0x92.

I/O clock speed: 100kHz or 400kHz
In Wisard Code, it is selected in the file: "i2c_master" with a define statement.

I2C Slave Addressing:
The address needs to be shifted by 1 bit, ie <<1 and the lsb set for either a read or write call.
For raw address = 0x20 I2CAdd = 0x20<<1 | 0x01 (for a read request)
I2CAdd = 0x20<<1 (for a write to device call)

Interrupt Service Routine:
void __attribute__((interrupt,no_auto_psv)) _MI2C2Interrupt(void)

Registers:

I2C2CON	Control Register
I2C2BRG	BaudRate in Master Mode: 100
I2C2STAT	Status Register

PIC Interruptible Control Lines:

RB0

Version2 Board PIC24FJ64GB004 – Pin Assignments

Pin	*PIC24FJ64GB004*	*Functional Assignment – V2*	
1	**RP9** / SDA1 / RB9	GPS – Rx (Uart2)	
2	RP22 / RC6	GPS – PPS	
3	RP23 / RC7	GPS Int0 (Not used functionally)	
4	RP24 / RC8	RS232 – Tx (Uart1 to MAX3323)	
5	RP25 / RC9	RS232 – Rx (Uart1 to MAX3323)	
6	DisVreg	Gnd – Enables OnBoard Voltage Regulator, for core logic to operate off 3.3	Allows low voltage detection.
7	Vcap / VddCore	LowESR cap to above to support Vreg.	
8	RP10 / PGED2 / RB10	ORG led, J3-pin3 aux. use	
9	RP11 / PGEC2 / RB11	J3 – pin4 aux. use	
10	Vusb	n.c.	
11	RP13 / AN11 / RB13	AN11 = Analog Vin	
12	RA10	Xbee SleepReq – (Not functional)	
13	RA7	XBEE Reset	
14	RP14 / AN10	AN10 = Iin	
15	RP15 / AN9	AN9 = I3.3	
16	AVss	Gnd: Caps tie together with	
17	AVdd	Vcc (+3) Positive Supply for ADC Modules	Vref here instead of 19?
18	MCLR		
19	RP5 / PGED3 / Vref+ / RA0	XBEE Tx – Uart1	
20	RP6 / PGEC3 / Vref- / RA1	XBEE Rx – Uart1	
21	RP0 / PGED1 / AN2 / RB0	XBEE Status	
22	RP1 / PGEC1 / **CN5** / RB1	CN5 (white) button: ScanI2c, Swap	
23	SDA2 / RP2 / AN4 / RB2	I2C2-Clk, Sensors	
24	SCL2 / RP3 / AN5 / RB3	I2C2-Data, Sensors	
25	RP16 / AN6 / RC0	RED led	
26	RP17 / AN7 / RC1	GRN led	
27	RP18 / AN8 / RC2	AN8 = Isensors	
28	Vdd	+3.3	
29	Vss	Gnd	
30	CLKI / OSCI / RA2	16Mhz TCXO in	
31	CLKO / OSCO / RA3	Clock Output Test Point	
32	RA8	Sensor On/Off Power Switch	
33	SOSCI / RP4 / RB4	RTCC Osc.	
34	SOSCO / RA4	RTCC Osc.	
35	RA9	YEL led	
36	RP19 / AN12 / RC3	MicroSD SPI- CS	
37	RP20 / RC4	MicroSD SPI- SDD	
38	RP21 / RC5	MicroSD SPI- Data	
39	Vss	Gnd	
40	Vdd	+3.3	
41	RB5	GPS On/Off Power Switch	
42	Vbus	n.c.	
43	RP7 / INT0 / RB7	MicroSD SPI – CLK	

44	RP8 / SCL1 / RB8	GPS Tx – Uart2	

Version3 Board Notes

Version3 Board PIC24FJ256GB206 – Pin Assignments

Pi	*PIC24FJ256GB206(206,210) Pin Functions Available*	*Pin Function Used – V3*	*Application Assignment – V3*	*V2*	*V2 Func*
1	PMD5 / CN63 / RE5	RE5	GPS-PowerSw		
2	SCL3 / PMD6 / CN 64/ RE6	I2C3-SCL3	GPS		
3	SDA3 / PMD7 / CN65 / RE7	I2C3-SDA3	GPS		
4	C1IND / RP21 / PMA5 / CN8 / RG6	RP21 / RG6	RX3 from GPS		
5	C1INC / RP26 / PMA4 / CN9 / RG7	RP26 / RG7	TX3 to GPS		
6	C2IND / RP19 / PMA3 / CN10 / RG 8	CN10 / RP19 / RG8	INTx from PPS	2	CN18 / RC6
7	MCLR				
8	C2INC / RP27 / PMA2 / CN11 / RG9	CN11 / RG9	SCAN BUTTON	22	CN5 / RB1
9	Vss				
10	Vdd				
11	PGEC3 / AN5 / C1INA / Vbuson / RP18 / CN7 / RB 5	PGEC3 / RP18 / RB5	TX1 to UART		
12	PGED3 / AN4 / C1INB / USB-OEN / RP28 / CN6 / RB4	PGED3 / RP28 / RB4	RX1 from UART		
13	AN3 / C2INA / VPIO / CN5 / RB3	AN3 / RB3	Vin	11	AN11 / RB13
14	AN2 / C2INB / VMIO / RP13 / CN4 / RB2	AN2 / RB2	Iin	14	AN10 / RB14
15	PGEC1 / AN1 / Vref- / RP1 / CN3 / RB1	Vref-			
16	PGED1 / AN0 / Vref+ / RP0 / PMA6 / CN2 / RB0	Vref+			

Pi	PIC24FJ256GB206(206,210) Pin Functions Available	Pin Function Used – V3	Application Assignment – V3	V2	V2 Func
17	PGEC2 / AN6 / RP6 / CN24 / RB6	AN6 / RB6	I3.3	15	AN9 / RB15
18	PGED2 / AN7 / RP7 / RCV / CN25 / RB7	RB7	LED-Org		
19	AVdd				
20	AVss				
21	AN8 / RP8 / CN26 / RB8	RP8 / RB8	TX2 to Xbee	19	RP5 / RA0
22	AN9 / RP9 / PMA7 / CN27 / RB9	RP9 / RB9	RX2 from Xbee	20	RP6 / RA1
23	TMS / CVref / AN10 / PMA13 / CN28 / RB10	RB10	Xbee-SleepRQ	12	RA10
24	TDO / AN11 / PMA12 / CN29 / RB11	RB11	Xbee-Reset	13	RA7
25	Vss				
26	Vdd				
27	TCK / AN12 / CTEDG2 / PMA11 / CRED2 / CN30 / RB12	RB12	USB-PowerSw Xbee-Status		ProtoV3 had
28	TDI / AN13 / CTEDG1 / PMA10 / CTED1 / CN31 / RB13	RB13	I5.0 USB-PowerSw		…
29	AN14 / CTPLS / RP14 / PMA1 / CN32 / RB14	AN14 / RB14	Xbee-Status I5.0		Need this to be
30	AN15 / RP29 / REFO / PMA0 / CN12 / RB15	AN15 / RB15	Isensors	27	AN8 / RC2
31	SDA2 / RP10 / PMA9 / CN17 / RF4	I2C2-SDA2	SENSORS		
32	SCL2 / RP17 / PMA8 / CN18 / RF5	I2C2-SCL2	SENSORS		

Pi	PIC24FJ256GB206(206,210) Pin Functions Available	Pin Function Used – V3	Application Assignment – V3	V2	V2 Func
33	RP16 / USB-ID / CN71 / RF3	USB-ID	J5-Pin4		
34	Vbus / RF7	Vbus	J5-Pin1		
35	Vusb				
36	D- / CN84 / RG3	D-	J5-Pin2		
37	D+ / CN83 / RG2	D+	J5-Pin3		
38	Vdd				
39	OSCI / CLKI / CN23 / RC12	CLKI	16Mhz TXCO		
40	OSCO / CLKO / CN22 / RC15	CLKO	T.P.		
41	Vss				
42	RTCC / DMLN / RP2 / CN53 / RD8	RD8	LED-Red		
43	DPLN / SDA1 / RP4 / PMA14 / PMCS1 / CN54 / RD9	RP4			
44	SCL1 / RP3 / PMA15 / PMCS2 / CN55 / RD10	RP3			
45	RP12 / PMACK2 / CN56 / RD11	RD11	LED-Grn		
46	DMH / RP11 / INT0 / CN49 / RD0	RD0	SensorPowerSw		
47	SOSCI / C3IND / CN1 / RC13	SOSCI	RTCC		
48	SOSCO / SCLKI / T1CK / C3INC / RPI37 / CN0 / RC14	SOSCO	RTCC		
49	Vcpcon / RP24 / Vbuschg / CN50 / RD1	RD1	LED-Yel		
50	DPH / RP23 / PMACK1 / CN51 / RD2	RP23 / RD2	SDspi-CS	36	RP19/RC3
51	RP22 / PMBE0 / CN52 / RD3	RP22 / RD3	SDspi-SDD	37	RP20/RC4
52	RP25 / PMWR / CN13 / RD4	RP25 / RD4	SDspi-Data	38	RP21/RC5
53	RP20 / PMRD / CN14 / RD5	RP20 / RD5	SDspiCLK	43	RP7/RB7
54	C3INB / CN15 / RD6				
55	C3INA / SESSEND / CN16 / RD7				
56	Vcap				
57	ENVREG				
58	Vbusst / Vcmpst1 / Vbisvld / CN68 / RF0				
59	Vcmpst2 / SESSVLD / CN69 / RF1				
60	PMD0 / CN58 / RE0		ThruHolePad		

Pi	*PIC24FJ256GB206(206,210) Pin Functions Available*	*Pin Function Used – V3*	*Application Assignment – V3*	*V2*	*V2 Func*
61	PMD1 / CN59 / RE1	CN59 / RE1	J6-Pin8		
62	PMD2 / CN60 / RE2		ThruHolePad		
63	PMD3 / CN61 / RE3	CN61 / RE3	J6-Pin6		
64	PMD4 / CN62 / RE4		ThruHolePad		

Changes in V3 Board/Schematic from V2

PIC24FJ256GB206 – connections

- Vdd/Vss caps still are placed under board despite uChip's recommendation to 'same-side' but still are close connections and now have power-via on 'other-side' of caps from pic.
- Mclr pull-up is still distant from pic, despite uChip's recomm. to keep all components assoc. w/it close. Current limit resistor r11 is. Cap c20 is assoc. w/reset button and distant. uChip suggests cap should be removed for programming but save arrangement worked for V2 board.
- Lans are .008 at pic's pins but moved to .01 outside (with exception of Vdd/Vss which are .01). Did this to improve/help some very tight routing esp. next to pin 1,2,3,64 where 2 lans are very very close. One trace could, maybe should be moved to bottom.

I2C vias are new smallest ones: .026x.008"hole….will they work?

RTCC / TCXO Both on bottom of board. Tcxo has enable jumper for when not used.

Vin Connector Moved away from L0, etc so that we can switch to screw-terminal version. Footprint same-old

Xbee Removed cutable/jumperable pull-option for sleepRq. retained that line because now available with xb-8062 firmware. Added an led and push-button for xb commissioning: allows switching modes dynamically (handy for setting up a sleeping network maybe). Added XbeePower jumper so it can be utterly turned off. Pic still decides when +enabled.

MAX3323 Moved to bottom of board to make more room. May not be needed and may make fab. more awkward. Removed cutable/jumperable null-modem option vias.

J6 – PIC Programming Header Retained wiring for programming side, and 2x5 arrangement. Disconnected Xbee related signals. Added some gpio and still have CN-button (CN11, RG9, RP27) on pin 10. Has cmos level signals. Added J3 for off-box sensor with control, routing signals from console port (232txrx) to pins 2,4 through jumper pad J3. Potential use: 9X radio modem but would need 232, or other radio.

GPS/Ublox Moved toward 'top-of-box' and toward side to make room and hopefully make antenna cable have more slack. Should for V3 we consider Terry's on-board/active-amp and passive antenna without any external? Ublox gnd permanently attached.

Holes added for batt.

Changed ldo cap C21 to 0603 and expanded size of filled planes to increase thermal sink.

J7-Ublox Now has both 'txrx' and 'i2c' to ublox. These are cmos level but could be used for aux. sensors/etc. Have not yet tried i2c on ublox, and with this pic probably won't need to.

microSD buffer moved to bottom. moved over to make more room. Power distribution moved to off sensor chain downstream of current monitor so we can determine that load also. Diode added to block back-flow when sensor power switch is off. Still need to revisit software to determine if it's possible to totally shutdown and prevent io line on spi back-bleeding.

ADC/Vref Added Vref U5 Ref3325 to stabilize internal adc ref. Retained the attempt at RC filter on all adc monitor channels. This may be wrong because the PIC24F family reference on the A/D sampling requirements says the total recommended source impedance be <=2.5k (Rsource + Rinterconnect). There is also the internal <=3k sample switch resistance. The sample/hold capacitor is 4.4pF and must fully charge before sampling begins to meet accuracy spec. For Imonitors, the MAX4372 claims output impedance of 1.5ohms which should be ok.

Note: ADC can run in sleep mode to help reduce digital noise if the trigger source is set to ensure this and the 'auto conversion trigger option can be used (AD1CON1,SSRC<2:0>=111. To use this the 'ADON bit should be sit in the intstruction prior to doing a PWRSAV. In IDLE mode you must have the ADSIDL bit=0 (AD1CON1<13>) set for it to continue in idle mode. With adc interrupts set, it'll wake the processor. Beware that if it's set=1 the adc will stop in idle mode and not resume a partially completed reading.

V2 Noise appears most related to vref, but also uChip recommends either a buffer or low impedance sources. Have not looked or thought about buffers….maybe too much cost/effort for value added. Check how proto board works with just vref.

USB OTG added, using D+,D-,USBID. Vbus provided either, as slave from bus, or as host through U12- MCP1252 charge-pump controlled by RB13 for shutdown. 'Pgood' not being used. Current monitor for +5 'host-mode' through U14 and adc AN14.

The USB connector is an issue. Putting a microA/B or mini right-angle won't have room for connector since they're normally used on board edge. For now I used a .1" header (Bulgin sells a 4" long .1" adaptor for either 'A' to host or 'B' for device) forcing to build our own cable. I've seen 1 right-angle mini-b plug also.

Spacing Issues Can the USB connection/tors fit? Sensor connectors and 'console/power' connectors same. Programming header moved. GPS header different but about same place. Clearance to remove GPS battery should be better but the J1 max3323 power header may still be issue. Expanded the board length a bit: will din connectors still be ok, etc?

Ferrites L3,L4 for I2C removed. Replaced by simple resistors for current limit protection.

The ferrites are/were intended for high-frequency noise suppression.

However, their characteristically high resistance (500ohms for the ones used)at I2C frequencies clobbered the signal.

Current limiting the port for potential short circuits (observed in the field from bad sensors) is more important. For both the processors used with V2 and V3 the max supply current is 18mA so a 200 ohm value was chosen.

TVS Diode vs Varistors

The varistors used in V2 for the I2C in particular had excessive capacitance even though in our application it worked. For driving longer lines or insuring better signals, a low capacitance ESD device is preferred even though the varistors potentially handle a bit more power. The 2nd V3 board proto will include roughly .006" pads on the lines as a Spark-Gap for higher pulses.

Issues: V3 Differences from other PIC/Board

MPLAB Error: Device ID doesn't match expected on ICD2 or 3; Programming problems

"Invalid target device id (expected=0x4104, read=0x0)"

The PIC24FJ256GB206 is much more sensitive to stray capacitance on the PGEC/PCED/MCLR lines. The spurious reset suppression cap, c20, on the reset/MCLR button may need to be jumpered out for programming. Similarly PGEC/PGED are routed to RS232/MAX3323 chip and it causes enough added capacitance to prevent device programming.

Fix: Jumper out RS232 lines from PGEC/PGED3

Notes: From the PIC forums Re IDC2 and 3.3v PIC24 failure 'rbreesems writes:

"Well, the configuration bit settings don't matter for ICD2 to PIC24 communication purposes, so it just comes down to the MCLR, VDD, GND, PGC, PGD lines.

Have you tried a different pair of PGC, PGD lines on the PIC24? Which ones are you using?

Is the cable from the ICD2 to the PIC24 short? There have been some documented crosstalk problems between the PGC and PGD lines in that cable because they run right next to other, some people put a small RC filter on these to slow down signal transitions to reduce crosstalk. They recommend a cable length of 6 inches."

and 'ohmite adds:

"If you are sharing your pgc & pgd pins that you are using to program with something else, you may not be able to program the chip. Try moving to one of the other (choice of three) pgc/pgd pin combinations or disconnect the other connections while programming. I have found they can be very intolerant of sharing the program pins with other devices. Also, if you have made a special programming cable (something other than the orginal ICD2 cable and normal ISCP connector), make sure your connections are correct and the length of the cable is not too long."

Error: CLKO monitored by scope messes processor

The scope probe may have enough added capacitance to effect the signal. In one case of a user who tried to interface with the ICSP (in circuit debugger) via ICD2/3 the scope probe helped the processor run by adding capacitance. In his case he added a 10pf cap to the signal to stabilize it.

Problem: I2C2 won't run / locks up.

On the 2nd prototype board the i2c wouldn't open and it locked up/reset. Removing the 'varisters' at least allowed the sensorON power to work…….maybe…..

MPLAB Error: Odd behavior with v3.25

Difficulties with timing, putchar, etc. were encountered during initial testing of the V3 board. These were "solved" by stepping back to MPLAB-C30 V3.23….

V3 – Timers:

Timer Period Calculation: Fosc*Count*PreScaler = Amount of Time for Interrupt to fire

External TCXO: Fosc=16,000,000/2 mHz, Period=.000000125 Sec/count

So for 1-Sec with Prescaler 256, Count = 8000000/256 = 31250 (0x71A2)

Timer1	Bresenham Clock	Priority-1
Timer2	InputChange Button	Priority-1
Timer3	Xbee	Priority-1
Timer45	GPS No-Lock-Timeout	Priority-1, 16-bit

V3 - PIC Hardware Serial Port:

UART1-3 are on Reprogrammable Pins (PPS)

Board:		
SensorNode	PIC Tx to Max3323 RS232	Pin11, RP18
	PIC Rx from Max3323 RS232	Pin12, RP28
	PIC Tx to Xbee	Pin21, RP8
	PIC Rx from Xbee	Pin22, RP9
	PIC Tx to GPS	Pin5, RP26
	PIC Rx from GPS	Pin4, RP21

BaudRate Generation: See specific chip data sheet and is similar to the version2 processor.

The 16-bit BRGenerator supports either sync or async modes. It controls a free-running timer which samples the i/o in either a rate of FOSC/32 (Standard) or FOSC/8 (High).

The Standard Calculation for the UxBRG register is: (FCY/16*BaudRate)-1. With bit BRGH=0 in the UxMODE register.

The High Mode has UxBRG = (FCY/4*BaudRate)-1. With BRGH=0

NOTE: **Use BRGH1bitHIGH. BRGH1bitSTANDARD only works for 9600,19200bps...i.e. don't bother.**

NOTE: 115200bps can only be used with the TCXO / 16mHz clock via 'USE_EXTERNAL_OSC'.

Otherwise all of the standard rates are available: 9600, 19200, 38400, 57600bps

PIC Programming-MPLAB Setup:

SensorNode and Repeater Boards include a 5-pin header for programming denoted J6:

J6 Pin	Description	Dual Use
1	Vcc – Usually +5 from MPlab ICD2	
2	PGC – Program Clock from MPlab	PIC-Tx to ADG736 (repeater)
3	PGD – Program Data	PIC-Rx from ADG736 (repeater)
4	MCLR – Pull-Up resistor provided on-board	
5	Gnd	

ICDx USB Driver path for PC:

c:\Program_files\Microchip\ MPLAB IDE/ICD2/Drivers/icd2w2k.inf

If you plug your ICD2/3 into a different USB port you may need to reload the driver which should be in the above directory. Don't use the windowz version, it'll ask for an install disk.

Directories used in search path:

Project-tab
Build Options
Project
Directories-tab Shows (Include, Output, Assembler, Library, Intermediate):
For MPLABC30-V3.23,V3.25
Include Search Path \c\mcc18\h
Library Search Path \c\mcc18\lib

'Project.mcw' Pop-up Window Shows the 'Source Files', 'Linker Script', etc.
Linker Script: right-click and add 18f2520.lkr (for the chip we're using)
\c\mcc18\lkr\18f2520.lkr
you can enter this by hand
Linker Scripts: /cygdrive/c/Program Files/Microchip/MPLAB C30/support/PIC24F/gld/p24fj64gb004.gld

Source Files: right-click to 'add'/'remove' files

MPLAB C30 Install Directory:

Should look something like this:
Windows7 c:\Program Files (x86)\Microchip\MPLAB C30\

MPLABC30 Build Tool Suite/Locations:

Install Directory: This partly depends upon your version of Windows and where you want to put them. There can be multiple versions such as: v3.25 or v3.31. The executables/etc. can be setup for either. With the 'old-default' it isn't obvious which version it holds.

c:\Program Files\Microchip\MPLAB C30 (XP / old mplab default)
c:\Program Files (x86)\Microchip\mplabc30\v3.25 (7 better, more obvious)
c:\Program Files (x86)\Microchip\mplabc30\v3.31

Project-tab "Set Language Tool Locations"
Registered Tools: Microchip C30 ToolSuite
Project-tab "Select Language Toolsuite"
Active Toolsuite: "Microchip C30 Toolsuite"
Toolsuite Contents: …

Location: "Install-directory"\bin

Executables:
MPLAB ASM30 Assembler: …\bin\pic30-as.exe
MPLAB C30 C Compiler: …\bin\pic30-gcc.exe
LIB30 Archiver: …\bin\pic30-ar.exe
MPLAB LINK30 Object Linker: …\bin\pic30-ld.exe

NOTE: For PIC18 compiler (other projects) :
For v3.23: c:\mcc18\bin\mcc18.exe (Was using this before)
…\mpasm\mpasmwin.exe
…\bin\mplib.exe
…\bin\mplink.exe
For v3.42: c:\Program Files\Microchip\mplabc18\v3.42\bin\mcc18.exe
(Kaiser's pic)

Programmer Setup:

Programmer-tab
Select Programmer: MPLAB ICD3
Settings
Communication: USB
Status: AutoConnect at startup, AutoDownload firmware if needed
Note the 'Run Self Test' tab
Power: Uncheck 'Power target circuit from MPLAB ICD2
Note: Power is +5vdc from the programmer. The XBee radios are tolerant only up to 3.6volts so don't run an 'xbee' board from the mplab
Program: Allow ICD2 to select memories and ranges
Program2: IMPORTANT: Preserve EEPROM on Program. This lets you keep permanent application setups without them getting erased between programming cycles.

Co**nfiguring the Device Operation**:

Configure-tab
Select Device = PIC24F64GB004 (V2) or PIC24FJ256GB206 (V3)
Settings
Workspace: 'Auto Save: Prompt'
Progm Loading: 'Clear memory before'
Projects: 'set all checked'
Configuration Bits: 'Configuration Bits set in code'
Note: These are for the configuration (held in eeprom) which can be set either by the MPLAB 'Configure' tab under 'Configuration-Bits' or 'Set in Code'. For set in code, you can #include "config_settings.h" and put the various words in: ie _config1 ..etc..

PGEDx/PGECx Selection (for multiple-option parts)s:

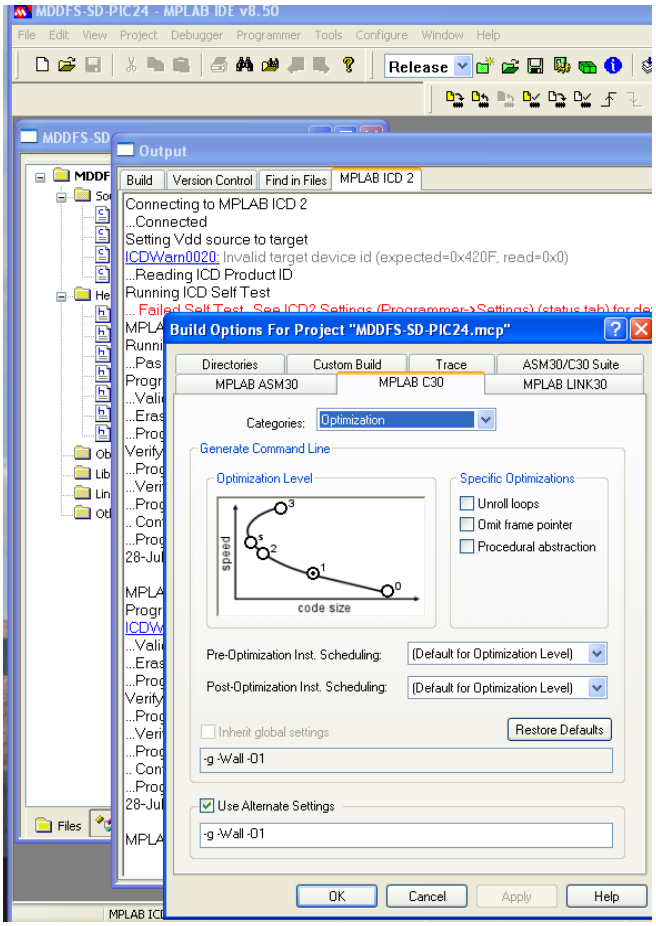
Designated by ICS Config. Bits in ICD

MPLAB C30 Compiler Code Optimization Levels:

In MPLAB-IDE: Select 'Project', 'Build Options', 'Project' – Select Tab 'MPLAB C30' and use the Pulldown selection under Categories: 'Optimization'

-O0 Default: Do Not Optimize
-O1 Optimize. Compiler tries to reduce size and exe. time
-O2 Optimize More (Full Licensed Version)
-O3 Optimize Even More (Full Licensed Version) turns on 'inline-functions' option
-Os Optimize for Space (Full Licensed Version)

See Interactive Help File: "hlpMPLABC30.chm"



Example: -O0

Total program memory used (bytes): 0x107b5 (67509) 25%

Total data memory used (bytes): 0x1c40 (7232) 7%

heap 0x1c40 0x800 (2048)

stack 0x2440 0x5bc0 (23488)

Maximum dynamic memory (bytes): 0x63c0 (25536)

Note: You may not see any big difference in code size but the speed, hopefully should be higher in some cases between –O0, -O2, -O3

MPLAB Errors:

Failure to recognize PIC / Board OR Device ID does not match:

This can happen when there is board corrosion esp. near or on the Xbee/connector because the pgm/pgc pins of the programming header are shared with other tx/rx uses.

"Failed to load C:\dir-filename.cof" This will stop you in your tracks. The .cof file is the binary file generated by the linker and ultimately what gets loaded into the processor. The pic30-bin2hex utility can be used to convert into hex file format use by device programmers.

Solution: Project-BuildOptions-MPLAB LINK30-General
Select: 'Keep all' to get rid of…
The 'Strip debugging info' button is what got me into trouble.
In the link command option controlling this is '-S'
"Omit debugger symbol information (but not all symbols) from the output file."

Note: This would appear to be related to using the 'Project - BuildConfiguration' Debug option versus 'Release version control. Setting 'Release' did allow inclusion of –S, however when toggling back and forth between release and debug the switch did cause issues; perhaps noting that the setting may not immediate effect passing of options to the build tools. You can't use –S with 'debug.' Select 'release' and remove –S to be sure.

Test Logic Issues The compiler is dumb……Put Parens around every logical segment:

Compiler/Linker:

Generating Compiler Errors/Warnings:

You can add compiler directives #if…#error…#endif, or similarly #warning

…example-1…

```
#if  I2C2BaudRate > 157
#error Cannot set up I2C2 for the SYSCLK and BAUDRATE >5.\
 Correct values in main.h and uart2.h files.
#endif
```

...example-2…

```
#if Nchannels*NsamplesEach > 16
#error Nchannels * NsamplesEach is limited to 16 buffer slots / interrupt. \
#endif
```

MPLAB LINK30 Options:

--heap see below in run-time env

--report-mem Causes linking to show the memory map…..very useful!

Language / Run-Time Environment:

See: "MPLAB C Compiler for PIC24 and dsPIC User Guide ds51284j.pdf"

Heap / Printf The heap is dynamically allocated memory used by malloc, calloc, realloc as well as standard i/o functions such as printf, etc.
It must be created at compile/link time via the linker build options:

```
pic30-gcc foo.c -Wl,--heap=512
```

At least 40-bytes per file 'open'ed, and 514 for files that need buffering. If you only use 'stdio' then the heap size must exist but can be 0. The linker allocates heap from unused data memory.

Specifying the Heap size when using MPLAB C30:
'project'-'build options' – 'project', 'MPLAB LINK30' tab, 'general' category; and notice the generate command line 'heap size:' you can fill in and have the command line created for you. NOTE you can also fill in the min stack size here. See *"16-Bit Language Tools Getting Started ds70094.pdf"* linker build options.

Printf/Stdio Also: For a PIC24FJ64GB004 and devices with multiple uarts, you must:
__C30_UART=2; // Defines which UART is stdio, default=1.
Note: This can be dynamically changed between ports.

While Statement must include operable commands within it's control.
This: while(!value) __delay_ms(5); // works
Not: while(!value); // doesn't, stream is blocked even if 'value' is 1

Type Casting **Very Important**
The compiler is very sensitive and it can be a factor with 16-bit representation. Example:
#define TIMED_SAMPLING 1
This: unsigned char Sampling_Mode=(unsigned char)TIMED_SAMPLING;
Not: unsigned char Sampling_Mode=TIMED_SAMPLING;
because the storage can be corrupted and not what you think it 'should be.'

To Install/Verify the MPLAB C30 Compiler for PIC24 MCUs

If using MPLAB IDE, be sure to install MPLAB IDE v8.10 or later before installing these tools.

To install the MPLAB C Compiler for PIC24 MCUs and dsPIC DSCs (previously MPLAB C30) tools, perform the following steps:

1. Locate the setup program on the compiler CD ROM.
2. Run the setup program.
3. Follow the directions on the screen.
4. If the installation program indicates that it is necessary, reboot your computer to complete the installation.
5. When the installation is complete, verify that the executable directory has been correctly added to your PATH (i.e., if you chose to install the tools in the default directory, c:\Program Files\Microchip\MPLAB C30, then ensure that c:\Program Files\Microchip\MPLAB C30\bin has been added to your PATH). From a DOS prompt, type: c:\>PATH

To verify installation, perform the following steps:

6. At a DOS prompt, go to the examples directory (by default, it is c:\Program Files\Microchip\MPLAB C30\examples\MPLABC30_Getting_Started)
7. Type: run_hello.bat
8. If the tools are installed correctly, the output should show the various steps in the compilation and execution process ending with the text: Hello, world!

License Key for MPLAB C30 Compiler Standard Edition

Purchase a valid license key from www.microchipdirect.com. (for ISFF MTI102550893)

To update an Evaluation or Lite edition of MPLABC30 to 'Standard': execute the following command line:

```
pic30-lm -ulicensekey
```

PIC24 Hardware Programming Notes:

Peripherals Support Library:

Library Source	c:Program Files/Microchip/MPLAB C30/src/peripheral_24F/libpPIC24F/pmc Where a copy of the source code is located
Documentation	c:Program Files/Microchip/MPLAB C30/docs/peripheral_lib Has a compiled help file: "Microchip PIC24F Peripheral Library.chm" and html files

WatchDog Timer:

The WDT is driven by the LPRC oscillator, so when the WDT is enabled so is this internal clock. The WDT setup is done through the Config1 settings (ie CW1 register) via pre-compiler or MPLAB

Nominal LPRC:	31kHz which feeds a prescaler of either 5 or 7-bits (1:32 or 1:128 divide-by). In 5-bit mode this ~=1mS, in 7-bit mode ~4mS.
Config1:FWDTEN	Can be set to always enable it (1), or if 0 allows software to dynamically set the RCON register SWDTEN bit to do so and/or disable it.
Config1:FWPSA	sets the prescaler to 5/7-bit mode
Config1:WDTPS	Post scaler for binary increments up to 32768 * 1or4mS mode rates (max=~131 Sec in 7-bit mode.
EnableWDT(x)	Either WDT_ENABLE or WDT_DISABLE allows software dynamic control
ClrWdt()	clears it so that it doesn't fire
Operation:	The watchdog does not have an interrupt vector for it. Instead it will either reset the processor, or wake operation after a sleep or idle PWRSAV instruction after which the sleep or idle bits must be cleared by the user.

Code Support Files:

iomapping.[c,h]	Used the define the i/o pin assignments and usage. Definition of '__PIC24FJ64GB004__' by MPLAB when using 24FJ64GB004 device controls which section is used.

CPU Clock/Oscillator Source:

Fosc	Processor Clock Frequency, Internal=8Mhz, External on Wisard=16Mhz which is the 'Primary Oscillator-EC' see below.
FCY	Instruction Cycle Clock Frequency = Processor Clock Source/2 = Fosc/2
Tcy	Instruction Cycle Time period = 2/Fosc, ex. for CLKI-16Mhz = 125nS. The minimum allowed is 62.5nS
EC	External Clock Input (0-32mhz) via the TXCO is the mode used for WisardV2. The PLL is not needed, and in this mode the following works:
CLKO	Pin 31 is used as a Fosc/2 signal in the V2 board. This is setup in the pre-compiler CONFIG2 word: OSCIOFNC_OFF // OSCO pin 31 is clock output (CLKO, to measure TXCO) Note that this clock will stop when in low-power 'IdleMode'

RTCC:

Secondary Oscillator =32768kHz crystal

Config Words	The RTCC can be used with either the Internal or External Clock. Depending upon Processor, the config setup may be in diff.words.
Drive Strength	SOSCSEL_SOSC = In high drive strength. SOSCSEL_LPOSC = low drive strength. High may needed on some boards, although usually not. On a few it was required for the rtcc function to work, however, on most boards it can cause a bit unstable/high-freq. oscillations*…..Ie look for this setting if RTCC problem. Normally use Low-Power.*
Register Ops	In order to write bits in the ctrl and value registers, you need to set the RCFGCAL.RTCWREN bit to 1. That should be done with a short assy. routine or the periph. support library may/should have a call or macro for this. It must be done in 1 instruction to let the above do it. This is also true for the RTCEN bit to be set, and to setup the chime/alarm register.
Idle Mode	RTCC stays awake
Sleep Mode	RTCC stays awake, but not peripherals and clock
Calibrating	The RCFGCAL can be adjusted by +127,-128 counts/minute (8bits) It is very temperature dependent. Note: do not try to put a scope directly on the osc. because it's impedance changes the circuit performance and frequency. For room temp offset cal: Write a test program (see my test stuff) that uses TIMER1 to monitor the error of the oscillator: 1) setup RTCC to fire at 1sec 2) run Timer1 at fastest rate and have it toggle an output pin

```
TIMER-1 SETUP:
    // Enable Timer Interrupt, Nearly highest priority level
    ConfigIntTimer1(T1_INT_ON | T1_INT_PRIOR_6);
    // Timer configured with Base-Rate:
    //      T1_PS_1_256 = ~7.81mS = ( (1/F-SOSC)*0x1 ) * 256
    //      T1_PS_1_64  = ~1.95mS = ( (1/F-SOSC)*0x1 ) * 64
    //      T1_PS_1_8   = ~.244mS = ( (1/F-SOSC)*0x1 ) * 8
    //      T1_PS_1_1   = ~.030mS = ( (1/F-SOSC)*0x1 ) * 1
    // Then Multiply above pre-Scaler by the Timer-Period value to determine
    // the actual interrupt rate, and toggled i/o^1 = 1/2 of that
    //      ie. PS_1 with Timer-Period 0x4000 (16384) ~= .50sec int, i/o^1 = 1sec
    //      ie. PS_8 with Timer-Period 0x400 (1024)   ~= .25sec int, i/o^1 = .5sec
    // So, for fastest toggling of i/o lines to measure with counter/scope....
    //  PS_1 ~.03mS (( (1/F-SOSC)*0x1)*1)*1 period ~= 1/32768Sec, i/o^1 = 16384hz
    //  Then we can measure the difference from ideal 32768hz
    // Continue running timer in idle mode, source is SOSC, RTCC secondary oscillator
    OpenTimer1(T1_ON | T1_PS_1_1 | T1_SOURCE_EXT | T1_SYNC_EXT_OFF | T1_IDLE_CON |
                T1_GATE_OFF , 1);
TRISDbits.TRISD9 = output
//***************** ISR for Timer1: RTCC Freq. Output **********************
void __attribute__ ((interrupt,no_auto_psv)) _T1Interrupt (void)
{
   T1_Clear_Intr_Status_Bit;
   LATDbits.LATD9  ^= 1;    // toggle io lines
   LATDbits.LATD10 ^= 1;
}
```

3) monitor the output pin with the scope/counter

4) Run and accumulate for a reasonable amount of time (min-ish)
5) Compute clock errors/minute
    a. (32768-measured)*60=error count
    b. Set RTCC RCFGCAL register bit0-7 =error-count/4

I2C:

Reference Files: /cygdrive/c/Program\ Files/Microchip/MPLAB\ C30/…..
src/peripheral_24F/src/pmc/i2c/ Many support functions here
src/peripheral_24F/include/i2c.h

FS – File System:

Reference Material: Locations of some help files on disks:
/cygdrive/c/Microchip\ Solutions
/net/isf/isff/picwork/Microchip\ Solutions
Compiled Help Files are under these directories at:
Microchip\ Solutions/Microchip/Help/MDDFS Library Help.chm
AN01045b:
ImplementingFileI/OFunctionsUsingMicrochipsMemoryDiskDriveFileSystemLibrary

Files Used: FSconfig.h Contains Options to Configure the library firmward
Such as: 'ALLOW_FILESEARCH', 'ALLOW_DIRS'
These config. settings will cause modules to be loaded, increasing or decreasing the program size
FS_HardwareProfile.h Declares what the hardware interface is and sets which pins are assigned to SPI, CS,SDD,Data, etc.
FSIO.h

```
// Summary: Contains file information and is used to indicate which file to access.
// Description: The FSFILE structure is used to hold file information for an open file as it's being
                modified or accessed.  A pointer to
//          an open file's FSFILE structure will be passeed to any library function that will modify
                that file.
typedef struct
{
   DISK   *      dsk;          // Pointer to a DISK structure
   DWORD         cluster;      // The first cluster of the file
   DWORD         ccls;         // The current cluster of the file
   WORD          sec;          // The current sector in the current cluster of the file
   WORD          pos;          // The position in the current sector
   DWORD         seek;         // The absolute position in the file
   DWORD         size;         // The size of the file
   FILEFLAGS     flags;        // A structure containing file flags
   WORD          time;         // The file's last update time
   WORD          date;         // The file's last update date
   char          name[FILE_NAME_SIZE];      // The name of the file
   WORD          entry;        // The position of the file's directory entry in it's directory
   WORD          chk;          // File structure checksum
   WORD          attributes;   // The file attributes
   DWORD         dirclus;      // The base cluster of the file's directory
   DWORD         dirccls;      // The current cluster of the file's directory
```

} FSFILE;
mSDdatafile is the handle used in the code

ADC:

MPLABC30/support/peripheral_24F/adc.h

Sampling — Switching time for the channel to be connected through the sample-hold reg.

Conversion — Time for the digitization of the analog voltage

ADC Conversion Clock — 'Tad' 12 cycles are needed for a conversion and 3 for sampling to be stabilized (important when in manual triggering).

CONFIG3 sets up the Tad based on a multiple of the Instruction Clock Time Period, Tcy when using the system clock. Otherwise it has a separate RC oscillator that should be used if sampling during Sleep mode is needed.

Tad1 = 12*Tcy, ex 16Mhz CLKI = 12*2/Fosc = 1.5uSec

Peripheral Library: The peripheral library, referenced above includes a variety of functions and macros that can be used for controller ADC operations. These may include for example setup such as these:

```
unsigned int channel,config1,config2,config3,configport,configscan;
// Configure adc
config1= ADC_MODULE_OFF |       // Remains off until turned on
        ADC_IDLE_STOP |         // does not run when cpu is in
                                idle/sleep
        ADC_FORMAT_INTG |       // Integer output
        ADC_CLK_AUTO |          // Internal counter ends sampling and
                                starts conversion: auto-convert
        ADC_AUTO_SAMPLING_OFF; // Sampling begins when the
                                SAMP bit is set
config2= ADC_SCAN_ON |          // enable scanning
        ADC_INTR_16_CONV ;      // Interrupts at completion of each
                                16 sample/conversions
config3= ADC_CONV_CLK_SYSTEM |  // system clock for
                                conversions
        ADC_SAMPLE_TIME_15 |    // sample time for auto setup to
                                15 Tad cycles
        ADC_CONV_CLK_1Tcy;      // conversion clock 1 Tad cycles

configport = AD1PCFG;       // Uses AD1PCFG we did in 'setups'
configscan = ADC_SCAN_AN11 | ADC_SCAN_AN10;
                                //channels10,11
OpenADC10(config1,config2,config3,configport,configscan);
CloseADC10();     // closes adc and disables interrupts....function call to:
                  // AD1CON1bits.ADON = 0; IEC0bits.AD1IE = 0;
                  // IFS0bits.AD1IF = 0;
```

However, Direct Register Manipulation seems better than incurring function call overheads, etc….

```
AD1CON1 = 0x20E4;   // Configure sample clock source and conversion
                                trigger mode.
                  // Module starts up in off state (AD1CON1bits.ADON =
                                OFF
```

```
                    // No operation in Idle mode (ADSIDL=1)
                    // Output Format Integer (FORM<1:0>=00)
                    // Auto Conversion
                    // Auto Sampling
                    // S/H in Sample (SAMP = 1)


AD1PCFG = 0xF0FF;   // Configure A/D port....make sure unused remain
                                off or else i2c/pps won't go
                    // AN8-AN11 input pins setup analog, all others are digital
                                i/o
                    // This is same as doing a =0xFFFF and then these:
                    //  AN_Vin_CHAN = ANALOG_MODE;  // ch11, Wisard
                                V2 Vin
                                divider....AD1PCFGbits.PCFG11
                    //  AN_Iin_CHAN = ANALOG_MODE;
                    //  AN_I3_CHAN = ANALOG_MODE;
                    //  AN_Isnsr_CHAN = ANALOG_MODE;
AD1CSSL = 0x0F00;       // Channels 8-11 configured for scanning, in that
                                order!...
AD1CON1 = 0x00E0;       // Continue in Idle Mode; Integer Format;
                                Internal Counter ends sampling and
                        // starts conversion...auto-convert; Sampling begins
                                when SAMP bit set
AD1CHS = 0x000B;        // AN11 selected for sampling
AD1CON3 = 0x0F00;       // SystemClockUsed, Sample time=15Tad,
                                Tcy*1 Conversion Clock
AD1CON2 = 0x043C;       // Set AD1IF after every 16 cycles with
                                scanning turned on
AD1CON1bits.ADON=1;     // Turn on ADC
```

Power Saving Modes:

Idle Mode: mPWRMGNT_GotoIdleMode();
defined asm PWRSAV #1, Clock remains active,
so peripheral interrupts are still enabled and active for wake, ie
Wakeup from Interrupts (enabled), wdt, reset
Operation begins here or within Interrupt handler
This is the mode to use if you want to keep 'uart' ingest ie incoming operator commands or other i/o running
It is what's used in the Wisard 'main'

Sleep Mode: mPWRMGNT_GotoSleepMode();
defined asm PWRSAV #0, Clk inactive, i/o locked and
any periph using Clk are disabled (uart,i2c,...)
but LPRC clock remains active for DSWDT or RTCC,
To do this properly, 1st setup/enable dswdt, rtcc,
change-notification, dsboren, save critical context in dsgpr0/1
then enable dsconbits dsen and issue this command above
Wakeup from PwrOnR, CN, MCLR, RTCC, ExtInt0, DSWDT
// This is the mode to use for deeper sleep where the deep-sleep watchdog is still available but not peripherals or the 'normal' watchdog for example.

Determining Wakeup: wakeup_source = PwrMgnt_WakeupSource();

Timers / Interrupts:

Timer Period Calculation: Fosc*Count*PreScaler = Amount of Time for Interrupt to fire
External TCXO: Fosc=16,000,000/2 mHz, Period=.000000125 Sec/count
ie for 1-Sec with Prescaler 256, Count = 8000000/256 = 31250 (0x71A2)
Internal 8mHz: Fosc=8,000,000/2 mHz, Period=.00000025 Sec/count
ie for 1-Sec with Prescaler 256, Count = 2000000/256 = 15625 (0x3d09)

Timer Rates: Fosc * Count * Divider
For 8Mhz internal Osc. the Fosc=4000000mHz = .00000025sec/tick

Timer1: Clock-Scheduler for Bresenham/TCXO.
Timer1 can run in Idle mode.

Timer2: CN button
16-bit timing to determine what option: (scani2c, switch output port between xbee/sio,)

Timer3: Xbee wait-for-reply timeout if no reply from radio
16-bit

Timer45: GPS – Failed Sync Timer
32-bit
NOTE: With the current source, the standard 'Open/ConfigTimer45' library functions are not working. Instead the wisard code had to be modified to setup timers 4 and 5 by direct register accesses. When using these in 32bit mode, Timer 4 is setup in 32bit mode, the counter registers PR4,PR5 are given lsb,msb counts respectively and the actual interrupt occurs under 'Timer5'. Fortunately the approach appears fairly generic between other processors in the 24 family.
NOTE: A secondary approach would be to setup timer4 for 1 sec (in 16bit mode max is about 4sec) and use a countdown for the seconds of gps run time before a failed to acquire shutdown is initiated.

INT1 GPS-PPS, ISR found in gps_functions.c
If RTCC is used it turns the RTCC back on and flags it's functions.

INT2: Xbee-Sleep Mode 'Awake' triggers 'sampling, ISR found in Wisard

GPS – U-BLOX Notes

Binary Command Examples

PIC24FJ64GB004	*Functional Assignment*	
GPS Reset	B5 62 05 01 02 00 06 04 12 3B	
	B5 62 06 04 04 00 00 00 08 00 16 74	Controlled GPS STOP
	B5 62 06 04 04 00 00 00 09 00 17 76	Controlled GPS START
	B5 62 06 04 04 00 01 00 09 00 18 7A	Warm Start
GPS Revert to last saved config.	B5 62 06 09 0D 00 00 00 00 00 00 00 00 00 FF FF 00 00 07 21 B7	

U-Center:

CFG-View: This is where to go for sending or polling the gps configuration.

TimePulse:

Wiring: The timepulse from pin-3 of the NEO5/6Q is connected to PIC pin 2, RP22/RC6.

Interrupt: The PIC is programmed for reprogrammable pin RP22 to be INT1, with it's dedicated predefined interrupt handler name _INT1Interrupt(void). The interrupt can be configured for rising or falling edge, as can the ublox signal. For now, both are set to 'rising edge'

TIM-TP This is the ublox message type for setting up the timepulse signal's length, rate, rising/falling-edge, etc. It controls whether to always send the pulse even if there is no fix, or to only send when there is a fix. For 'now' in testing it is setup 'always' with the clock source being 'UTC.' This message also controls:

Delays The TIM-TP can include delays for the RF processing in the ublox, Antenna to ublox, and 'user-application-delay' for signal delay plus processing delay. These adjust when the pulse is fired. It may be possible to measure these delays, and adjust the pulse length/edge so that it fires sufficiently after the GPRMC message is received and processed such that the interrupt will trigger a correct setting at the 'tail-end' …or not. For 'now in testing' there is no user or antenna delay..ie pulse occurs at UTC time.

Antennas / Testing:

Performance Tests: Use U-Center and the "Sky View" tool for testing outdoors for the antenna performance. See Application Note:

Power System:

Comparison of PIC24 Processor Power Consumption Specs:

Operating/(Idle) Typ. Current	*FJ64GB004*	*FJ256GB206 (206,210)*	*FJ256DA206 (106,206)*	*FJ256GB106 (106,108,110)*	*HJ256GP206A (206,306,506,210,310,510,610)*
4MIPS, 3.3V, 25C, Vreg	2.9 (.75)	3.0 (.6)	3.0 (.6)	4.3 (1.1)	N/A
16MIPS, 3.3V, 25C, Vreg	11.3 (2.9)	12 (2.3)	12 (2.3)	18.2 (4.4)	38/45 (5/25)

Component Voltage Limits:

Nominal Specifications for various parts used in wisard:

Component Normal Vcc range:	*Vcc range*	*Icc mA*		*Comment*
Xbee OEM Radio	2.8-3.4			
9Xtend OEM Radio				
PIC18F2520	2.0-5.5			V1 Sensor Nodes, Sensors
PIC18F26J11	2.0-3.6			V1 Repeater
PIC24FJ64GB004	2.2-3.6	11.5mA		Operating current, idle=3mA … 16MIPS=32Mhz Osc…..Expect ½?
MCP9800	2.7-5.5V	.2		I2C Temperature Sensor
MAX3323	3.0-5.5	1mA	1uA shutdown	RS232 Transceiver
MAX4372	2.7-28V	.03mA		Current Monitor/
ADS1112/1115	2.7-5.5	.24mA		ADC
MC74VHC1GT125 Buffer	3-5.5			
MCP1703 LDO	2.7-16		.625V dropout at 250mA	Used for USB power to board, V2...OLD
SA57000 LDO	.5-6.5		55mV Dropout at 50mA	CapFree, Used for 3V on GPS, V2
MIC94065 PwrSw	1.7-5.5		.002mA	

Power Consumption / Solar Charging:

Protection: F1 is a resettable fuse situated on the +12VDC input. It is nominally rated for hold current at .2A , trip current .4A within .05Sec at maxcurrent 8A. These devices trip point changes with respect to ambient temperature, input current and rated values are at 25C. At 60C expect ~.14A hold with trip .32-.5A. At 0C .25A hold, trip, .45-.8A roughly.

Measured Loads (From Version1 and 2) during testing nominally show the following:

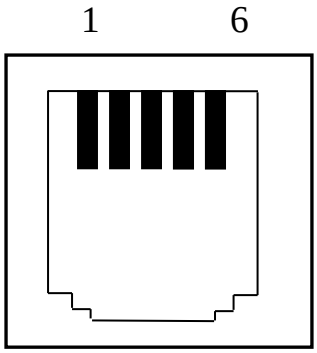
Component	mA	VDC	mWatts Load	Comment
Xbee Radio	50	3.3	~165mW	The baseline Rx power is most significant. Max Tx power makes essentially no differenct.
Ublox GPS	~64	3.3	~210mW	Turning it on/off and measuring supply
microSD			basically nothing	
Pic / SensorBoard Version1.0	9-14	3.3	30-46	Seems excessive, based on residuals. May still have MAX3323 enabled on these with nom. 1mA
Version 2.0	~6.8	13.0	88mW	Peak, No GPS, Sensors, microSD with TXCO serial out
	13.0	3.3	42.9mW	Peak, No GPS, Sensors, microSD with TCXO, serial out
	~5	13	65mW	Peak, No GPS, Sensors, microSD with RTCC, serial out
	6.4	3.3	21.1mW	Peak No GPS, Sensors, microSD with RTCC, serial out
Per Above: DC-DC			~45mW	At low power, Efficiency 50%ish See below, goes up at higher loads to ~75%
Per Above: TXCO vs RTCC			~23mW	
Sensors	13-17	3.3	43-56	Rough Total for sensors….
	1.6-4.0			Saw an occasional 15mA!!!!VERIFY ME
ft	1.3-4.5			….ditto….with even a 45mA
Version1 Total Consumptions measured			***310 – 345mW***	***High Load for 'Wireless Sensing!' Note 50%-Xbee, 22%=SG4, 15%=sensors, 14%=board. Plus….***
Version1 Actual Consumption			***390 – 425mW***	***Nominal efficiency of DC-DC switcher improves a bit at higher loads, 75%ish***

Version 2 board uses a commercial DC-DC module: Vinfinity V7803-500 versus my home-grown version. Efficiencies are roughly the same for either.

The Solar Charging System uses either a 5 or 10W PV panel with

Wiring & Cables:

MPLab PIC Programmer Cables:

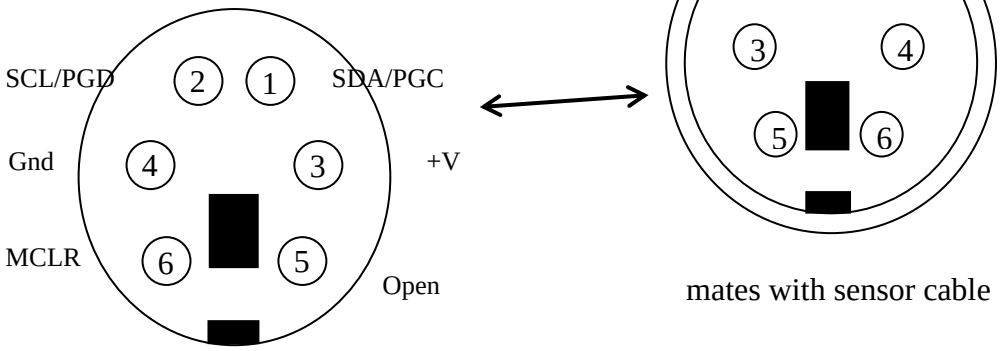


Looking at MPLAB ICD 2 RJ11 Jack Pinout (Female Recepticle)….b uild cable with male plug

Jack Pinout of MPLAB programmer for PIC

RJ11-Pin #	Signal	MiniDin6
1	NA	
2	PGC	1
3	PGD	2
4	GND	4
5	Vdd	3
6	MCLR	6

Looking at **Female Recepticle (socket) MiniDin-6 From MPlab ICD3 for Programming Sensors**



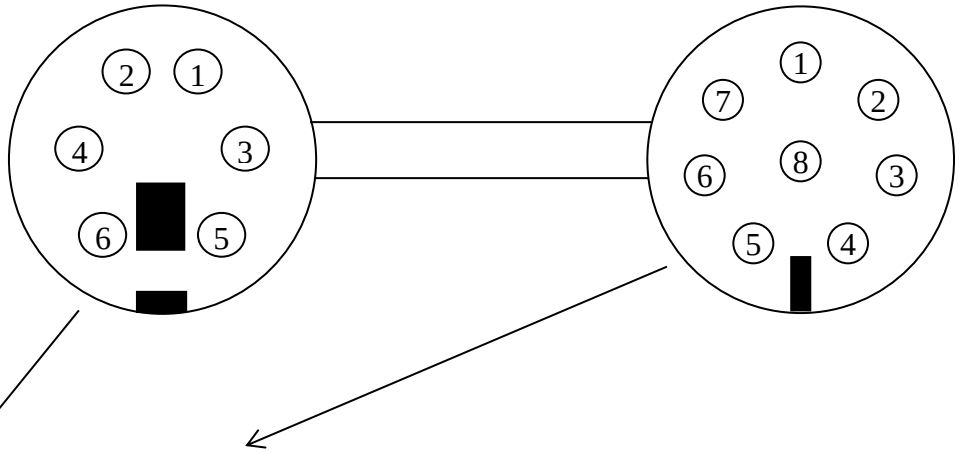
mates with sensor cable

Wisard Sensor Male MiniDIN connector

WhiteBox Console Cable (Serial/Power)

Front View
MiniDin-6 Connector (Male-Pins)
Plugs into Sensor/Repeater boards

Front View
Bulgin Connector (Male-Pins)
Plugs into ADAM Console Port



MinDinPin #	BulginPin#	Signal	(wire color..maybe)	DE9 (for ref.)
1	5	Rx, (From board to 'PC')	white (red)	2
2	6	Tx (To board from 'PC')	gray (orange)	3
3	1	+12 supply	yellow (brown)	
5	1	+12 supply	orange (black)	
4	8	Gnd	violet (yellow)	5
6	8	Gnd	red (green)	
case	8	Gnd	shield	

Sensor MiniDIN Cables

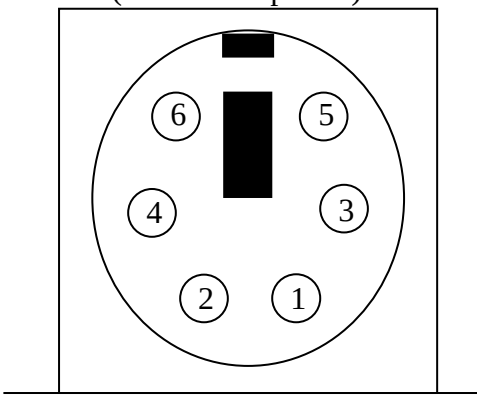
The sensors can be built with either a MiniDin-6 or MiniDin-7 male ended cable. Solder holes are provided via 'J1' on the sensor boards for the other end.

J1 Pins On Sensors
(solder holes)

Sensors J1-Pin#	Cable Pins MiniDin6	MiniDin7	Signal	Programming
1	1		I2C-SDA/Serial-Tx	PGC
2	2		I2C-SCL/Serial-Rx	PGD
3	3		Vin ("+12"supply)	Vdd
4	4		GND	GND
5	5		RB2 / Int2	n.c.
6	6		RE3 (input only)	MCLR
	7		+3.3 (LDO output)	n.c.

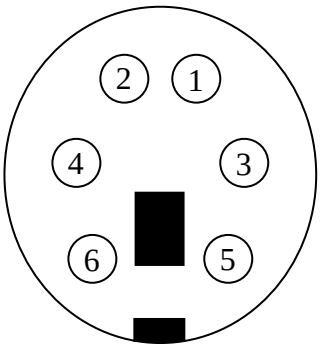
he J1 connections on all Sensors show signal labels only, not pin#'s on the silk-screen.
Pin designations shown on schematics only.

Looking at **Female Recepticle (socket)**
6 pin Mini-DIN
On SensorNode (WhiteBox) host board
(Mounted top Side)

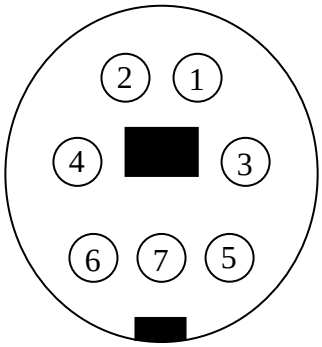


Looking at **Male Plug (pins) Cable from Sensors**:

MiniDIN-6



MiniDIN-7



Pin #	SensorNode Board Signal (WhiteBox)
1	SDA (Red)
2	SCL (Org)
3	3.3v (Brown)
4	GND (Yellow)
5	GND (Black/Int2,RB)
6	GPIO pad (Green/Mclr)

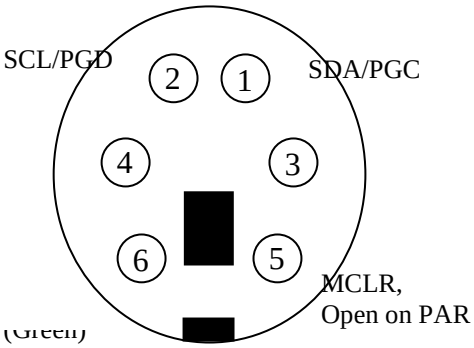
NOTE: Pin-5 (bussed GPIO), and Pin6 (individual GPIO) are open pads that can be soldered to PIC lines RA0-2 or RB4 that also have pads available for specialized control

Difference Between PAR-Wand and Wisard Sensor Cables

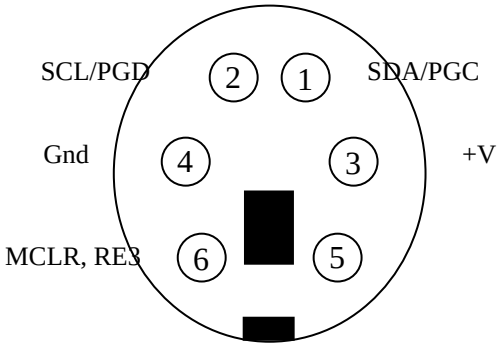
The PAR-Wands from Niwot-2008 were based on the same pic processor as most wisard sensors, however there were some wiring differences on the 6-pin din connector, as shown.

Looking at **Male Connector** from 'sensor'

PAR Sensors

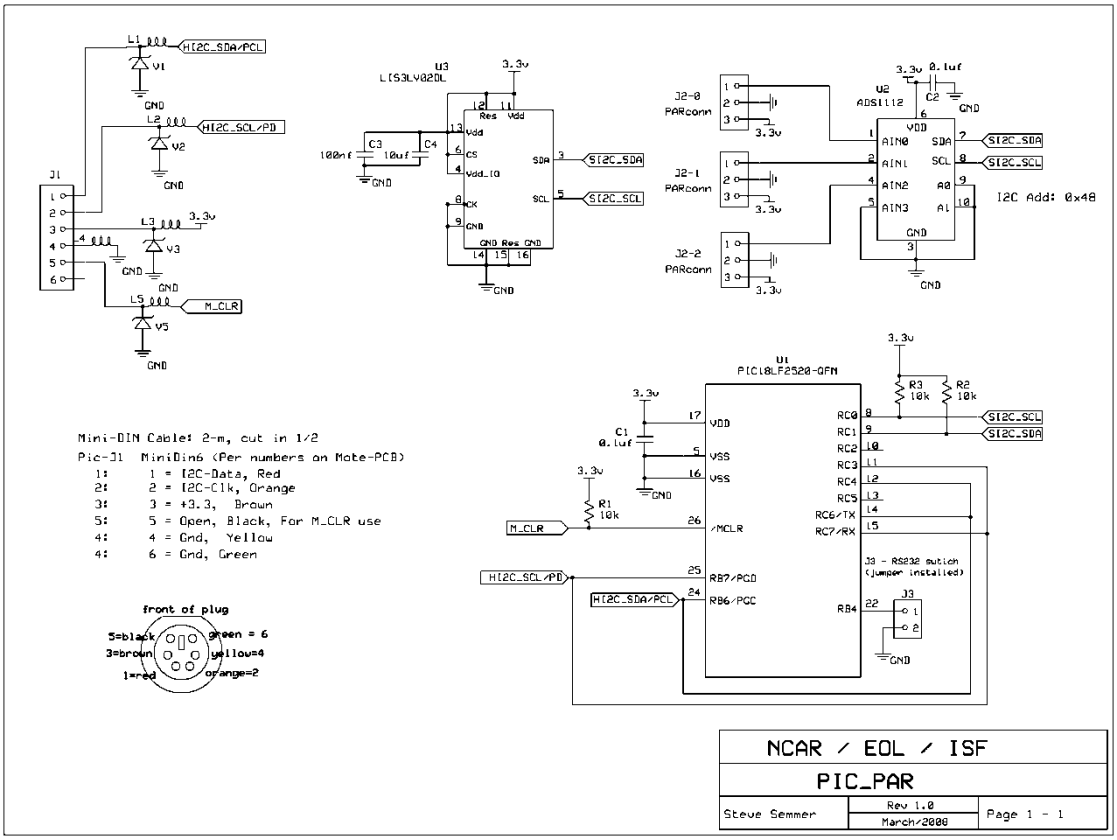


Wisard Sensors



Par Wand Layout

Par Wand Schematic



How To Pot Bulgin Connectors:

The Urethane potting materials come in small packets that have both mix and hardener.
Materials are ordered from 'Hardman'
Use nitrile gloves and 2 0mil syringe.
Mix a small amount (2 or 3 packets) and work fast: 5-minute setting time.
Use a cup and stiring stick to mix the materials.
Keep the connector oriented down because the material drips. You only need to cover the exposed pins, wires and holes and don't need to fill the entire back-shell.
Suck the material up into the syringe rather than trying to fill it from the back side.
Use a heat gun to warm the back-shell before pushing it on or else it may break.

.

PCB Board Design / Manufacturing

Board Design / Layout Software:

Schematics, Layouts and Board Manufacturing have been done using www.ExpressPCB.com

Boards are: 4-layer, 1.25oz copper.= Copper Thickness .0017”
.062” Laminate: FR-4 Epoxy glass: Dielectric Constant = 4.2-5.0
Stacking: TopSignal/Silkscreen, Ground, Power, BottomSignal
Assuming equidistant: ~.020” between layers

Milling: Board perimeters are cut to shape using a 0.093" router bit. Slots in the perimeter must be 0.100" or wider. Narrower slots will not be milled correctly. Routing slots or holes in the interior of the board are not offered.

Board Edge Clearance: .020" The edges of the board are cut with an accuracy of ± 0.015". A minimum of 0.020" blank space is recommended between the perimeter and all features on the board. Traces placed closer than 0.015" to the board's edge may be routed off.

PCB Component Stuffing and Assembly

Gerber Files: Assembly companies normally need: Gerber Files, XYRS and Panelized Paste Files; plug BOM and assembly notes.
The Bill Of Materials (BOM) can begin using output from most Schematic capture programs including expresspcb; however the generated files are far from adequate and need to include component identifications, packaging, ordering information and any *special assembly notes such as ‘Ublox is no-wash part*.’

Stencils: High-end, quantity manufacturing requires Stencils to apply paste to the PCB before components are automatically picked and placed for oven curing. Using expressPCB these cannot be made directly without the XYRS and Paste files.
The files needed to create stencils can be generated for a cost by most assembly houses for an additional cost. Otherwise, higher end Schematic Capture / Layout programs such as Altium and Eagle have the capability to generate these files.

AAPCB This company has been used for low quantity builds by ‘DropSonde’ group. They can make their own component templates for automated pick-n-place without needing tape and reels which are typical of large builds but impractical for small runs. Instead they can handle either ‘Cut-Tape’ (and prefer it), or loose parts.
Advanced Assembly
20100 East 32nd Parkway, Suite 225, Aurora, CO 80011
aapcb.com

ExpressPCB only provides Gerber files and only after boards have been manufactured by them. After that you can order them and actually have boards made elsewhere (maybe). They say:
We can not generate solder paste, stencil or Centroid files. Typically what our customers do is order gerber files from us which can be edited into other formats. I recommend that you speak with your board assembly house (or the company making your solder paste screens) to verify that they can work with these files.

You can order gerber files by email for PCBs that we have previously manufactured. The cost of these files is $60. This fee will be billed to the credit card number used in the original order for the boards. You will receive these gerber files in RS-274X format:

+ Top silkscreen layer
+ Top soldermask layer
+ Top copper layer
+ Bottom copper layer
+ Bottom soldermask layer
+ Drill file
+ 4 layer designs also include the 2 inner layers

To order the files, send email to support@expresspcb.com with this exact information:

1) Write a note indicating that you would like to purchase gerber files.
2) Acknowledge in your email that you understand there is a $60 fee for each set of gerber files that you order, and that the fee will be billed to your credit card.
3) Specify the order number of the previously manufactured design by noting our original 8 or 9 character order number. Our order numbers have the form ABCD-1234-E.

We will generate the gerber files using the original PCB design file associated with your order number. We will then email the gerber files back to you as an attachment. The email will also include a receipt for the $60 fee marked "Paid in full". Typically it takes one 24 hour business day to fulfill the order.

Thermal Considerations in PCB Board Design and Layout:

Care must be taken to provide adequate component cooling under load for power components especially LDO's and other dc-dc devices. Manufacturer's typically specify *junction temperature* in addition to *output current capacity*. Care must be taken to avoid heat-dissipation causing temperature rise above specified limits to avoid damaging the part, and perhaps worse, having their internal safety-monitor shutting them off disconnecting your circuit or even worse: component total failure.

Adequate PCB area must be provided for the anticipated operating temperatures to provide this cooling, or other heat sink techniques in 'big-part' situations. Often we simply provide an area that 'looks good and fits' without doing a specific calculation. However a second factor is what can happen when attaching the part. If the area is too large then *when hand-soldering the component can also be over-heated* and be damaged or just as likely not have a good joint. Manufacturer's generally suggest controlled 'Re-Flow-Soldering' (ie oven) over hand-attachment.

The following sheets provide an accurate calculation of PCB thermal pad area sizing: See:
PCB_Thermal_Copper_Area_3_CalculatorforThermalMgmt.xls
LDO_PowerDissipationCalculator_Limits.xls

Coating PCB Boards:

There are significant differences between types of coating materials and depending upon application, one may work better than others:

1) Epoxy – Heavy Duty, UV, Good for durable, out-door applications. This is the material we have been using for the Wisard Sensor boards, soils and such that will be exposed to water and abrasion.
Supply: White Rapid Coat Aircraft Paint. Jeff Bobka supplies this stuff. It has a shelf life and cannot be stored for long term.
Mix: 1 to 1. This is critical for good results
Stir / Rest for 15-Minutes. You'll probably see some bubbles.
Stir – Dip Boards
NOTE: if you're going to spray the paint on instead of dipping the 'Rest for 10-Minutes not 15'.

All Other Coatings below are 'Light Duty' and 'Non-UV'

2) Acrylic Do not used for exposed boards. Light duty for protected locations. This coats well and doesn't bubble. It is good for use inside sensor 'clam shells.'
Chemtronics – KonForm spray.

3) Urethane Similar to Acrylic, but hard to coat with this and it bubbles easily…so don't use it for the most part. For this one you must have the board extremely clean

4) Silicone Good for temporary light duty use. It is the only one that can be cleaned off if needed

Nominal Current Capacity for PCB trace sizes.

Current capacity depends on board and component temperature. In general current capacity goes down with higher temperature for lans/wire, but improves for components that are limited on internal junction temp/heat sinking.

For 1.25-Oz copper traces (=1.68 mil thickness, used by ExpressPCB) the approximate current capacity is shown (for ~20degC temperature rise). Elevated temperatures reduce trace capacity.

Trace Width	Sq.Mils(1.25Oz)	20degC temp Rise
.010"	0.17	0.3 Amps
.015"	0.25	0.4A
.020"	0.34	0.7A
.025"	0.42	1.0A
.050"	0.84	2.0A
.080"	1.35	3.2A
.100"	1.68	4.0A
.150"	2.53	6.0A

Chart of PCB Current Capacities vs Temperatures

(For use in determining current carrying capacity and sizes of etched copper conductors for various temperature rises above ambient.)

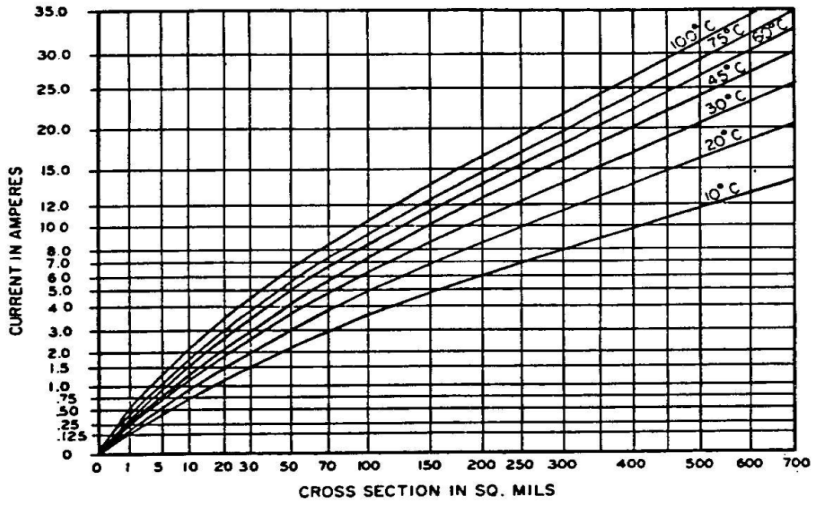


Figure A External Conductors

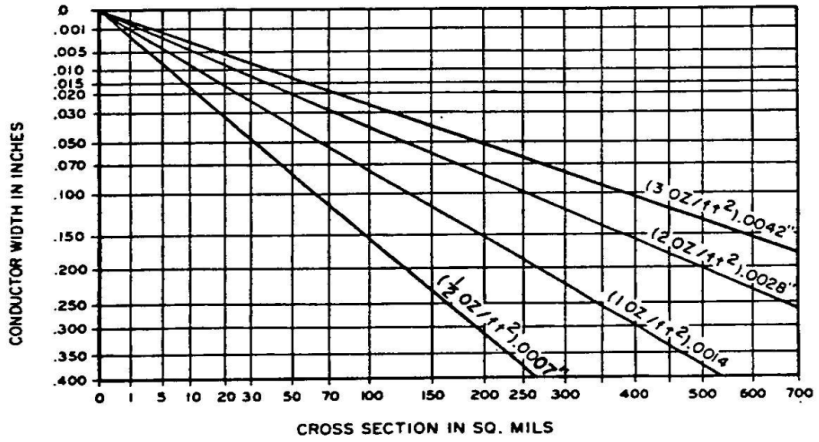


Figure B Conductor width to cross-section relationship

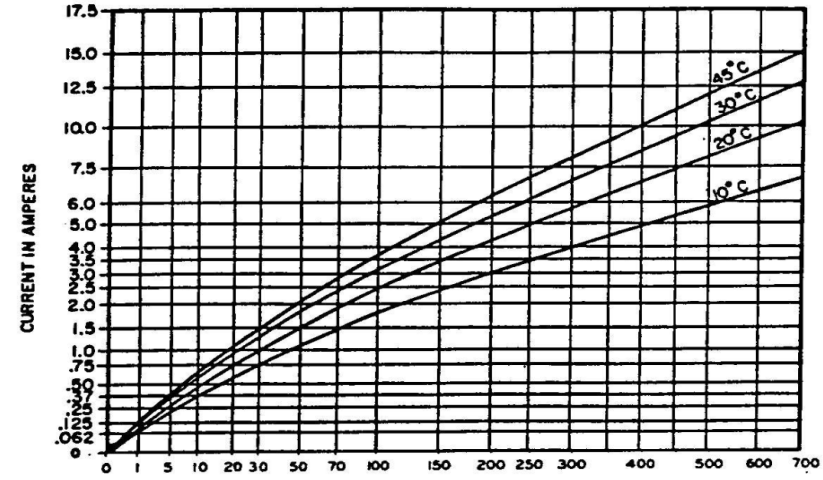


Figure C Internal Conductors

Figure 6-4 Conductor thickness and width for internal and external layers

Design: Tank Capacitors

On boards with power traces running longer than six inches in length, it is suggested to use a tank capacitor for integrated circuits, including microcontrollers, to supply a local power source. The value of the tank capacitor should be determined based on the trace resistance that connects the power supply source to the device, and the maximum current drawn by the device in the application. Select the tank capacitor so that it meets the acceptable voltage sag at the device. Typical values range from 4.7uF to 47uf.

Design: PCB Trace Impedances

In general, resistance and impedances on PCB traces are extremely small and can often be largely ignored. Resistance effects both DC (voltage sags, recovery) and AC characteristics, but impedance (capacitance/inductance) influence only AC and are consequently frequency dependent factors. For the most part in AC situations the trace behaives like a LC circuit that will effect switching signals/transition times. This can be an issue if the signals are deteriorated enough to effect reliability. For example if a signal doesn't reach high enough levels or adequate settling time for detection. Worst case is marginal operation that can introduce 'random' errors. There are several on-line calculators for trace resistance, capacitance and inductance.

Resistance relates to trace cross-section (PCB express uses 1.25oz or .017mil thicknesses), length and temperature. As temperature goes up so does resistance. For copper it is generally very small, so unless current draw and fluctuations are significant for a IC, or temperatures a high, the voltage sag may not be a factor. It may not be obvious how the current changes on the IC, so recommended tank-capacitors should be use, or larger.

Inductance is dependant and inversely related to trace width/thickness. It is unevenly distributed throughout the trace and is generally uncontrolled. Vias also introduce some inductance.

Capacitance is dependant upon trace width/thickness, dielectric material, and distance from the signal return path to the driver. It is inversely related to the distance and is distribute along the path.

Controlled vs Uncontrolled: Unless the traces are carefully designed so that the impedance matches through the line, and thus acts like a transmission line, they are uncontrolled. The need for impedance matching is a factor if for example a bus needs termination to prevent reflections (thus false triggering, etc.). Factors that must be controlled:
Trace Width/Thickness
Spacing between trace/return path. Planes make control over this easier.
Dielectric of insulating material.

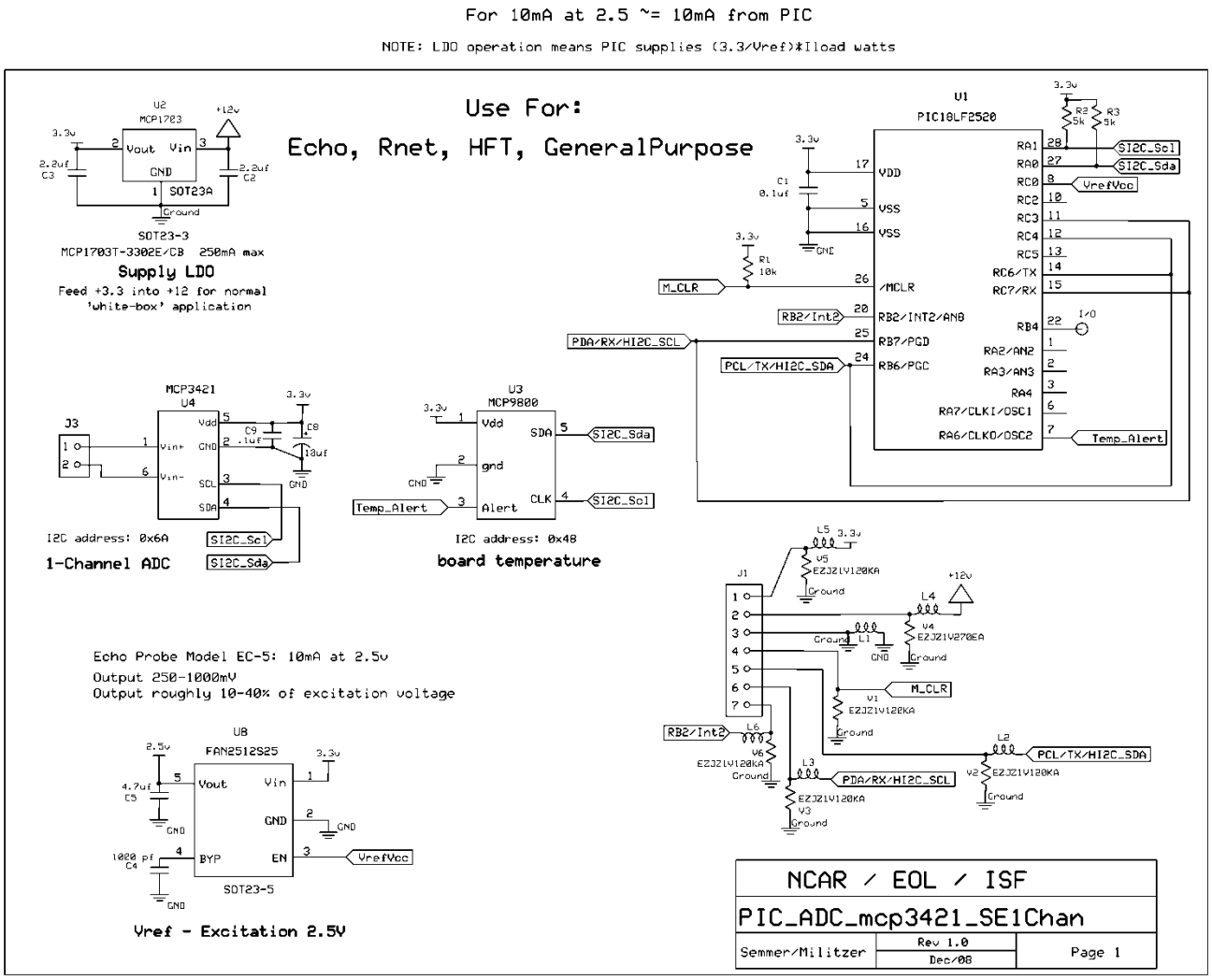
For a controlled line, trace dimensions in themselves aren't important, they can change, but the electric field surrounding the trace needs to be constant. Simple impedance formulas aren't really adequate, instead on-line calculators are best:
High-End design/layout tools show trace impedance along the path.
Simulation Tools: Mentor's HyperLynx,
Calculators: Polar Instruments Si8000 calculator, and also see UltraCAD

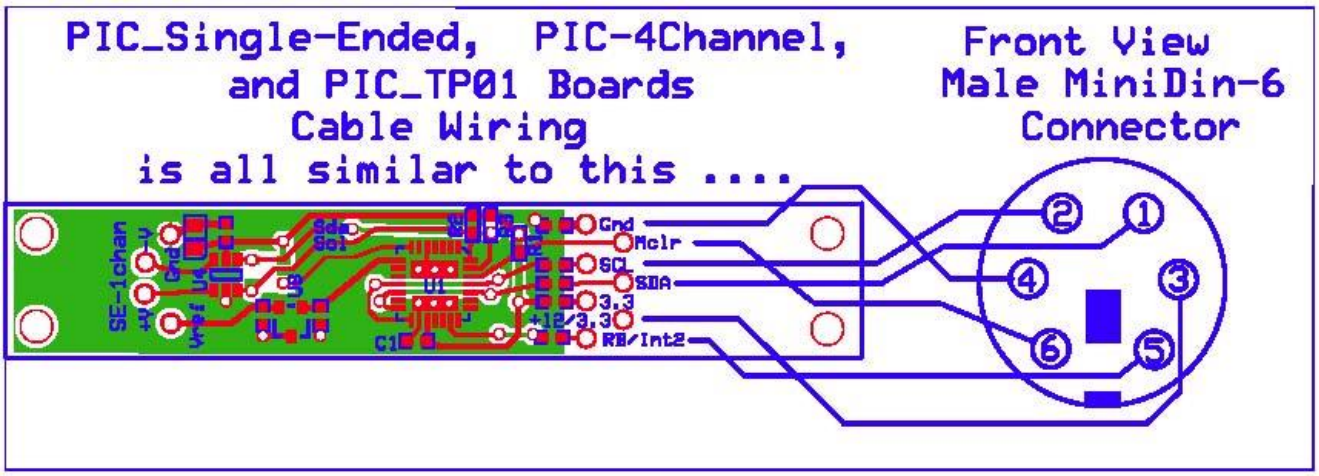
For the Wisard Design, PCB trace issues are considered neglibible, however they can come into play with the I2C buss communications. Normally we're running slow, 100mb, but if we go to 400, it needs to be tested. Also, use of the I2C circuit switch/mux. is desirable. We should look at the transition between the board, connector, and wires to our sensor. Bottom line is: It Has Worked for the low speed stuff.

Schematics / Layouts:

PIC-MCP3421-SingleChannel: Echo, Rnet, HFT

This is the ORIGINAL Version. The newer, Spring 2011 board, removed ferrites and some other changes.





Pinout for Connector to sensors (Rnet, HFT, General Purpose)
Note: Wiring from sensor-board is direct

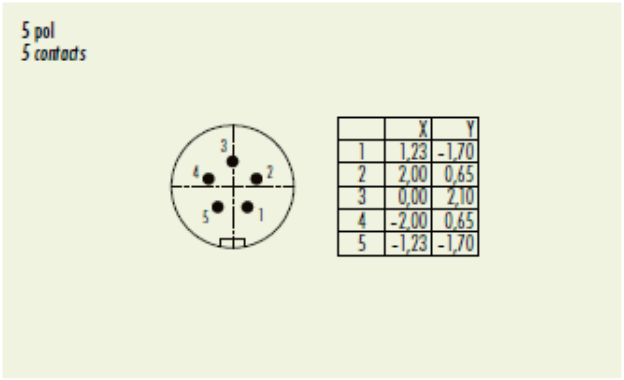
SE 1 Channel boards

Pin 1 V+
Pin 2 V-
Pin 3 no connection
Pin 4 Vref
Pin 5 Gnd

For new ECHO probe:

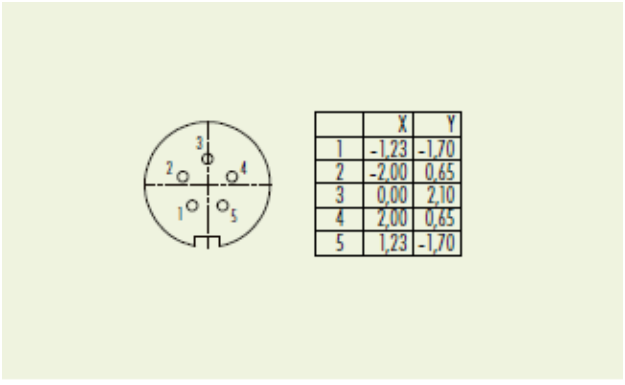
Binder Pin 1 (V+) ------- echo red wire
Pin 2 (V-) -------- echo bare wire/shield
Pin 5 (gnd) ------- echo bare wire/shield
Pin 3 no connection
Pin 4 (Vref) ------ echo white wire

Front View – Male Connector



	X	Y
1	1,23	-1,70
2	2,00	0,65
3	0,00	2,10
4	-2,00	0,65
5	-1,23	-1,70

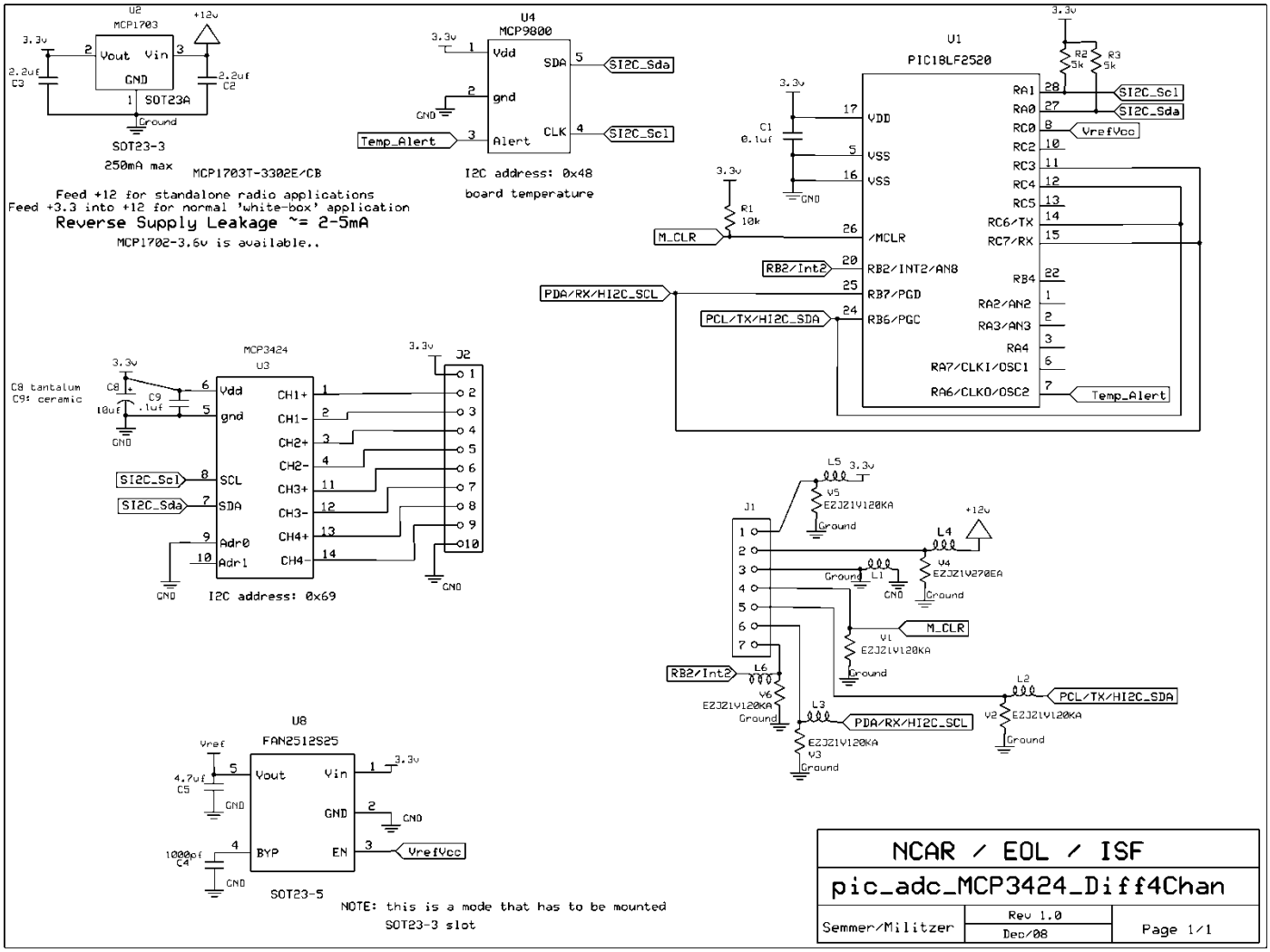
Front View - Female Connector

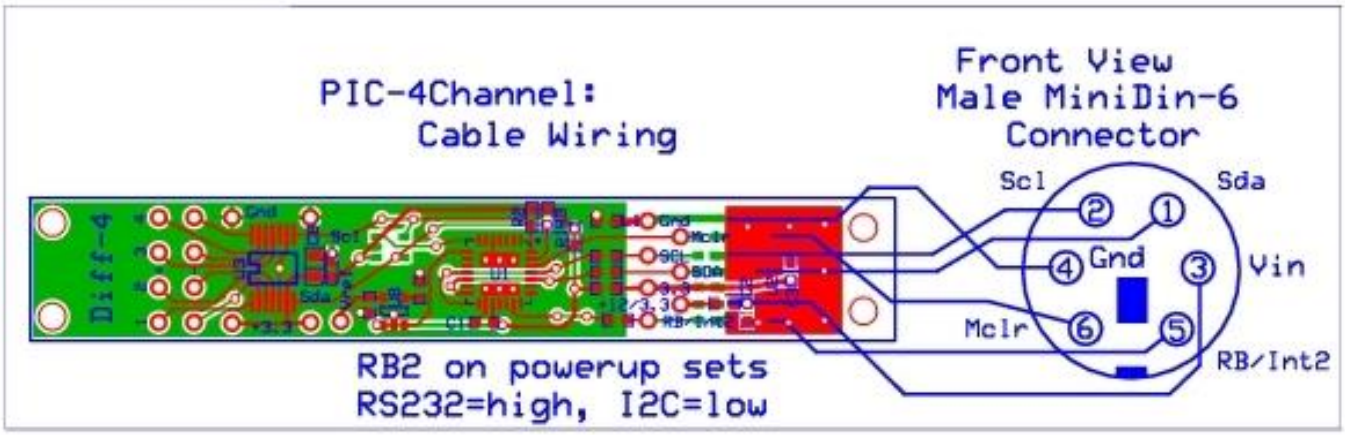


	X	Y
1	-1,23	-1,70
2	-2,00	0,65
3	0,00	2,10
4	2,00	0,65
5	1,23	-1,70

PIC-MCP3424-Four Channel Board

This is the ORIGINAL Version. The newer, Spring 2011 board, includes a RS232 option

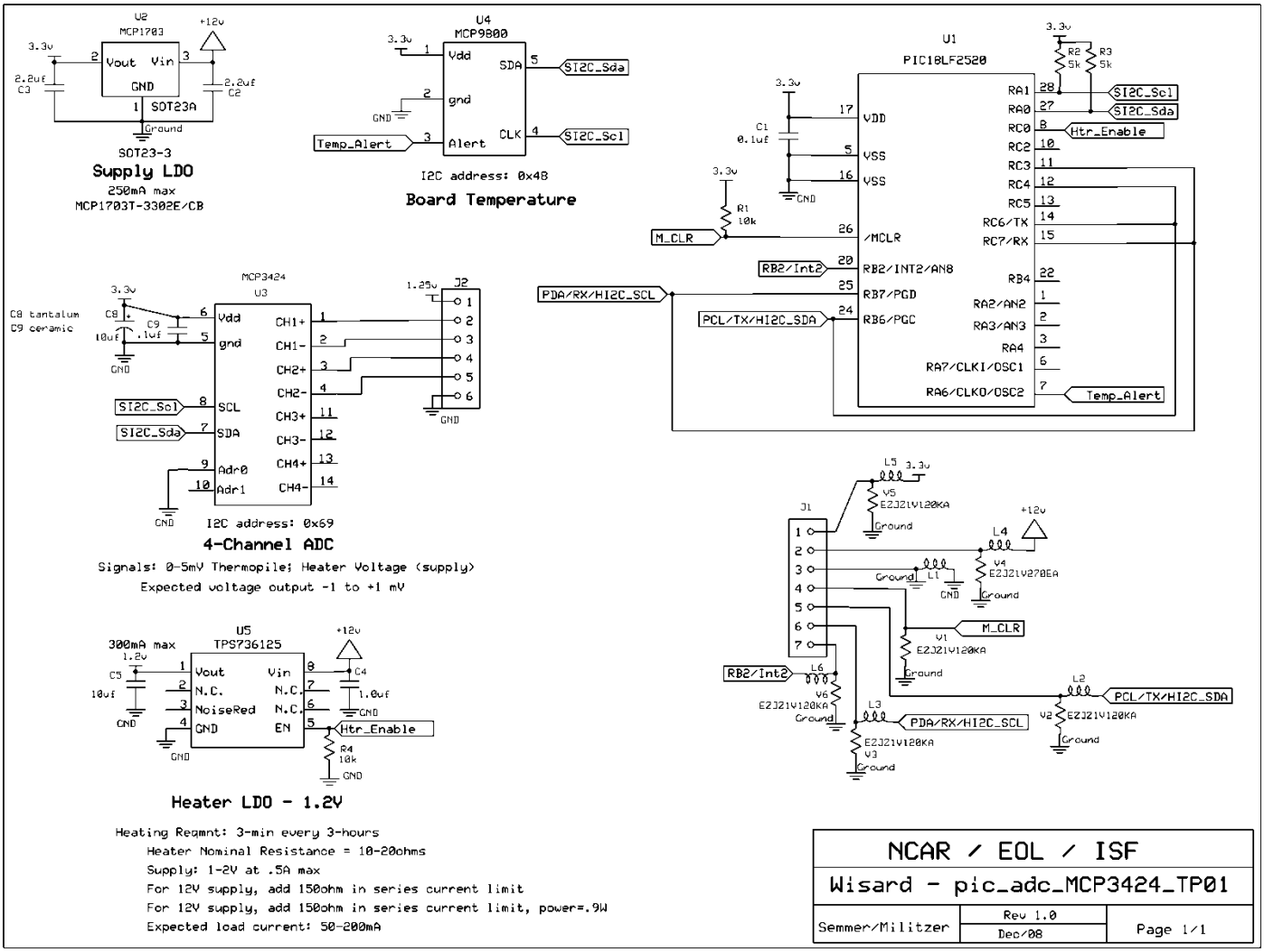


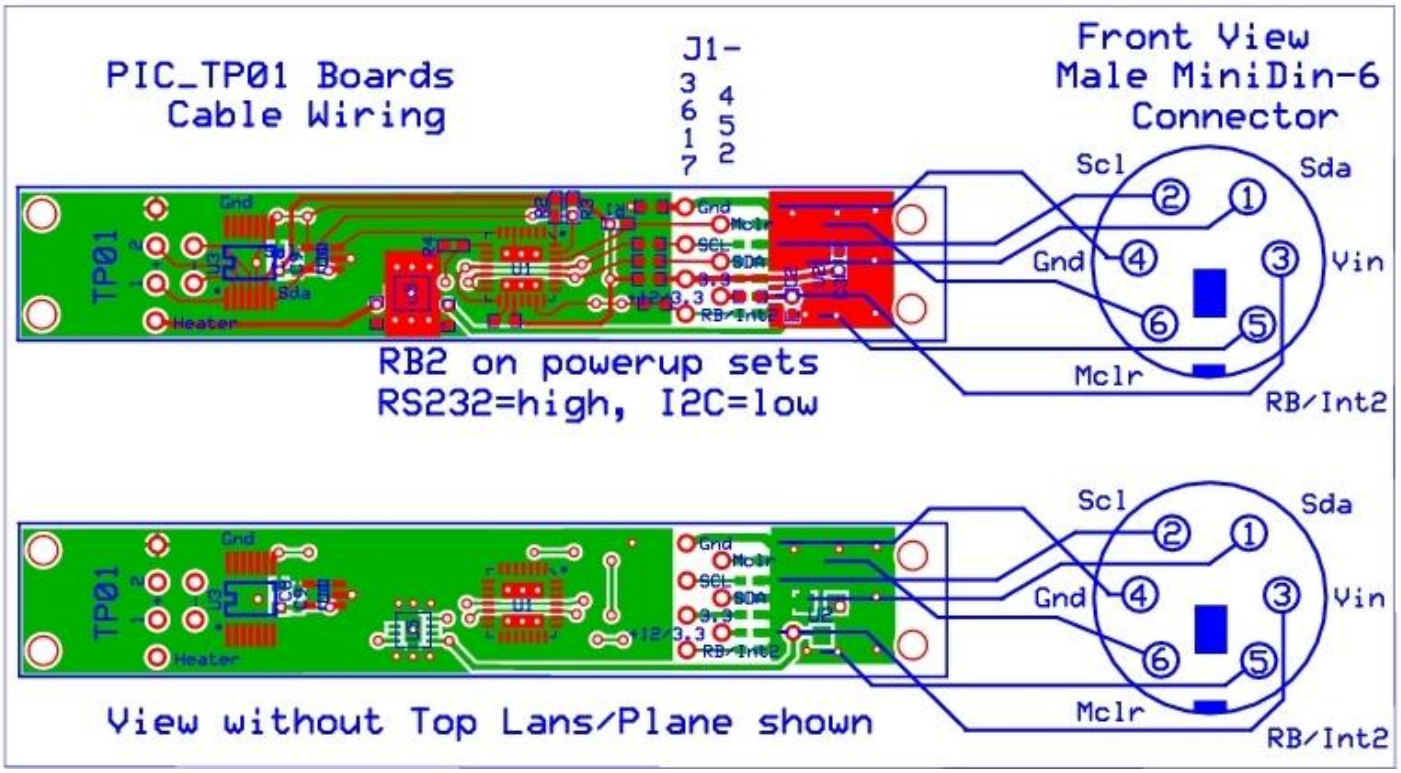


TPO1

Note: *There was a schematic and layout problem for the 2nd run of these boards.* The 'NR' or noise reduction pin of the fixed version Vheater LDO, TPS736125, was connected directly to Vout. That caused the Vheater to rise to 4.9volts when the supply was sufficient which is beyond the tpo1 specification. The boards were fixed and the schematic and layout below were adjusted appropriately.

CURRENT Version / 2011





Pinout for Binder Connector to TP01 soil probe (From probe to the sensor board with din cable.

Binder connector	tp01 board (male 8 pin)
Pin 1 (wht)	channel 1 +
2 (red)	ch 2 +
3 (Blu)	gnd
4 (yel)	ch 2 -
5 (grn)	ch 1 –
6 (brn)	heater

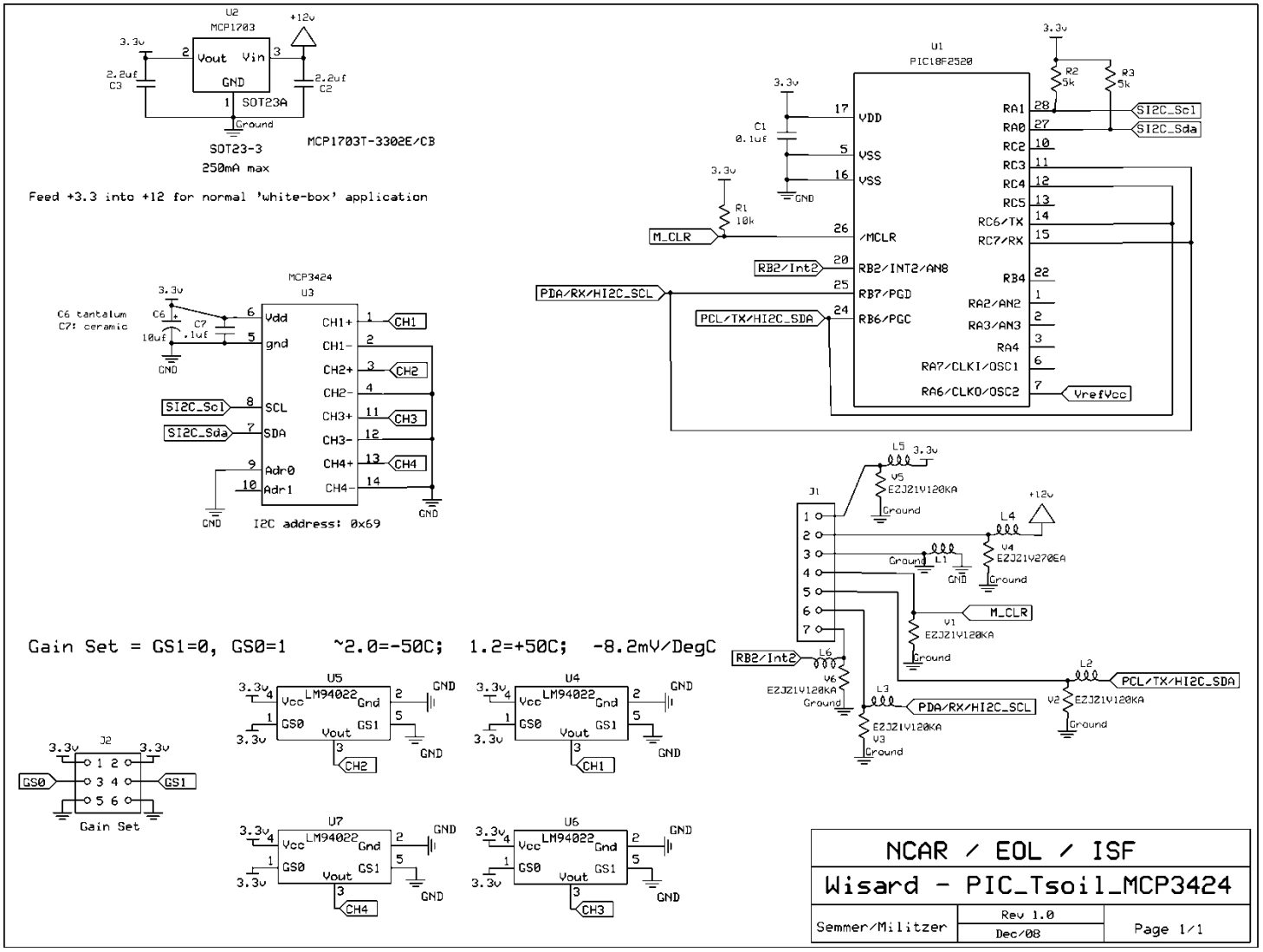
8 pol
8 contacts

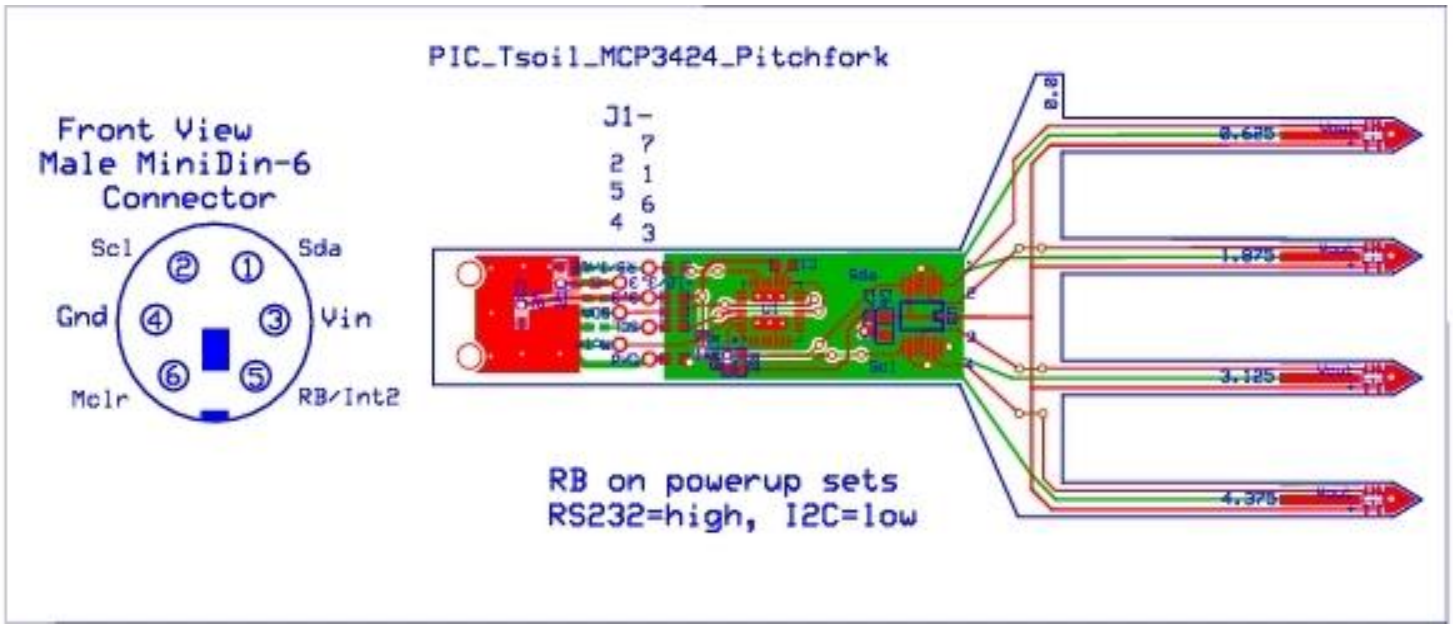
	X	Y
1	0,96	-1,98
2	2,15	-0,49
3	1,72	1,37
4	0,00	2,20
5	-1,72	1,37
6	-2,15	-0,49
7	-0,96	-1,98
8	0,00	0,00

Front View – Male Connector

PIC-Tsoil

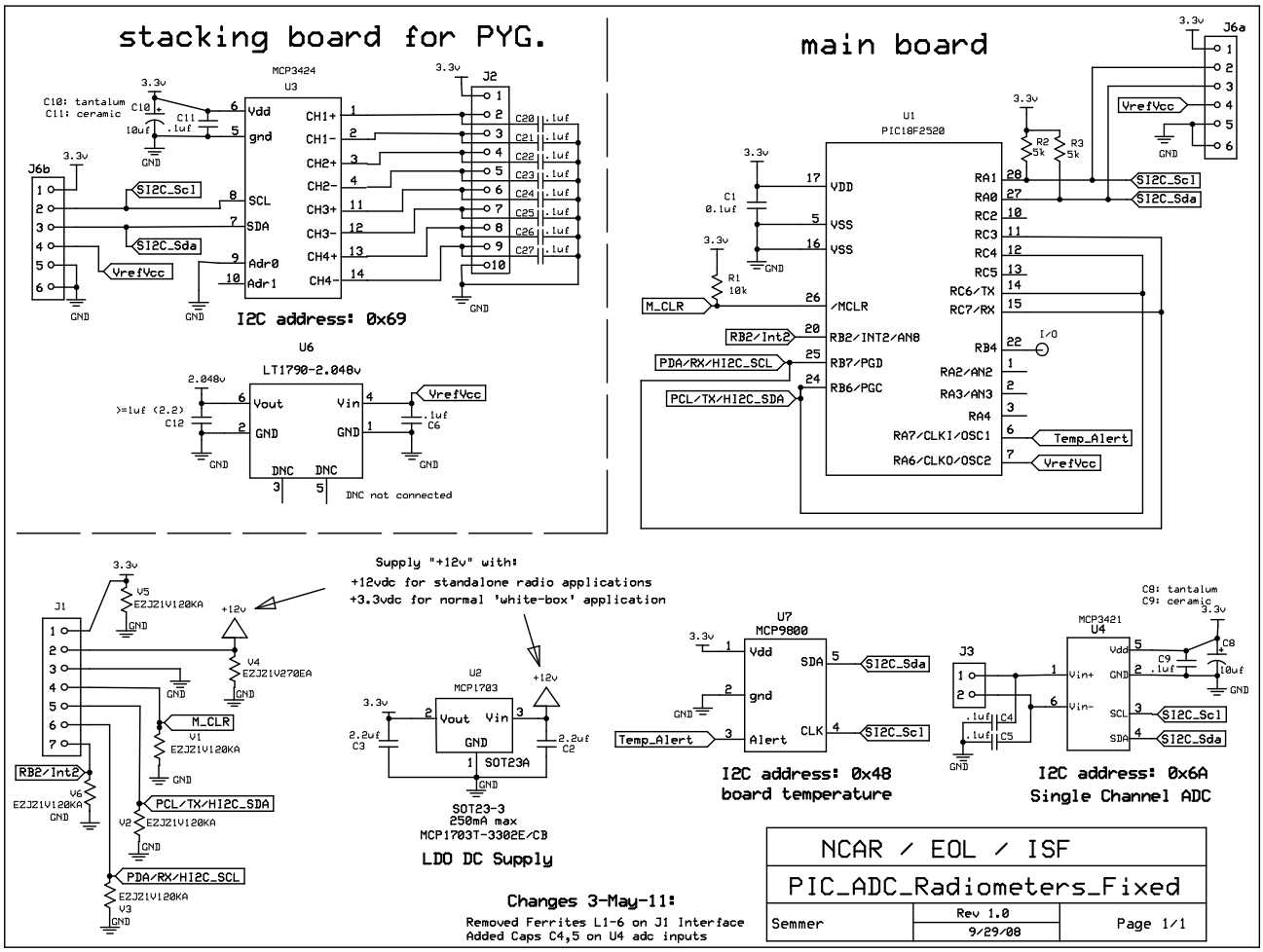
This is the ORIGINAL Version. The newer, Spring 2011 board fixed a problem with the temp sensor layout and removed ferrites, etc. NOTE: in I2C mode, polling must be 2-sec or slower for sampling to finish. Sampling is at ~3.75/second and the averaging process needs time to complete for a result to be successfully delivered.

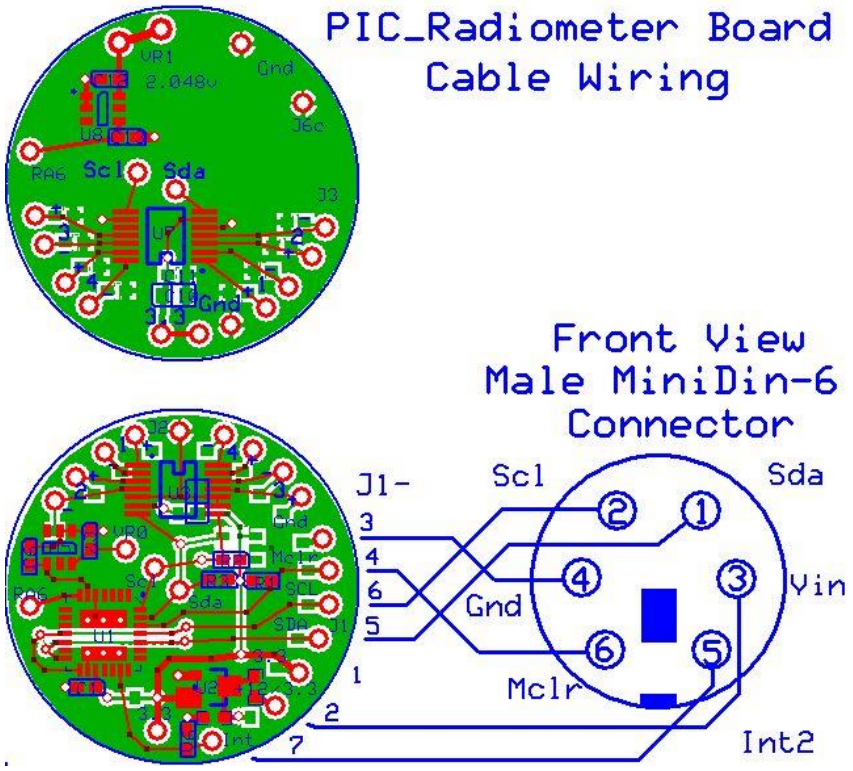



PIC_Tsoil_MCP3424_Pitchfork
Front View
Male MiniDin-6
Connector
Scl
Sda
Gnd
Vin
Mclr
RB/Int2
J1-
7
2 1
5 6
4 3
8.8
8.625
1.875
3.125
4.375
RB on powerup sets
RS232=high, I2C=low

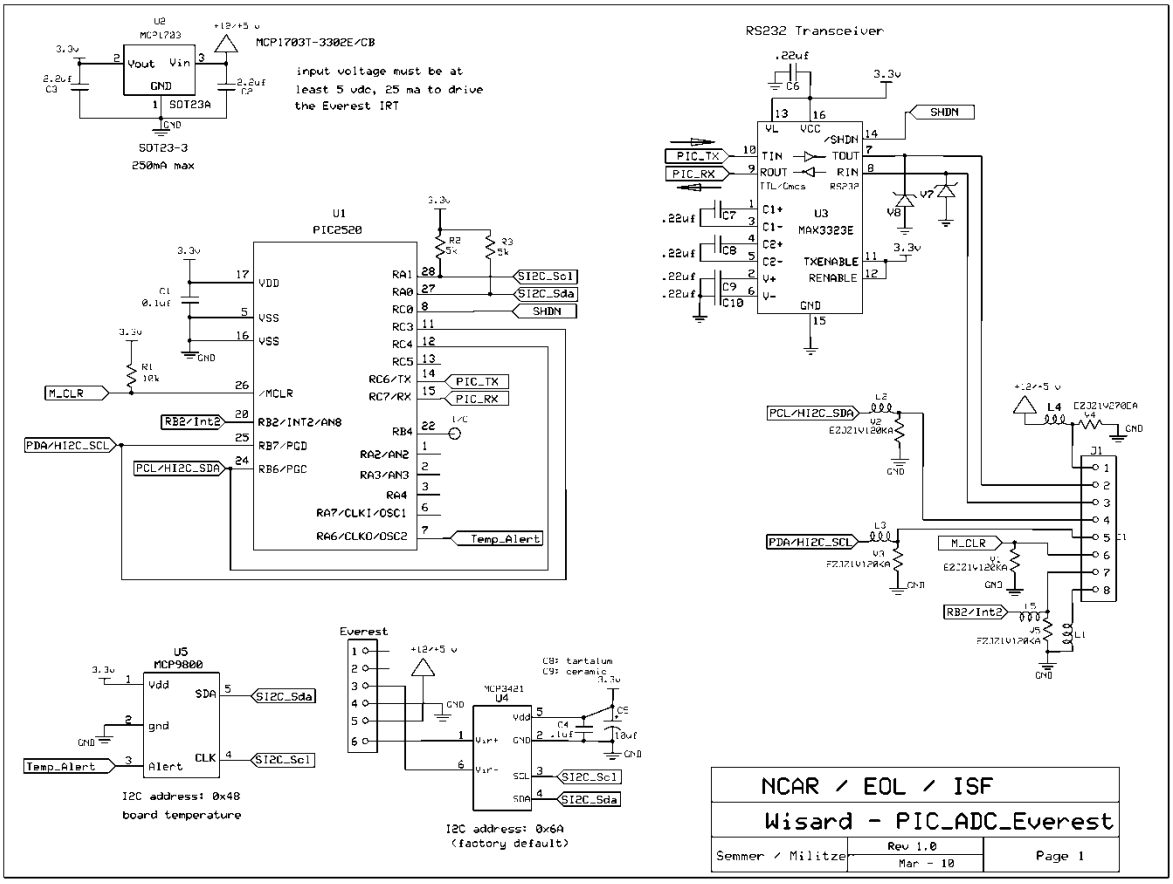
PIC Radiometer Boards

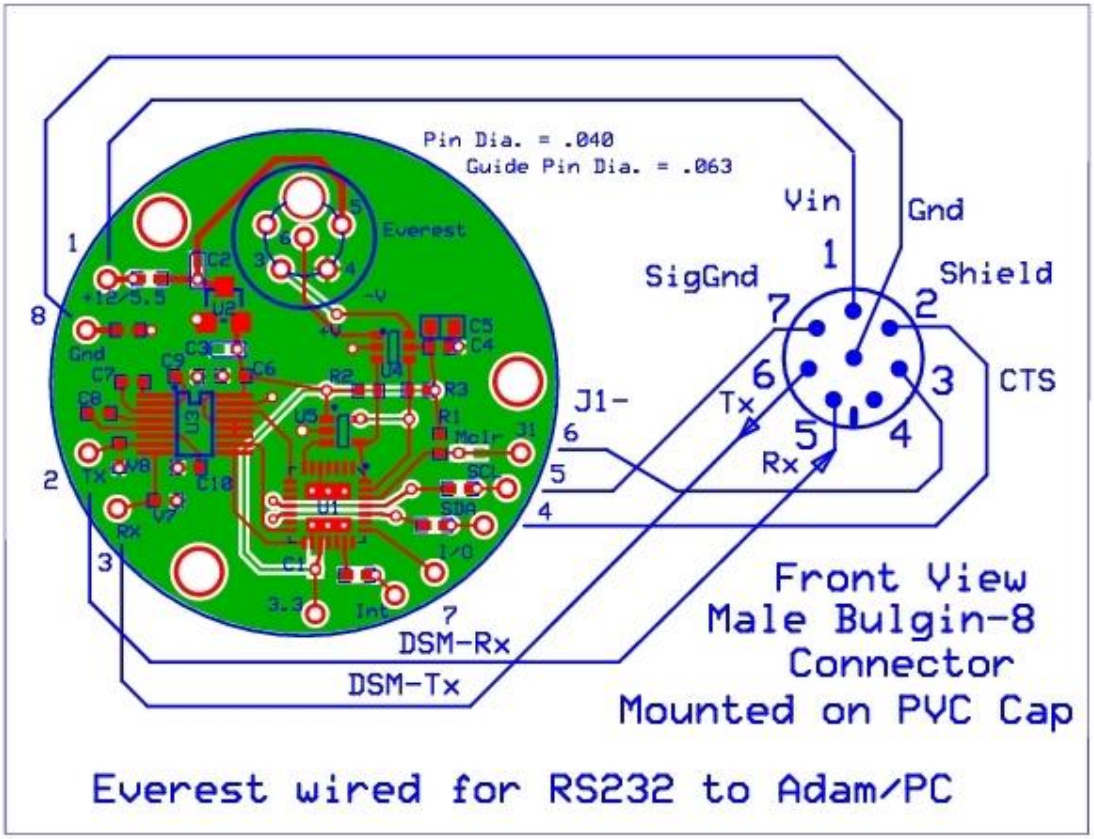
This is the newer, Spring 2011 board, with ferrites removed and some other changes.





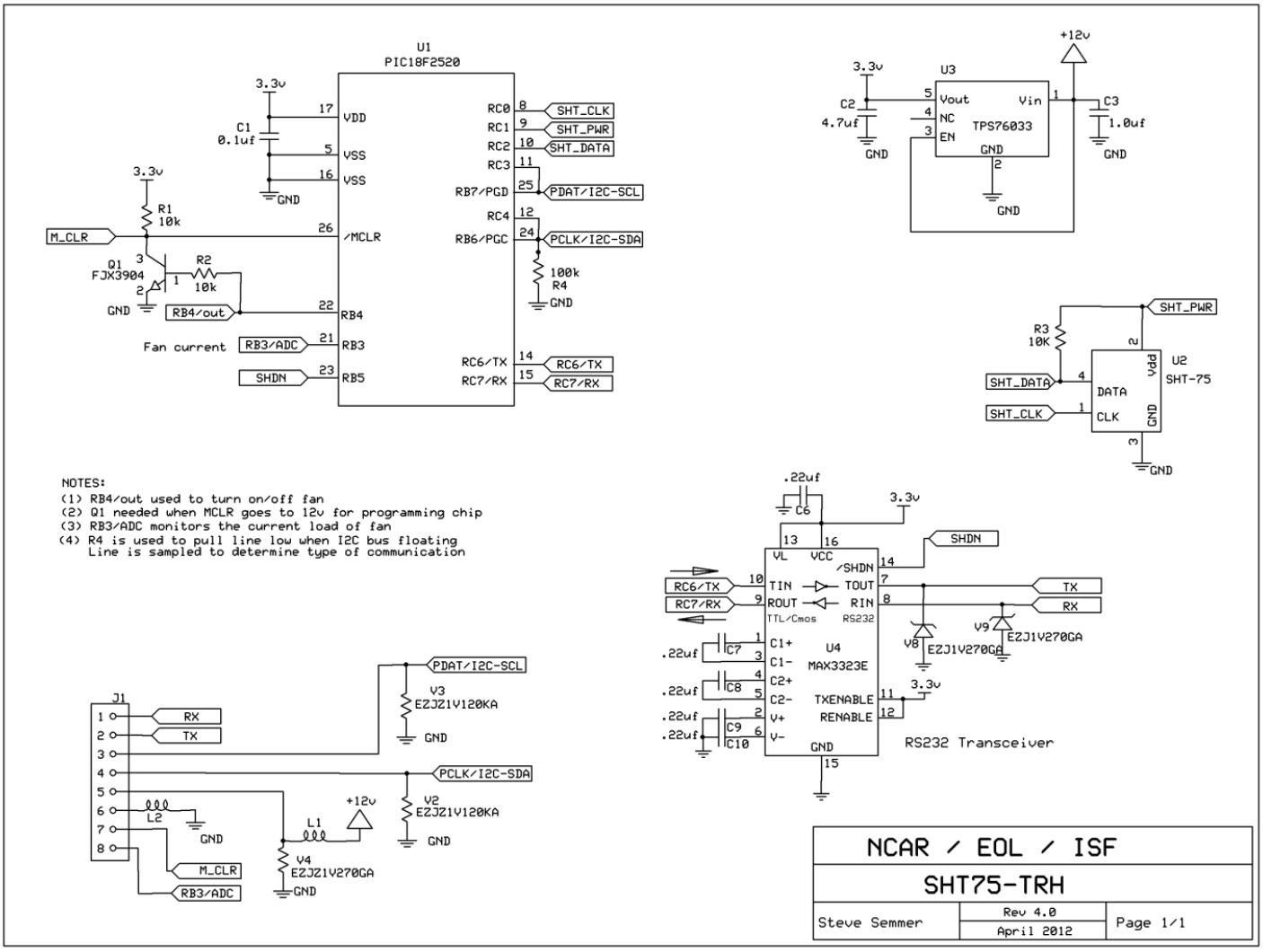
PIC Everest IRT

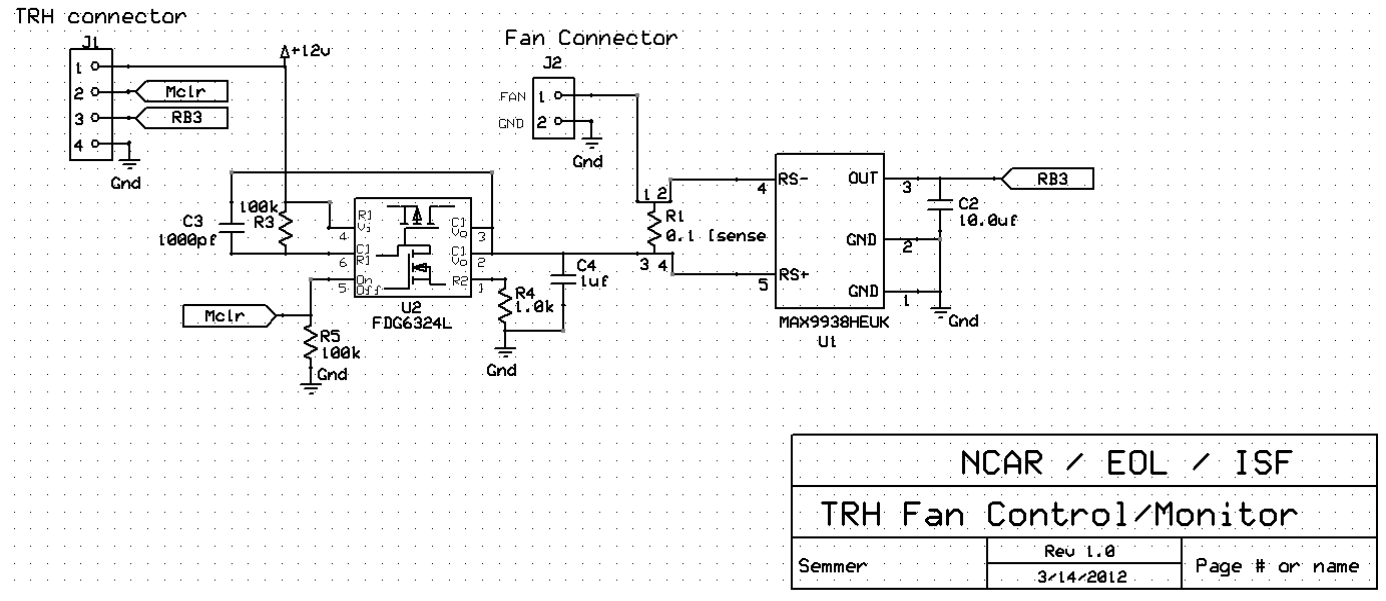


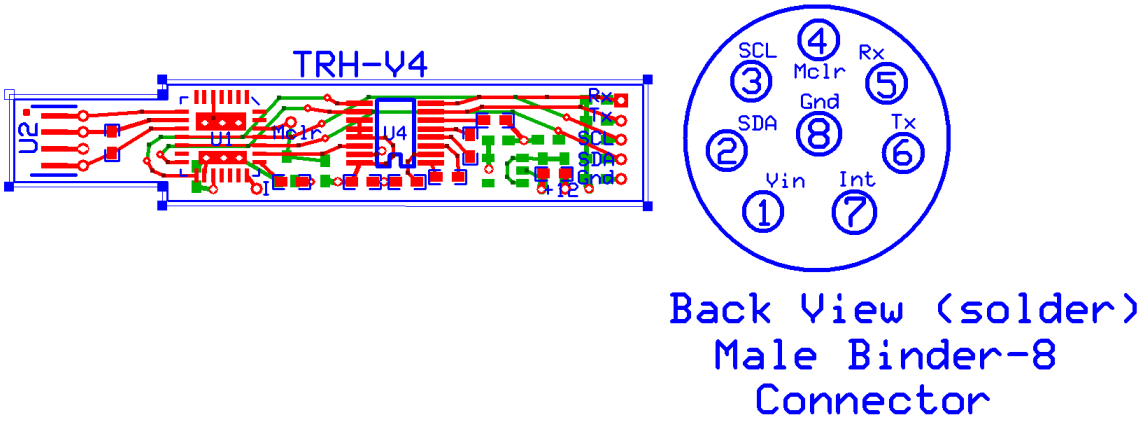


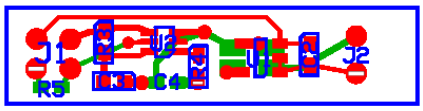
NCAR SHT75-TRH

The **NEW (V4 June2012) TRH** circuit is based on a PIC18F26K20, QFN-28 same footprint as 2520 but is on a 4layer board. It includes the MAX3323 chip and has a separate fan-monitor board to measure the fan current. If the fan current exceeds a preset mA limit, the fan is turned off.

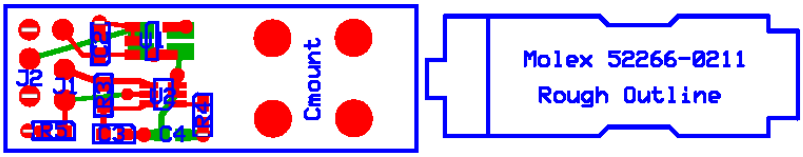








FAN Monitor



Both versions of the TRH can be run in either I2C, TTL (software uart) or RS232 mode.

TRH Message: **TRH44 24.04 7.79 31 0 1595 29 97**

Fields: ID#=44 T=24.04DegC RH=7.79% FanCurrent=31mA FanPwrStatus RawTemp RawRH RawIfan

Sample ISFS-DSM XML entry to decode data:

```
<serialSensor ID="NCAR_SHT" class="isff.NCAR_TRH"
   baud="9600" parity="none" databits="8" stopbits="1" init_string="\x12">
   <!-- TRH004 23.35 26.76 6301 820\r\n -->
   <sample id="1" scanfFormat="TRH%*d%f%f%f%*f%f%f">
      <variable name="T" units="degC" longname="Air Temperature, NCAR
      hygrothermometer" plotrange="$T_RANGE">
         <linear>
            <calfile  path="$ISFF/projects/$PROJECT/ISFF/cal_files/$QC_DIR"
               file="T_${HEIGHT_}_${SITE}.dat"/>
         </linear>
      </variable>
      <variable name="RH" units="%" longname="Relative Humidity, NCAR
      hygrothermometer" plotrange="$RH_RANGE">
         <linear>
            <calfile
             path="$ISFF/projects/$PROJECT/ISFF/cal_files/$QC_DIR"
               file="RH_${HEIGHT_}_${SITE}.dat"/>
         </linear>
      </variable>
      <variable name="Ifan" units="mA" minValue="20" maxValue="70"
         longname="Aspiration fan current, NCAR hygrothermometer"
         plotrange="$IFAN_RANGE"/>
      <variable name="Traw" units=" " longname="SHT T raw counts, NCAR
      hygrothermometer" plotrange="0 10000"/>
      <variable name="RHraw" units=" " longname="SHT RH raw counts, NCAR
      hygrothermometer" plotrange="0 10000"/>
   </sample>
   <message separator="\n" position="end" length="0"/>
</serialSensor>
```

TRH Fans: **Supplier**

Micronel Safety USA
12115 Insurance Way
Hagerstown, MD 21740
301 733-2224

Model: D589L-012GA-2. The breakdown as follows:

D589 = 58mm axial fan, no flange, pressure counterclockwise rotation (flow into the motor end and out the impeller end)
L – motor wires straight off the back of the motor
012 = 12vdc
GA = sleeve bearing, aluminum fan housing
-2 = CLL commutation

The fans should be mounted ‘reverse flow’ sucking air past the TRH into the fan and exhausting above the housing. The wiring is reverse: red=negative black=positive.

TRH Commands:

CTRL-R to software reset the sensor
'Oc' will toggle on/off the calibrated output message
'Or' will toggle on/off the raw data output message
'Ob' will toggle on/off the calibration output data message. This command is reserved for doing calibrations and should not be used in normal operations.
'f' will toggle on/off the fan
CTRL-U to enter EEPROM command mode. The allows changes to be made to specific operational parameters. The user should apply caution when entering this command. Do not make changes unless you really know what you are doing!

EEPROM Commands Menu of the TRH: (Mar2014)

input command format: cmd [value] [[value] [value] ...]

command	desc	type	quantity
MOT	1 sec mote flag	byte	1
FID	Fan ID	byte	1
RES	Bit resolution	byte	1
SID	Sensor ID	byte	1
I2C	I2C address	byte	1
RAT	Data Rate	int	1
CUR	Max Fan current	int	1
TM0	timer0 count	int	1
TA0	T probe a0 Cof	float	1
TA1	T probe a1 Cof	float	1
TA2	T probe a2 Cof	float	1
HA0	H probe a0 Cof	float	1
HA1	H probe a1 Cof	float	1
HA2	H probe a2 Cof	float	1
HA3	H probe a3 Cof	float	1
HA4	H probe a4 Cof	float	1
FA0	Fan current Cof	float	1
EEC	clear EEPROM space		
DEF	DEFAULT flag (reload default values)		
CMD	print command list		
EXT	exit EEPROM handler		

Warning: Upon exit EEPROM data will be downloaded to RAM EEprom:

Example To Change Timer and Fan Current:

Ctrl-u	
TM0<cr>	(to read counter value, 4036 = 1second)
TM0 xx<cr>	(to set a new value)
CUR<cr>	(to read current setting in mA)
CUR xx<cr>	(to set a new value)
EXT<cr>	(to get out of command mode and reboot)

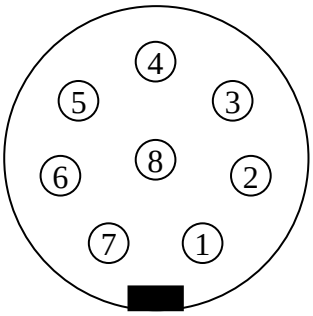
TRH Wiring: Modifications 2012 including Fan Monitor

TRH Signal	TRH binder	color	Housing binder (#wires connected)	FAN monitor	Bulgin	color
12	1	green	1 (3)	J1-1	1 (10")	green
SDA	2	red	2 (2)		2(10")	red
SCL	3	orange	3 (2)		7(10")	orange
Mclr	4	white	4 (1)	J1-2		white
RX	5	blue	5 (2)		6 (10")	brown
TX	6	yellow	6 (2)		5 (10")	black
I	7	purple (DK grn)	7 (1)	J1-3		blue
Gnd	8	black	8 (3)	J1-4	8 (10")	yellow

Binder Connector

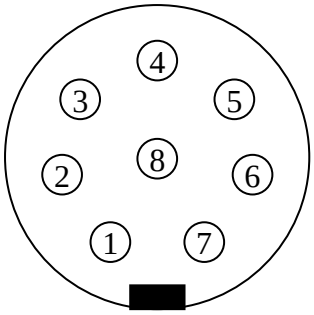
Wagon-Wheel = Female (sockets)
SHT = Male (pins)

Front View – Male Connector
(Rear View – Female)



White Tab

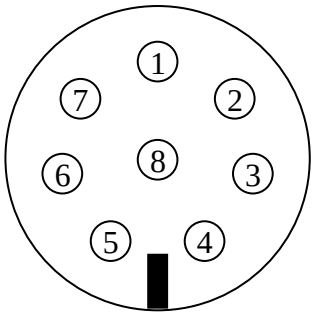
Front View -Female Connector
(Rear View – Male)



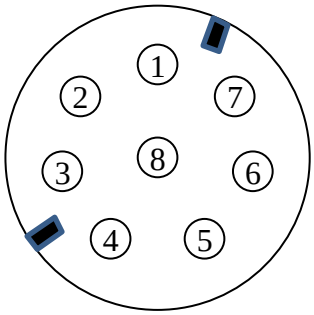
White Tab

Bulgin Connector (Male-Pins)

Front View



Rear View



ParWand – Niwot08

These are the PAR/Accelerometer boards built for Niwot08 deployment to sample the sun penetration in trees.
AS OF …. THEY HAVE NOT BEEN CONVERTED TO WISARD SENSORS…YET…

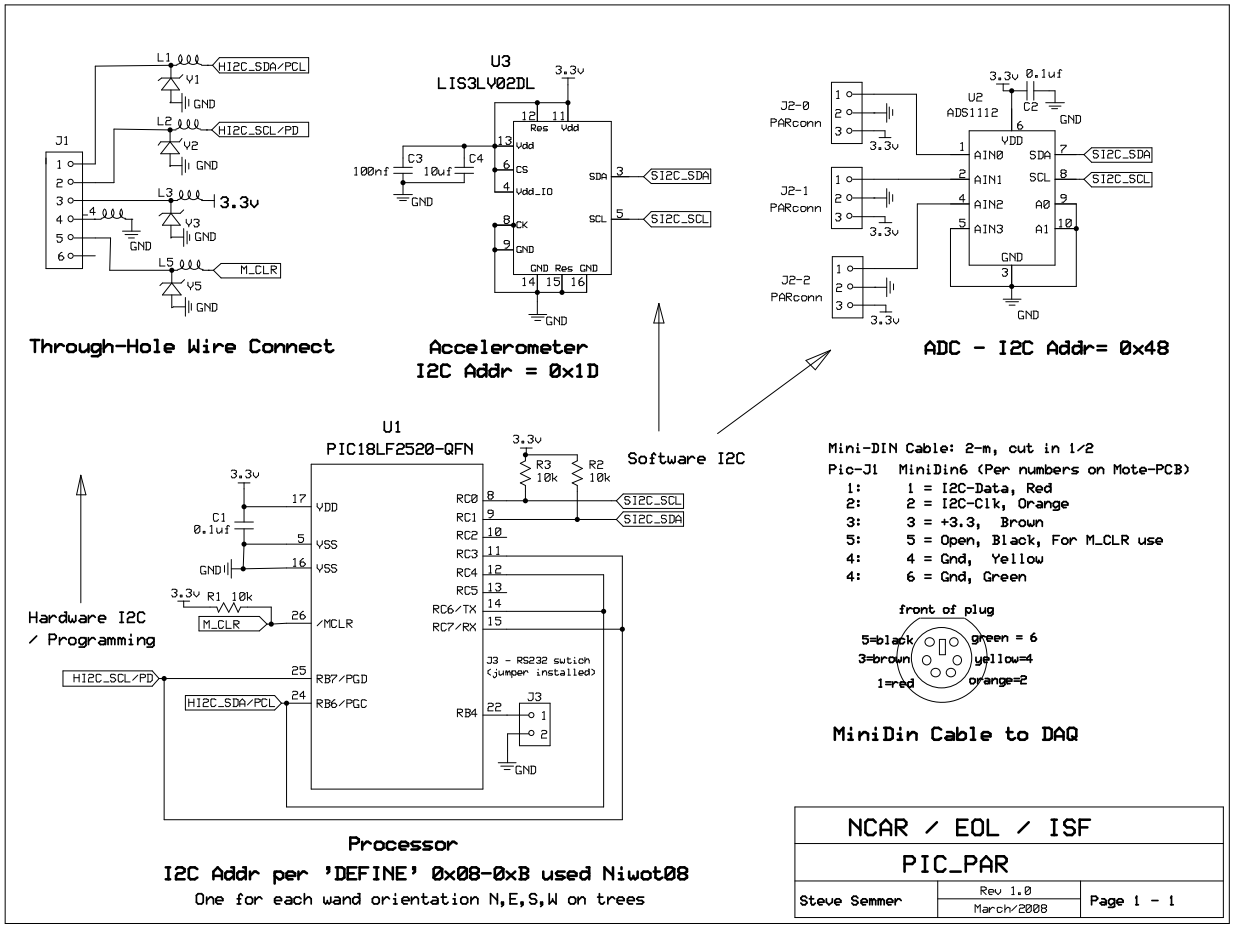
Software_I2C Compiler Issues: A softwareI2C is used on PORTC RC0(scl) RC1(sda) to communicate with the Accelerometer and ADC. Library functions are provided for this, however, there are differences between versions with EVERY STINKING C18/C24 RELEASE MICROCHIP DOES! SteveS nicely documented his code (more later if finding where the heck that is) indicating you must select PORTA or PORTC:

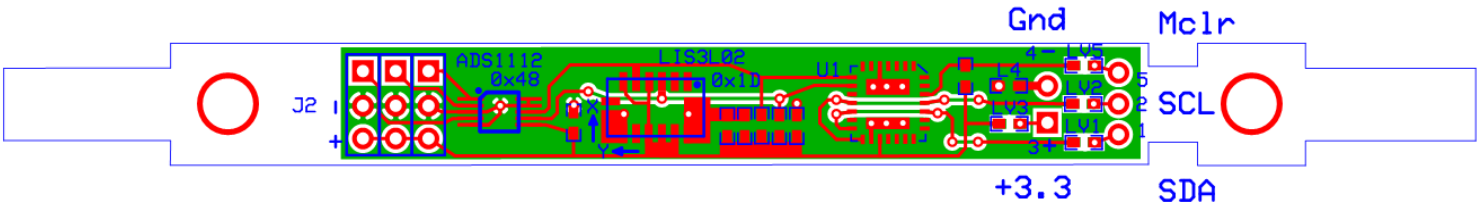
Software I2C control lines can be either PORTC or PORTA depending on the board being used. the default is PORTC.
PORTC0 = i2c clock PORTC1 = i2c data

If you need PORTA lines then you must go into the sw_i2C.h and enable a define statement. (c:\mcc18\h\sw_i2c.h)
PORTA4 = clock PORTA2 = data

But a newer version: mplabc18/v3.43 for example doesn't have C0/C1 under any ex. #if defined(SW_I2C_IO_V1), etc.; so you either need to add it in a local or the library version of sw_i2c.h. Fortunately the functions in the lib .c files are the same as older versions.

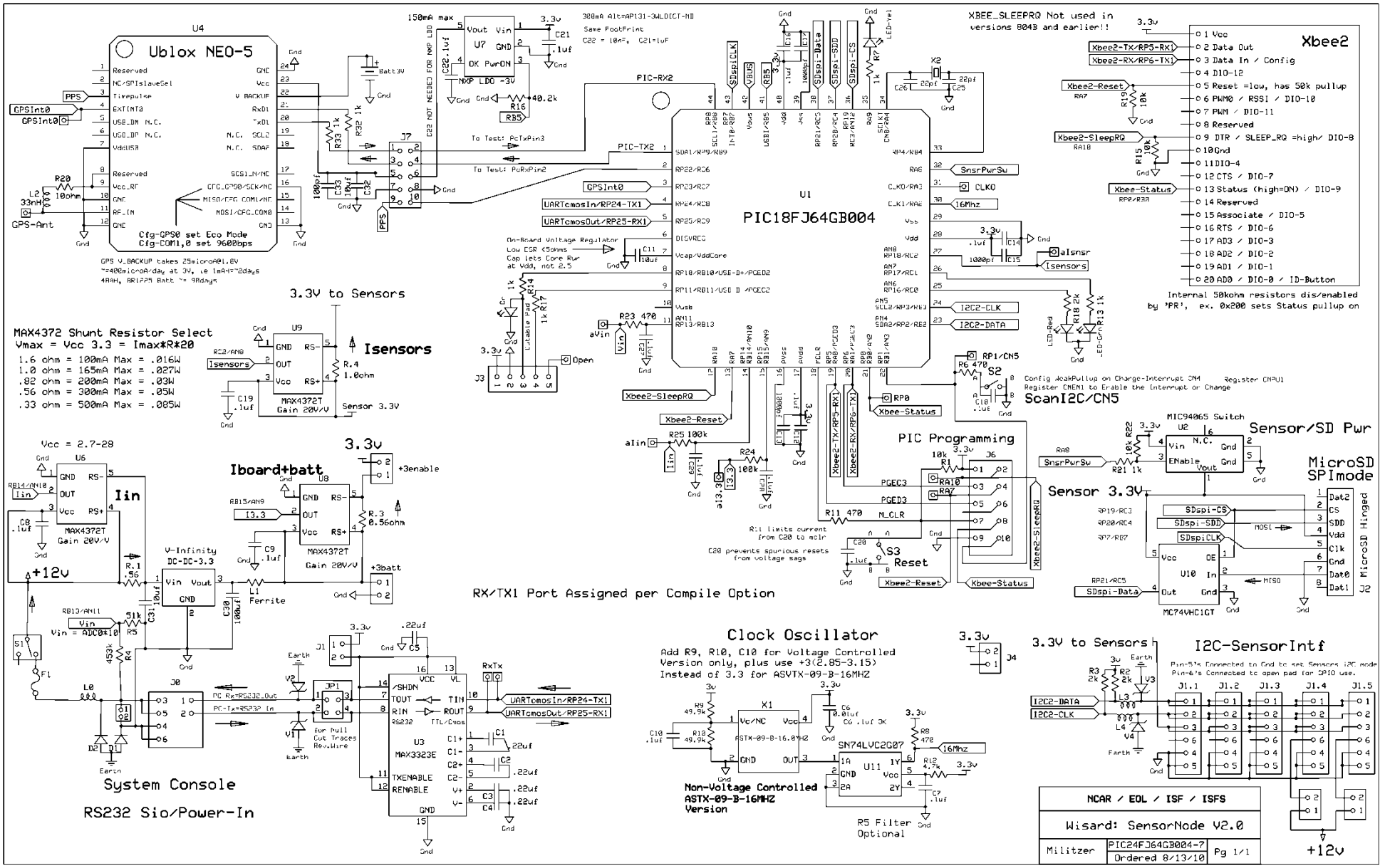
ParWand Schematic / Layout:



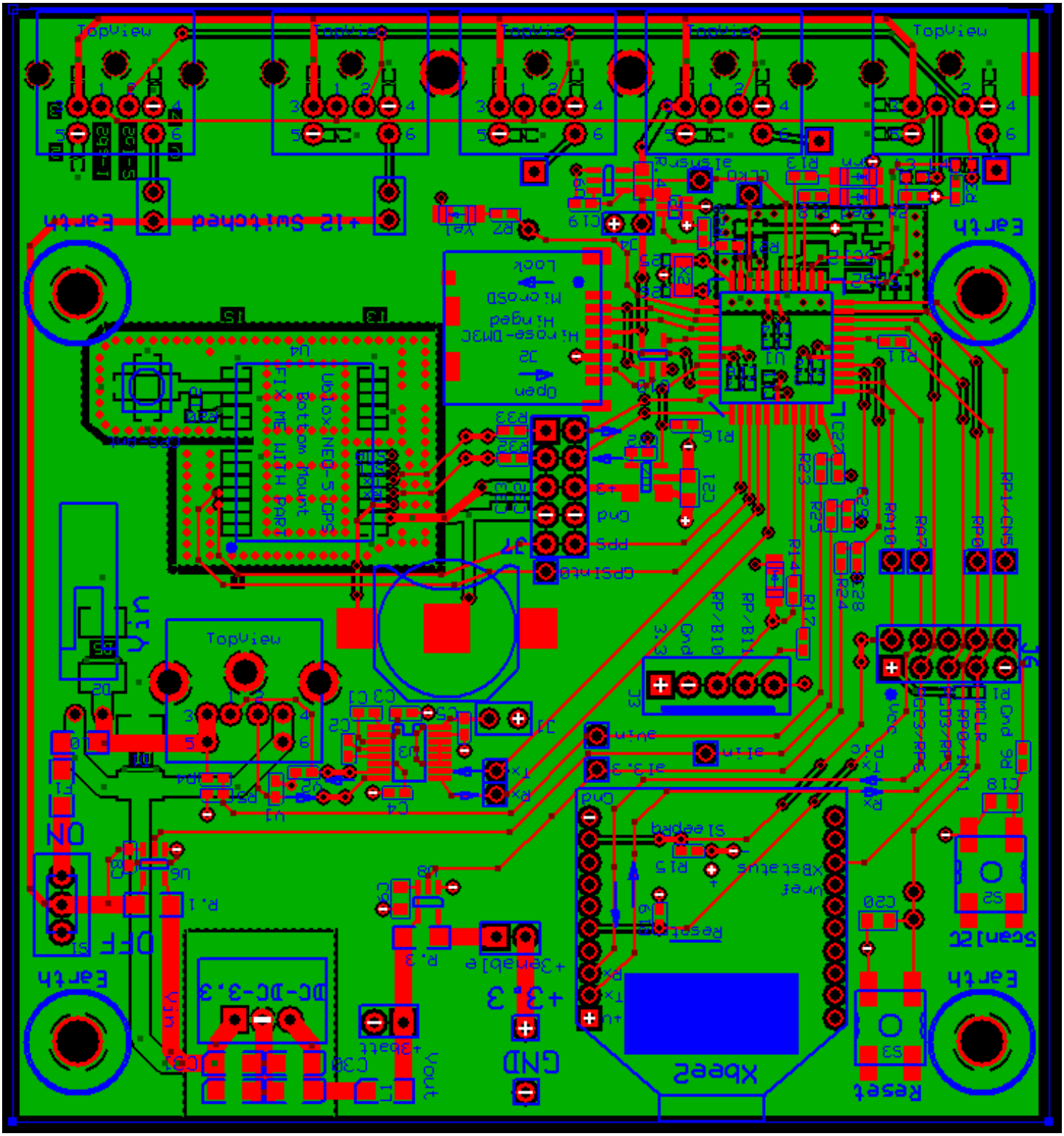


WhiteBox - SensorNode Version2 - Release 13Aug10

The Version2 boards are based on the PIC64GB004 processor and are the only boards/wisard-code that have been deployed in the field for official projects: PCAPS,SCP,SOAS,METCRAXII, etc.

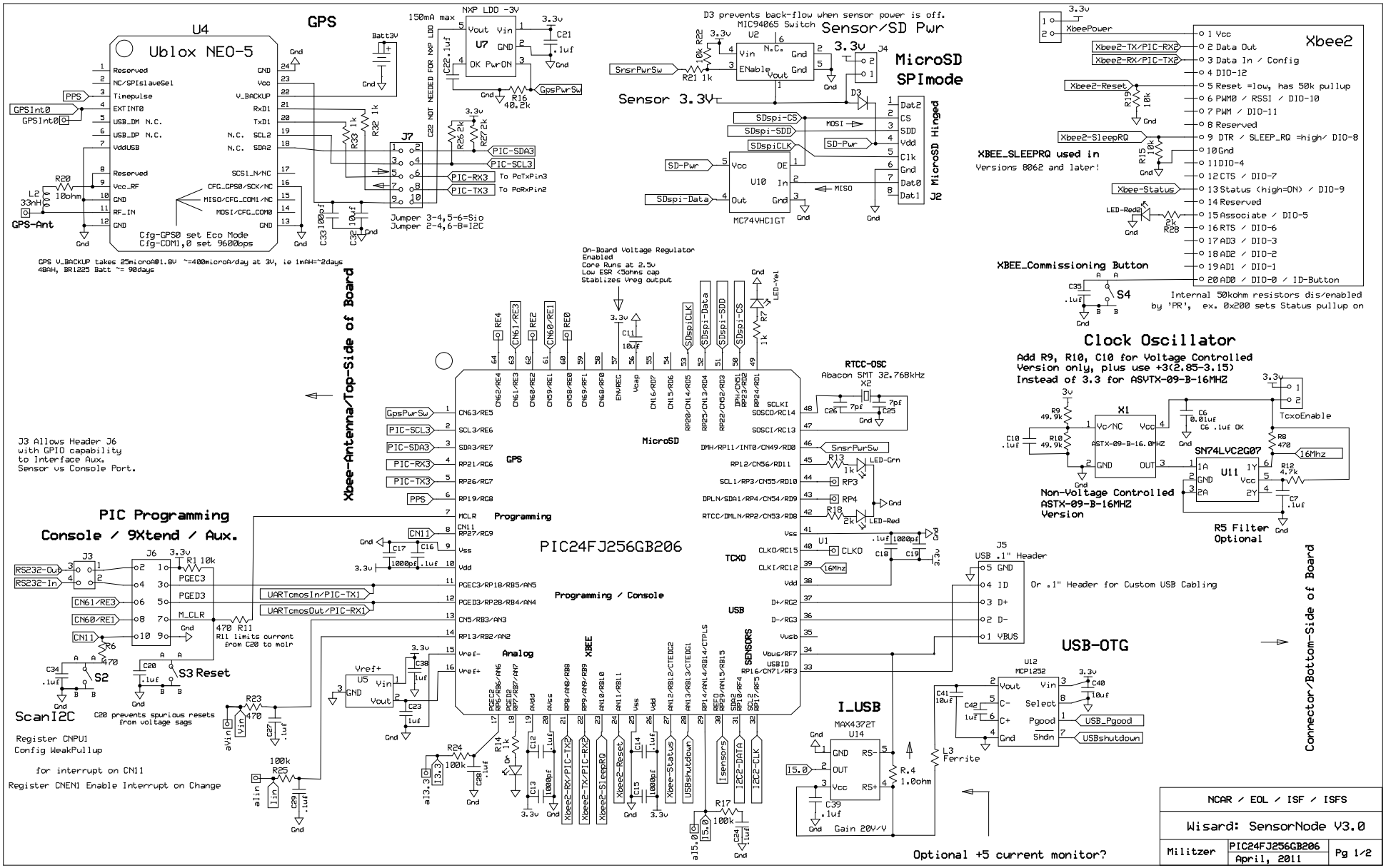


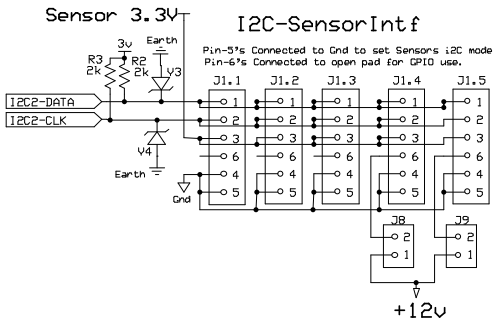
Version 2.0 Release 13Aug10

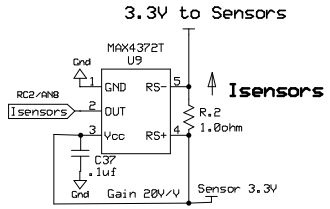


WhiteBox – SensorNode Version 3 Prototype 05Jun11

Because the Version2 board processor didn't have enough program memory for additional functionality and limited data memory for caching, or usb, etc. a prototype 3rd generation was built and wisard software functionality transferred to it (as of end of 2011). It has not been fielded.

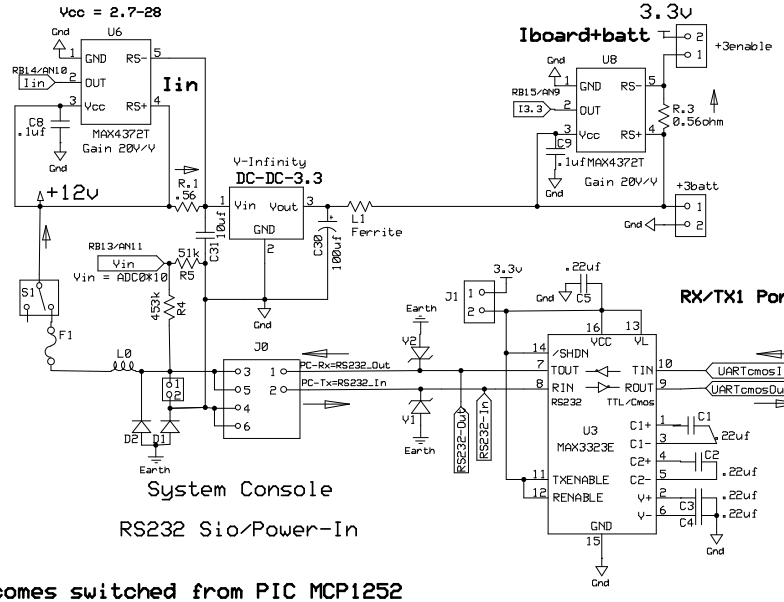






MAX4372 Shunt Resistor Select
Vmax = Vcc 3.3 = Imax*R*20

1.6 ohm = 100mA Max = .016W
1.0 ohm = 165mA Max = .027W
.82 ohm = 200mA Max = .03W
.56 ohm = 300mA Max = .05W
.33 ohm = 500mA Max = .085W



RX/TX1 Port Assigned per Compile Option

USB Power comes switched from PIC MCP1252

'PIC/Xbee/Sensors' Load Monitoring Circuit

Vout = 2.048 for 100mA load current, 1.0ohm R0.3, gain fixed=20V/V
Vout = 2.048 for 200mA load current, 0.5ohm R0.3, gain fixed=20V/V
Vout = 2.048 for 465mA load current, 0.22ohm R0.3, gain fixed=20V/V

Digikey PxxDCT-ND = 0805 1% resistors from .22-9.1ohms
- .22ohm = 465mA --- .0284mA resolution
- .27ohm = 380mA
- .33ohm = 336mA --- .0205mA resolution
- .51ohm = 200mA --- .0122mA /bit.
- 1.0ohm = 100mA --- .0061mA /bit
- 1.5ohm = 67mA
- 2.2ohm = 46mA

Digikey PxxBCT-ND = 0805 5% resistors from .1 - .18 ohms
Digikey PxxPCT-ND = 1206 5% 1/4W resistors from .1 - .18 ohms
Digikey PxxRCT-ND = 1206 1% 1/4W resistors from .22 - 9.1 ohms
MaxCurrent measurable = 2.048/20/R for fullScale of ADS1112

XbeePro: Rx=55mA, Tx=250mA max at 2.8-3.4vdc
Xbee: Rx=50mA, Tx=45mA max at 2.8-3.4vdc

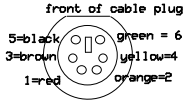
Voltage Requirements by Component
- 2.7-3.3 Antenna
- 2.7-3.6 Ublox NEO5
- 2.2-3.6 PIC24FJ64GB004
- 2.85-3.15 TCXO: VoltageCtrl: ASVTX-09
- 3.15-3.45 TCXO: Fixed: ASTX-09-B
- 1.7-5.5 MIC94065 PwrSw
- .5-6.5 SA57000 LDO for GPS
- 3.0-5.5 MC74VHC Buffer
- 2.8-3.4 Xbee3.0-5.5 MAX3323

Console Cable

Bulgin/232 MiniDin6
- 5: 1 = Rx, White (from telos to pc) (red)
- 6: 2 = Tx, Gray (to telos from pc) (or)
- 1: 3 = +12, Yellow (Brown)
- 1: 5 = +12, Orange (Blk)
- 8: 4 = Gnd, Violet (yellow)
- 8: 6 = Gnd, Red (green)
- 8: Case = Gnd, Shield

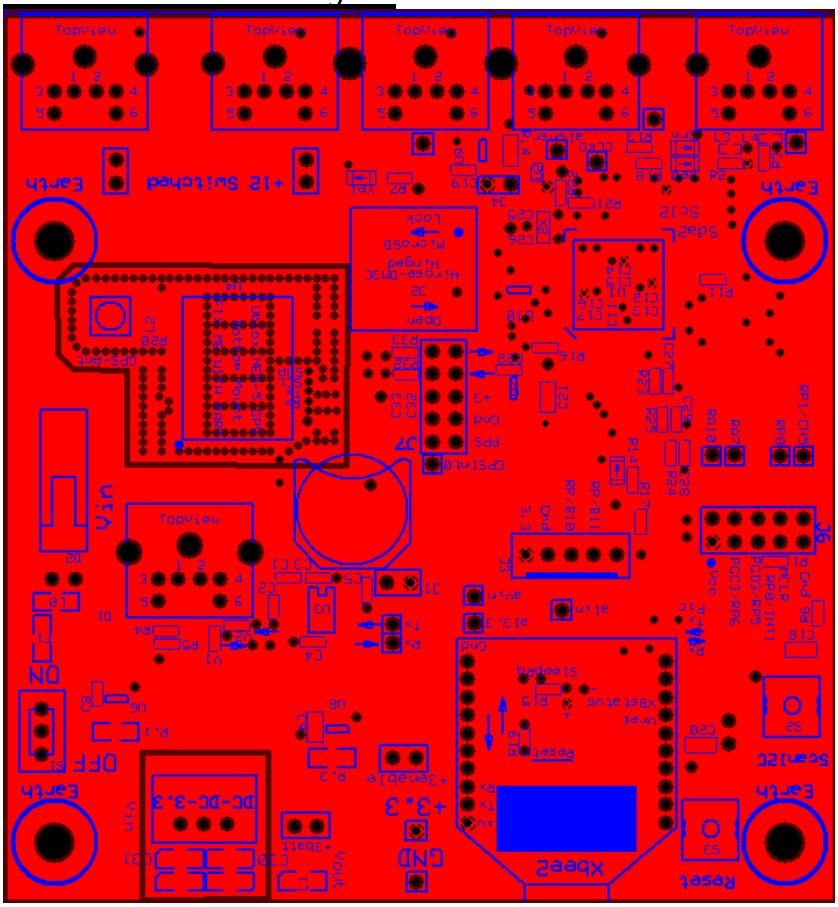
Sensor Cables

Pic-J1 MiniDin6 (Per numbers on PCB)
- 1: 1 = I2C-Data, Red
- 2: 2 = I2C-Clk, Orange
- 3: 3 = +3.3, Brown
- 4: 4 = Gnd, Yellow
- 5: 5 = Available, ThruHole, Black, For M_CLR use
- 4: 6 = Available, ThruHole, Green

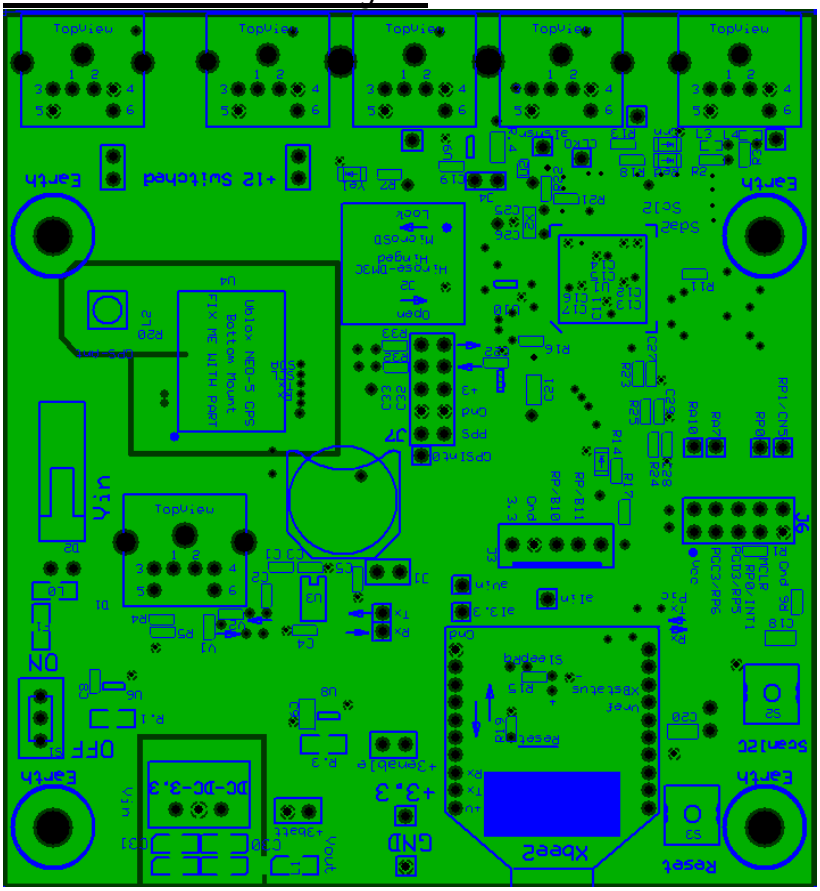


NCAR / EOL / ISF / ISFS		
Wisard: SensorNode V3.0		
Militzer	PIC24FJ256GB206 April, 2011	Pg 2/2

Inner Power Layer



Inner Ground Layer



Version 3.0 Prototype 05Jun11