

PIC'n the Beehive

By

Carl the Beekeeper

Entry submitted for

Microchip's PIC32 Design Challenge

Final Prototype Phase 4, ending March 2009

Executive Summary

Data acquisition is a task that often requires many pieces of equipment, sensors and cables connecting all of it. The data then needs to be sent off to a computer to be processed for analysis. The Microchip PIC32 employed as a data collection, control, analysis and embedded web server device can do all of these tasks while being extremely cost effective. This is perhaps best illustrated by the project presented in this document, [PIC'n the Beehive](#). The basic model of this project can be packaged in a case smaller than a pack of cigarettes. The on board features of the PIC32 lend itself to monitoring many of the functions directly. A few tasks are easily handled by the use of additional Microchip devices connected through the PIC32's integrated I2C and SPI ports. The Ethernet interface is one such use of the SPI port. The future expansion possibilities are endless using these ports for upgraded models of the project.

Installed hardware and software function levels at the phase 4 date include the following:

- Functional embedded PIC32 web server collecting data, analyzing the data, and presenting it in a flexible display of graphs and analytical analysis as requested by the user on a web page

These tasks are achieved by the use of the following items:

- 5 MCP9801 temperature sensors installed and under software control and web graphs
- 1 LDR Daylight sensor installed and under software control and web graphs
- 2 Hall sensors for security installed and under software control and web page access
- 4 PIR motion sensor installed and under software control and web page access
- 1 Humidity sensor installed and under software control and web graphs
- 2 Entrance servos installed and under software control and web page access (8 designed in)
- 2 Bee traffic counters installed and under software control and web page graphs (8 designed in)
- 5 voltage monitors for the system are installed and under software control and web page access, with 1 having an email alert for an alarm condition

Many more sensors are being implemented in the next scaled up model. Some of these are currently in different stages of development. They include the following:

- 1 non-contact IR temperature sensor hardware installed and software tested but not currently implemented in phase 4 software.
- 1 Microphone installed but not under phase 4 software control
- Auxiliary controls for the video cameras are installed and usable but not currently connected to external video.
- The cell phone dialer hardware would use one of the available Aux ports but is not currently programmed to do so.
- UART2 interface is installed and is currently being used as a diagnostic tool during development and will be reconfigured to accept ambient environment data.
- The GPS interface hardware is installed but not under software control

Not currently installed or under software control are the bee and hive weight scales and the inductively coupled brood frames. These are at various levels of design and testing currently, as are additional sensors to be added on.

The basic functions of using the PIC32 as a system controller and web server to collect, analyze and present data has been met with less than 60% of the PIC32 memory used, and not very tight code at that. Clearly there is enough memory available for even more complex advanced features than what is currently in the works above.

I am in the process of setting up a web site CarlTheBeekeeper.com to post future updates after the contest ends on this project.

Table of Contents

Executive Summary.....	2
Table of Figures.....	5
Project Concept for PIC’n The Beehive	7
Project Design Overview	7
Functional areas monitored by the PIC32.....	7
Beehive environmental conditions that can be controlled	7
Other possible monitored functions.....	7
Project Design Approach.....	8
Project Block Overview	10
Hardware Block Diagram	11
Carl’s Bee Management System (BMS) Reference Guide.....	12
Project Summary.....	12
Design Background	12
User Interface Summary	12
Implemented Functions Currently Operational.....	13
Features in Development for Enhanced Functions:.....	13
Possible Product Configurations.....	13
Project Overview.....	14
Hardware Design.....	15
PIC32MX360F512 Pin & Function Assignments.....	15
Hardware Overview	20
Schematic Details page 1- System Processor.....	21
Schematic Details page 2- Power Distribution.....	21
Schematic Details page 3- Ethernet & UART Interfaces	21
Schematic Details page 4- Controller Environment Sensors.....	22
Schematic Details page 5- Bee Traffic Counters	22
Schematic Details pages 6, 7 & 8- Bee Scale Modules- advanced feature	22
Schematic Details page 9 & 10- Status & Control IO’s & Interfaces	23
Schematic Details page 11- SPI Memory	24
Schematic Details page 12- PICtail Plus pin mapping to PIC32 functions.....	24
Schematic Details page 13- Bee Temperature Subsystems.....	24
Schematic Details page 14 & 15- J10 & J11 net mapping to PIC32 functions.....	24
Schematic Details page 16- Prototype area and Servo Power Connections.....	24

Schematics	25
Software Design	45
Software Block Diagram.....	45
Software Design – Embedded web server user interface.....	46
Index Web Page	46
Status Web Page	48
Parametric Data Web Page	49
Auxiliary Controls Web Page.....	50
Hive Configuration- Security Web Page- Sub Menu	51
Hive Configuration- Environmental Web Page – Sub Menu	52
Hive Configuration- Bee Traffic Web Page– Sub Menu	53
Hive Configuration- Bee Scale Web Page– Sub Menu	54
Hive Configuration- Controller Web Page– Sub Menu	55
Utilities Web Page.....	56
Email Data Log Output	58
PIC32 System Controller Software Overview.....	59
Software Modules.....	59
MainDemo.c.....	59
Web Server code.....	61
Prototype Hardware Photos	62
Main Board.....	62
Bee Tunnels.....	63
Closing Statement.....	63
Bio- Carl the Beekeeper	64

Table of Figures

Figure 1 Project Block Overview	10
Figure 2 Inductively coupled brood frame temperature monitoring	10
Figure 3 Hardware Block Diagram.....	11
Figure 4 PIC32MX360F512 Pin & Function Assignments	19
Figure 5 Populated prototype project board.....	20
Figure 6 60-pin connectors allow stacking of the boards	20
Figure 7 Schematic page 1	26
Figure 8 Schematic page 2	27
Figure 9 Schematic page 3	28
Figure 10 Schematic page 4	29
Figure 11 Schematic page 5	30
Figure 12 Schematic page 6	31
Figure 13 Schematic page 7	32
Figure 14 Schematic page 8	33
Figure 15 Schematic page 9	34
Figure 16 Schematic page 10	35
Figure 17 Schematic page 11	36
Figure 18 Schematic page 12	37
Figure 19 Schematic page 13	38
Figure 20 Schematic page 14	39
Figure 21 Schematic page 15	40
Figure 22 Schematic page 16	41
Figure 23 Prototype board assembly drawing.....	42
Figure 24 Prototype board topside view loading.....	43
Figure 25 Prototype board bottom side view loading	44
Figure 26 Software Block Diagram.....	45

Figure 27 Home web page. Top function ‘T-Base Outside’, 2 different days selected. Bottom function ‘T-Top Inside’, same 2 days selected as in top graph. 46

Figure 28 Home web page. Top function ‘Daylight Sensor’, 1 day selected. Bottom function ‘T-Base Outside’, same day selected as in the top graph. Hours for data analysis selected as 8 to 16 in top graph. 47

Figure 29 Hive Status web page..... 48

Figure 30 Hive Parametric Data web page..... 49

Figure 31 Auxiliary Controls web page..... 50

Figure 32 Hive Configuration- Security web page..... 51

Figure 33 Hive Configuration- Environmental Controls web page. 52

Figure 34 Hive Configuration- Bee Traffic Configuration web page. 53

Figure 35 Hive Configuration- Bee Scale Configuration web page. 54

Figure 36 Hive Configuration- Hive Controller Configuration web page. 55

Figure 37 System Utilities web page. 57

Figure 38 Email data log sample output 58

Figure 39 Photo of prototype board and early bee tunnel in foreground. Non-contact IR bee temperature sensor can be observed on the end of a 4-wire cable lying across the bread board. 62

Figure 40 A later version of the bee tunnel is seen here with the non-contact IR temperature sensor mounted on the bee tunnel on the left. Not shown are additional versions with the entrance servos and the bee scales configured on them..... 63

Project Concept for PIC'n The Beehive

This project provides a means to monitor honeybee health parameters in a hive, to allow efficient bee management and research on the quality of the colony.

Bees are important to the food supply on which we all depend. They are the pollinators that result in many trees and vegetables producing fruit. The concern in recent years has been mites, viruses and the treatments for them stressing the bee's health. Lately the focus has been on a phenomenon called **Colony Collapse Disorder** (or **CCD**). Basically the worker bees disappear and then the hive dies out for lack of enough bees to maintain the hive. A Google search of 'Colony Collapse Disorder' will yield a wealth of opinions on the problem, but the verdict is still out as to the bottom line cause. Basic research of any degree depends heavily on having 'control' hives with known variables when doing experiments. This **PIC'n the Beehive** project is one solution to measuring and understanding those variable factors in the experiment. The PIC32's processing power brings the laboratory to the field. In this application the PIC32 is the basic foundation for bringing together all the functions needed to monitor a beehive.

Project Design Overview

A PIC32 will be used as the system controller to monitor, analyze and control several functional areas of the beehive. Remote beehive configuration and status is primarily controlled via an embedded web server. Communication via internet may be either by wired or wireless means.

Functional areas monitored by the PIC32

- Outside the hive ambient conditions
- Inside the hive environmental conditions
- Health of the hive (brood size and bee traffic)
- Bee behavior for swarm management, detection / prevention
- Honey production
- Security of the hive
- Self diagnostics (temperature and voltage levels of the system controller, and functionality of sensors throughout the beehive)

Beehive environmental conditions that can be controlled

- Ventilation
- Temperature (heat or cool)
- Entrance opening for hive protection against predators
- Rain simulation (to keep bees inside hive)
- Induce vibration, smoke and sound in the hive to measure bee behavior patterns

Other possible monitored functions

- Weight of the hive for honey production
- Weight of an Individual bee for research of bee health
- Pollen collection (amount and type)
- Track and map the queen bee location in the hive
 - This can be done by gluing a micro LC resonance circuit to the queen's thorax. Circuitry around the hive can then track its location.
- Veroa mite count
 - A detector can be placed under the beehive to qualify droppings as mites and not bits of debris.
- Automatic feeder (syrup and patty) with consumption data logging

Project Design Approach

Hive conditions will be monitored using sensors for temperature and humidity. A photo sensor will monitor the daylight conditions outside the hive to determine available bee flight hours per day. The main focus of this project is to provide a platform by which bee behavior and health can be monitored, and additional sensors can be readily implemented to enhance bee research. Ambient conditions outside the hive can be optionally monitored using standard sensors for temperature, humidity, rain, wind and relative air pressure that can be imported through an RS-232 port. These optional functions such as the ambient environment and a GPS security function are secondary to the hive monitoring functions and are not implemented in the initial software. The hardware components to attach these features are provided for, but the main focus of this project is beehive health and behavior.

User interface is through an embedded web server. The user plugs a cable into the RJ-45 jack and connects to the internet. All beehive data collection, analysis and control functions are done via web pages created by the PIC32 system controller.

The 'inside the hive environmental' conditions will require some creative approaches to measure. A short note about beehive configurations is in order here to understand what is required. At a minimum the beehive is composed of a brood box that contains up to 10 frames. These frames may contain any one of the following three items as the bees build their hive's strength: Eggs (or brood as it matures), pollen to feed the brood, or honey (bee food). The bees build these materials in a predictable fashion as the hive's bee population increases.

The bees also produce a brown resin called propolis from tree sap. They use the propolis to seal things up in the hive, such as air gaps and foreign objects like sensors. Dealing with this is always a challenge and one needs to be creative to work around it.

Since one of the application areas of this project is being designed as a research tool, as such one of the requirements is to be able to add new types of monitoring capability without a major redesign. The I2C ports are very useful for this application. Some of these possible features include an enhanced brood monitoring function. This feature will require 8 temperature sensors per frame (x10 frames = 80 temperature sensors per brood box), this could translate to 160 temperature sensors for a typical 2 brood box hive. The reason for this is to map the brood cluster to determine the size (or health) of the hive. The bees keep the brood nest warmer than the rest of the hive to raise new bees. Generally the larger the brood nest the healthier the hive. A collapsing hive will have a smaller brood nest. This temperature data (from the eight MCP9801's, I²C high accuracy sensor) will be processed on each frame by a PIC12F series device that is inductively coupled to the PIC32 system controller. The PIC12F will receive its power and transmit its data back to the PIC32 system controller via the inductive coupling. This is necessary to allow each frame to be removed without hindrance for normal hive inspections. Connectors would be a reliability concern in this type of environment. The PIC32 will process the temperature data and present a colored frame mapping display of the results via the embedded web server. Air flow analysis through the hive will be done using an analysis of this same temperature data.

At the entrance of the hive is a special entrance module that the bees must pass through. The PIC32 polls this entrance module. The initial implementation of this entrance tunnel will count the number of bees leaving or entering the hive, data log it with analysis presentation on a web page.

With the future enhanced implementation of this bee tunnel, this module will count, weigh, and measure the bee's temperature as it leaves or enters the beehive. This is done to determine how many bees are lost during the day, how much pollen or nectar the bees are bringing back to the hive, and determine individual bee health upon leaving and returning. Bees land and walk on a landing board at the entrance of the beehive. With the bee passing through a small tunnel hole to allow just one bee at a time, two photo diodes are used to determine the direction the bee is moving when going through the tunnel. The count of the number of bees leaving or entering is recorded. A magnetically compensated weighing surface measures the fractional gram that a bee weighs and averages several of these to get the typical weight of the bees. The current through the magnetic coil which is used to push or pull this weighing surface back into its starting position is measured and is calibrated to fractions of a gram. A photo diode determines the position of the weighing

surface as it is positioned back into the unloaded position before the bee walked on it. Thus the weight of the bee is then known. The temperature of each bee is measured with a non-contact IR detector. This bee temperature is also data logged for analysis. Eight of these 'tunnel detectors' at the entrance should be sufficient to handle the beehive's typical traffic. A possible enhancement to this entrance module would be the addition of a color detector. With this detector an individual bee could be marked and tracked. It would then be possible to determine all of the above data and the life span for an individual bee.

Another possible feature that could be performed by the PIC32 would be an 'audio analyzer' function to monitor the buzz level of the hive. This will be to determine the queen state of the hive. Each beehive is basically one colony of bees and every colony has just one queen. Without a queen the hive dies. If a colony's queen dies or otherwise disappears there is a noticeable level of buzz difference. A small amount of smoke, vibration or jarring with a mechanical vibrator will trigger this buzz response. Swarm management will also use this data. Research into this area of the sound bees make started back in the 1950's. It would be nice to update this with current state of the art real time FFT analysis tools presented via a web page.

The weight of the hive is an indication of the honey production by the bees (can weigh +200lbs.). A magnetically energized beam balancing type of scale will be used for this task of weighing the beehive. Spring type scales and load cells are very sensitive to heat affecting their calibration and strain repeatability. Since these beehives are typically in harsh hot areas this would suggest the spring and load cell approach should be avoided if possible. The weight data processed by the PIC32 will be recorded for data logging and analysis presentation on a web page.

To conserve power several approaches can be used. In the case of the temperature sensors monitoring the brood frames, they only need to be powered up and turned on for a very small fraction of the time. Once every 15 minutes would be a likely period. In the case of the bee counters, the IR LEDs have a power enable circuit designed in to only power the LEDs when required. With LED lights it would be possible to pulse modulate them to reduce power consumption.

Security of the beehive is done by monitoring motion around the hive with PIR detectors. Tampering of the beehive is determined by a 'lid open' switch. Movement of the beehive is monitored by a motion or tilt switch. Not currently implemented, but any security breach could be used to trigger a GPS location of the hive, and trigger a cell phone call. This could be programmed to call the beekeeper, researcher or SWAT team and give a voice status of the problem.

Project Block Overview

PIC'n The Beehive System Controller Block Overview

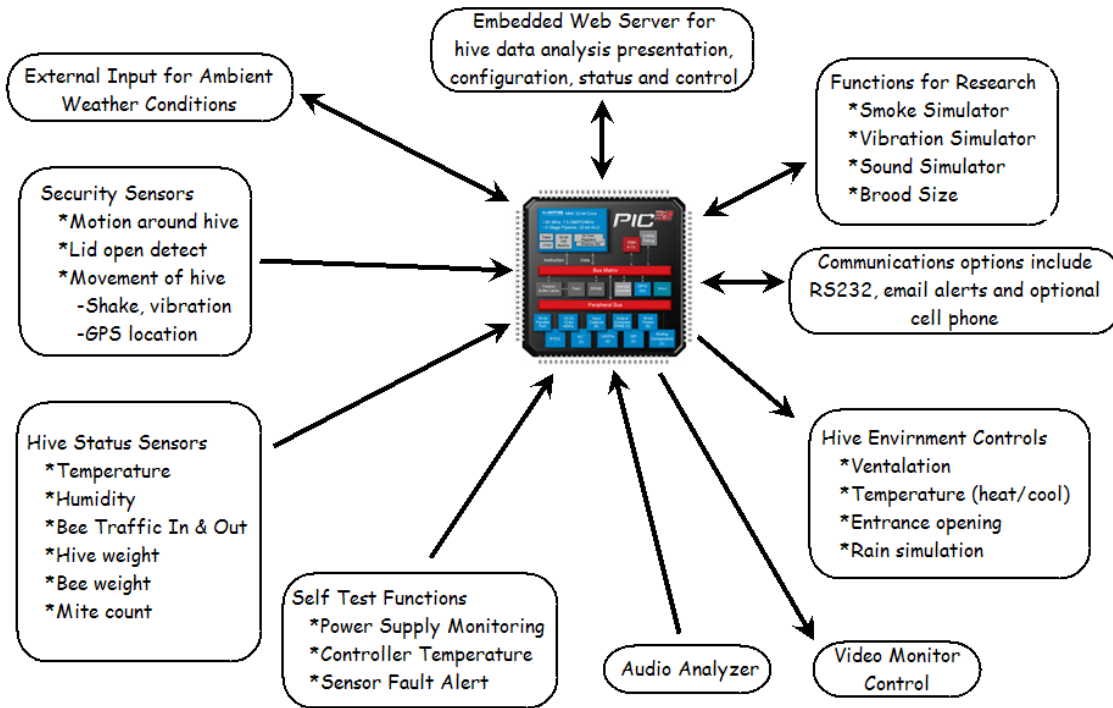


Figure 1 Project Block Overview

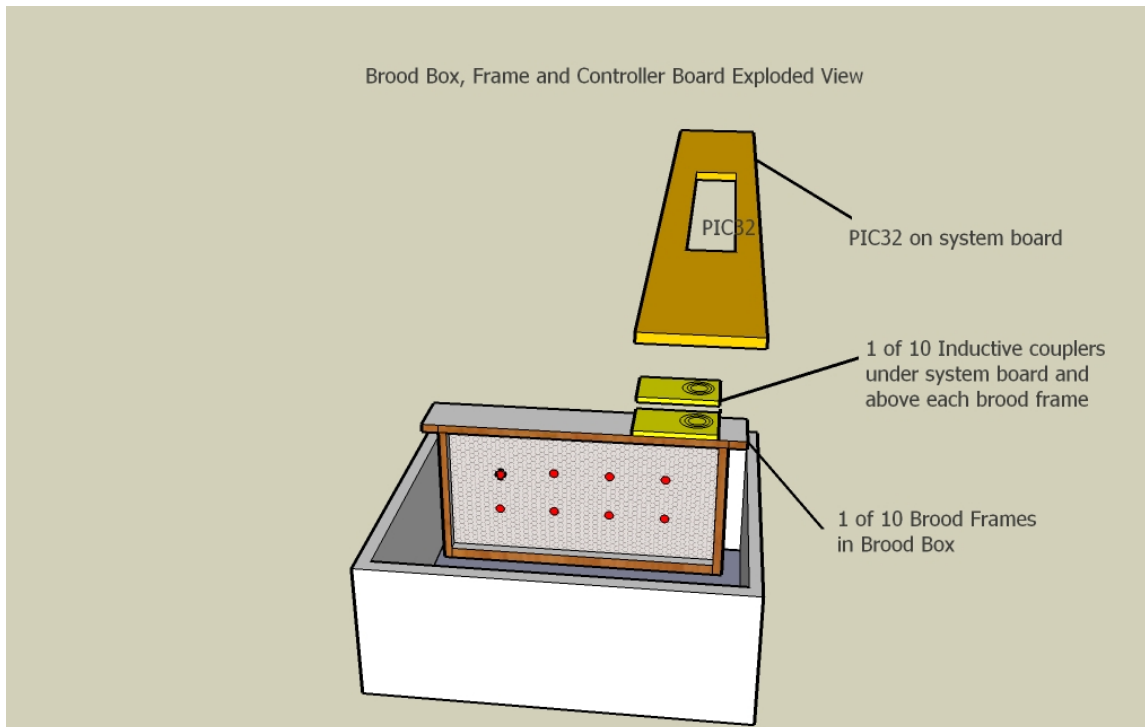


Figure 2 Inductively coupled brood frame temperature monitoring.

PIC'n The Beehive

Hardware Block Diagram

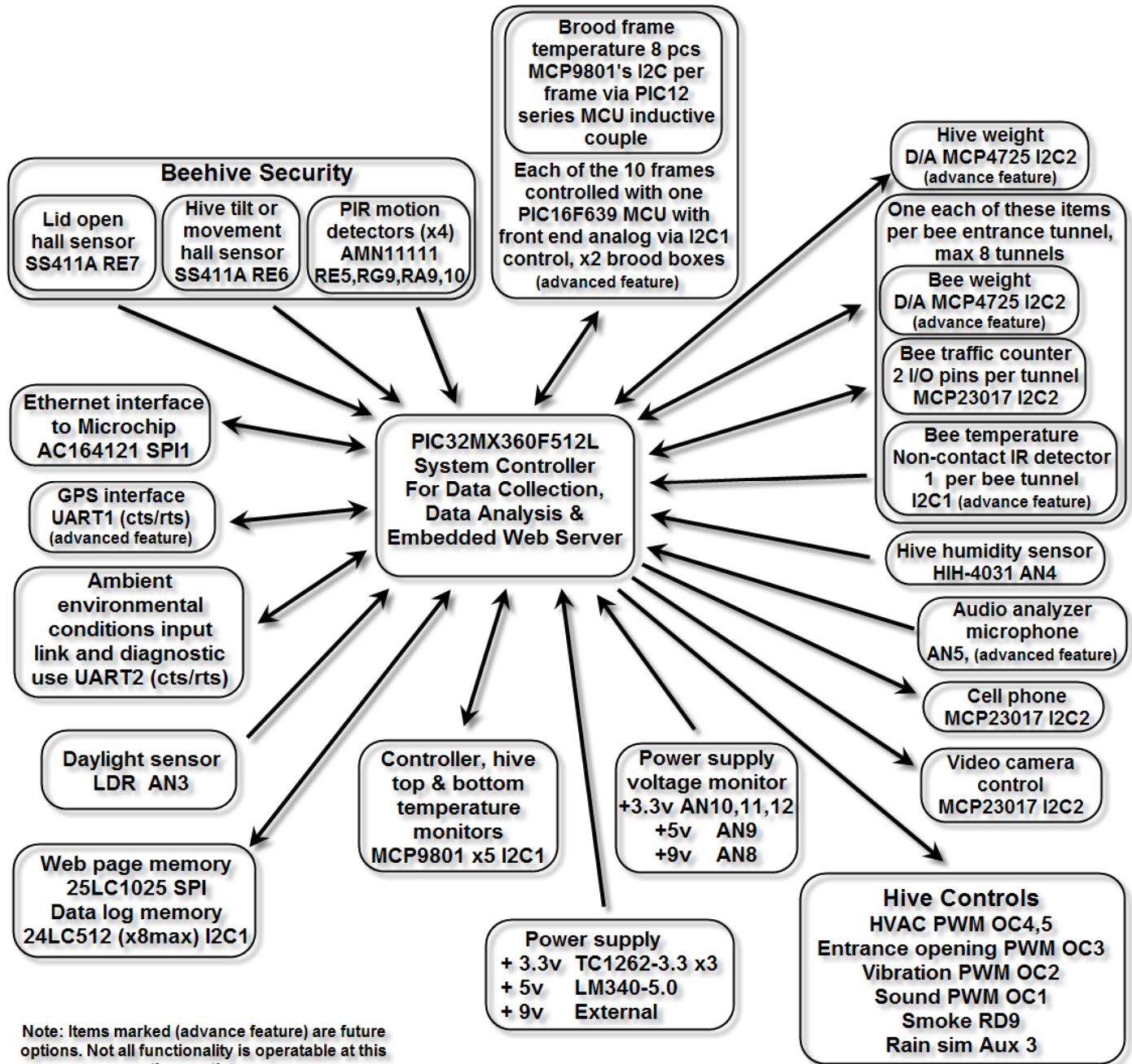


Figure 3 Hardware Block Diagram

Carl's Bee Management System (BMS) Reference Guide

This document has been updated to reflect all changes made during the previous Hardware Design Phase 2, the Software Development Phase 3, and the Final Prototype Phase 4 ending March 2009.

Project Summary

This Bee Management System (BMS) has a Microchip PIC32 configured as an embedded web server and system controller. The design objectives of this PIC32 Design Contest project has not only been met, but has exceeded performance expectations. The design objective was to provide a basic platform that could be easily expanded upon to add advance sensors and software features to monitor the brood size and health of the beehive.

The PIC32 works exceedingly well as a system controller in maintaining an active embedded server, monitoring a number of critical beehive sensors in real time, and then processing and presenting the data, both in a graphical and analytical display based on user request. A single PIC32MX360F512 processor, an AC164123 Ethernet interface module, couple of I2C and SPI EEPROM memory chips, and MCP23017 I/O expanders provide the hardware and software resources required to collect and control the functions of the beehive. The PIC32's performance and ability to connect up a large number of I2C devices allows for a very diverse selection of possible sensors to monitor the beehive.

The web server provides the user with hive status and control of a number of installed functions. These functions include not only reading parameters such as hive temperature, and bee traffic in and out of the hive, but also allows the beekeeper to control several hive environmental factors such as heating, cooling, feeding, vibration, entrance opening, simulating rain and etc.. Email security alerts and beehive performance data logging are selectable by the user. Basically this is what results when you turn a retired engineer loose that has beekeeping as a hobby.

Design Background

During the Hardware Development Phase a Microchip PIC32 Starter Kit DM320001, I/O Expansion Board DM320002, PICtail Plus Ethernet module AC164123, and a Prototype PICtail Plus Daughter Board AC164126 were used to design the BMS hardware and have it function as close as possible to an actual field installed BMS. A few of the initial features that were conceived during the Paper Design Phase were modified as the development progressed from both a hardware view and software view. As the software was being developed a more robust prototype board was built leveraging on the above hardware.

User Interface Summary

User communication to the beehive is via the Ethernet interface. The whole idea of this BMS was to provide an easier and more efficient method for the beekeepers or researchers to monitor their hives from a central location. A laptop could still be used in the field by simply connecting the BMS Ethernet to a wireless LAN. All of the user interface flows through the embedded web server.

All monitored data which is kept in the PIC32's RAM is periodically logged to on board EEPROMs for power fail recovery, as is the system configuration. Seven days of rolling history are maintained in memory. This data can be automatically emailed everyday to the user. In addition the user can request an immediate email data log, if desired.

Implemented Functions Currently Operational

- Precision temperature sensors that monitor the hive
 - Hive base temperature outside (1)
 - Hive base temperature inside (1)
 - Hive top temperature outside (1)
 - Hive top temperature inside (1)
- Humidity sensor (1)
- Daylight sensor (1)
- Bee traffic counters
 - Outgoing traffic (8)
 - Incoming traffic (8)

Visual graphs and data analysis are provided for on the home web page for all 8 of the above functions over a rolling 7 day period.

- Security sensor indication for:
 - Lid open detect (1)
 - Hive tilted detect (1)
 - Passive Infrared detectors (4)
- System controller monitors for:
 - Temperature (1)
 - Voltage monitors (1 implemented with upper and lower limits and email alert, but up to 5 available in hardware)
- Additional hardware hooks designed in for self testing of the bee traffic counters
- Web page control for configuration of:
 - Email data logs for manual and automatic transmission
 - Security sensors
 - Email alerts can be configured for the security sensors
 - Entrance openings
 - Cooling control
 - Heating control
 - Sound simulator
 - Vibration simulator
 - Rain simulator
 - Smoke simulator
 - Interior light control
 - Exterior light control
 - Bee Traffic Counters (8)
 - System status indicators (3)
 - Real Time Clock Calendar setting and calibration
 - Warm Restart Configuration (Save/Restore)
 - Data Log Memory (Save/Restore)

Features in Development for Enhanced Functions:

- Bee Scale Configuration
 - Web page control interface is present and hardware to drive the balance coils implemented, but not currently implemented in software as a completed system function at this time. Several different techniques are currently being evaluated. Ref. sheets 6, 7, & 8 of the schematics.
- Brood temperature monitors (Ref. sheet 13 of the schematics.)
- Total beehive weight scale
- GPS and the environmental subsystem on the RS-232 ports are not currently implemented. All of the hardware is present. The software can be added at a later time for these functions if needed.

Possible Product Configurations

As a result of designing, building and operating this project, it became obvious that there is a need for three variations of this product. The first level would be a very basic entry level device. The basic hardware functions could include the

processor with the Ethernet connection, basic hive sensors, temperature, light, humidity, and bee traffic counters as they are implemented in this contest entry. The actual cost of the bee traffic components is very minimal compared to the benefit of having the function. Extra EEPROM memory would not be necessary, instead the web pages could be stripped down to the basics, graphs and data analysis still functional, the PIC32's memory could be used, as this was the case in the early development of this design. A low cost, non-expandable single board design could be manufactured for an entry level product. A clever case design could implement the bee traffic counters as part of the case, and install it at the entrance of the hive. These basic functions would give the beekeeper a really good analysis of the strength of the hive.

The second level of product could have expansion connectors to allow for remote sensing and upgrades, such as the brood frame temperature monitoring, and other various functions such as controlling hive hardware features, i.e. the optoisolators and LED indicators.

The third level of product could be designed with the researcher in mind. It could have additional power supply capability and prototype areas for advanced sensor development. This current contest design falls closely into this product area, with the base functions implemented and working, while providing additional hardware and software hooks to expand it.

Project Overview

The PIC32 Design Challenge rules were pretty simple in that they stated the design had to be based on the *PIC32 Starter Kit*. I noticed that it did not say just PIC32. I didn't want to have to do the schematic design twice, once for a prototype and once for a production implementation. So the schematic design was implemented to allow for connections to the PIC32 Starter Kit hardware as well as a production version that could use just a PIC32 without the starter kit circuit board. As it turned out this worked very well into the design and debugging.

The following *PIC32 Pin Assignment* table works for either implementation. In the design contest prototype implementation two pages of the schematic are not used; page 1 System Processor and the voltage regulators on page 2. Analog voltage monitoring circuits for just the +12v (actually +9v on the prototype board) R201 and R202, the +5v R203 and R204, and just the one +3.3v P32_Vdd R205 and R206 on page 2 are implemented.

ICSP, JTAG and trace pins on the PIC32 were reserved for those functions. Three digital/analog I/O's are left unused for possible additional sensors later on. All other pins on the PIC32MX360F512 100-pin device are used in addition to three 16-bit MCP23017 I2C I/O expanders.

Hardware Design

The following table provides the relationship between the PIC32's processor pins and their function as used in this project.

PIC32MX360F512 Pin & Function Assignments

Pin No.	Name	Assignment	Functional Description
1	RG15	RG15	Output, Ethernet CS (Use /U1CTS pin47 RD14 in proto)
2	VDD	3.3v	P32_PWR
3	PMD5/RE5	RE5	Input, PIR 1 motion sensor
4	PMD6/RE6	RE6	Input, Hall sensor SS411A tilt or movement sensor
5	PMD7/RE7	RE7	Input, Hall sensor SS411A lid open sensor
6	T2CK/RC1	RC1	Output, CS0 for SPI2 memory 25LC1024
7	T3CK/RC2	RC2	Output, CS1 for SPI2 memory
8	T4CK/RC3	RC3	Output, CS2 for SPI2 memory
9	T5CK/RC4	RC4	Output, CS3 for SPI2 memory
10	PMA5/SCK2/CN8/RG6	SCK2	SPI2 clock
11	PMA4/SDI2/CN9/RG7	SDI2	SPI2 data in ←SPO2 memory
12	PMA3/SDO2/CN10/RG8	SDO2	SPI2 data out →SPI2 memory
13	-MCLR	-MCLR	CPU Reset, ICSP
14	PMA2/-SS2/CN11/RG9	RG9	Input, PIR 2 motion sensor
15	VSS	Ground	
16	VDD	3.3v	P32_PWR
17	TMS/RA0	TMS	Reserved for JTAG
18	INT1/RE8	INT1	INTB for I2C2 MCP23017 Bee scale balance
19	INT2/RE9	INT2	Ethernet Int
20	C1IN+/AN5/CN7/RB5	AN5	Audio in, buffer 2
21	C1IN-/AN4/CN6/RB4	AN4	Hive humidity, buffer 1
22	C2IN+/AN3/CN5/RB3	AN3	Daylight sensor, buffer 0
23	C2IN-/AN2/-SS1/CN4/RB2	-SS1	reserved

24	PGC1/EMUC1/AN1/CN3/RB1	PGC1	reserved
25	PGD1/EMUD1/AN0/CN2/RB0	PGD1	reserved
26	PGC2/EMUC2/AN6/OCFA/RB6	PGC2	ICSP
27	PGD2/EMUD2/AN7/RB7	PGD2	ICSP
28	PMA7/VREF-/CVREF-/RA9	RA9	Input, PIR 3 motion sensor
29	PMA6/VREF+/CVREF+/RA10	RA10	Input, PIR 4 motion sensor
30	AVDD	+3.3V	Analog +3.3V
31	AVSS	Ground	Analog ground
32	C1OUT/AN8/RB8	AN8	+12v supply monitor)9v proto) V=0.17544 x PWR_12V, 12vmon, buffer 3
33	C2OUT/AN9/RB9	AN9	+5v supply monitor V=0.5 x PWR_5V, 5vmon, buffer 4
34	CVREFOUT/PMA13/AN10/RB10	AN10	+3.3v supply monitor V=0.5 x P32_VDD, 3vmonp32, buffer 5
35	PMA12/AN11/RB11	AN11	+3.3v supply monitor V=0.5 x IO_PWR,3vmonIO, buffer 6
36	VSS	Ground	
37	VDD	+3.3V	P32_PWR
38	TCK/RA1	TCK	Reserved for JTAG
39	-U2RTS/BCLK2/RF13	-U2RTS	UART2 RTS control
40	-U2CTS/RF12	-U2CTS	UART2 CTS control
41	PMA11/AN12/RB12	AN12	+3.3v supply monitor V=0.5 x FRAME_PWR, 3vmonFRAME, buffer 7
42	PMA10/AN13/RB13	AN13	Configured spare 1 analog, buffer 8
43	PMALH/PMA1/AN14/RB14	AN14	Configured spare 2 analog, buffer 9
44	PMALL/PMA0/AN15/OCFB/CN12/RB15	AN15	Configured spare 3 analog, buffer 10
45	VSS	Ground	
46	VDD	3.3v	P32_PWR
47	CN20/-U1CTS/RD14	-U1CTS	UART1 CTS control, used for GPS
48	-U1RTS/BCLK1/CN21/RD15	-U1RTS	UART1 RTS control, used for GPS (Use for Ethernet CS in proto)
49	PMA9/U2RX/CN17/RF4	U2RX	UART2 receive, used for ambient

			environmental link and diagnostic use
50	PMA8/U2TX/CN18/RF5	U2TX	UART2 transmit, used for ambient environmental link and diagnostic use
51	U1TX/RF3	U1TX	UART1 transmit, used for GPS communication
52	U1RX/RF2	U1RX	UART1 receive, used for GPS communication
53	SDO1/RF8	SDO1	SPI1 Ethernet control AC164123
54	SDI1/RF7	SDI1	SPI1 Ethernet control AC164123
55	SCK1/INT0/RF6	SCK1	SPI1 Ethernet control AC164123
56	SDA1/RG3	SDA1	I2C1 data, used for temperature sensors controller & brood PIC16F639's
57	SCL1/RG2	SCL1	I2C1 clock, used for temperature sensors
58	SCL2/RA2	SCL2	I2C2 clock, used for MCP23017 IO expanders, bee traffic modules
59	SDA2/RA3	SDA2	I2C2 data, used for MCP23017 IO expanders, bee traffic modules
60	TDI/RA4	TDI	Reserved for JTAG
61	TDO/RA5	TDO	Reserved for JTAG
62	VDD	3.3v	P32_PWR
63	OSC1/CLKI/RC12	OSC1	8mHz crystal in
64	OSC2/CLKO/RC15	OSC2	8mHz crystal out
65	VSS	Ground	
66	INT3/RA14	INT3	Interrupt open drain from PIC16F's brood
67	INT4/RA15	INT4	INA & INB open drain from I2C2's MCP23017 IO expanders U501 traffic counter,U909 Aux1 to Aux10 IO's
68	IC1/RTCC/RD8	RTCC	Output, RTCC Alarm clock
69	IC2/RD9	RD9	Output, Smoke module on/off
70	IC3/PMCS2/PMA15/RD10	RD10	Output, Ext IO reset
71	IC4/PMCS1/PMA14/RD11	RD11	Output, I2C1 WP, data log memory

			write protect
72	OC1/RD0	OC1	PWM, Sound module PWM drive
73	SOSCI/CN1/RC13	SOSCI	32kHz crystal in
74	SOSCO/T1CK/CN0/RC14	SOSCO	32kHz crystal out
75	VSS	Ground	
76	OC2/RD1	OC2	PWM, Vibration module PWM drive
77	OC3/RD2	OC3	PWM, Entrance module PWM servo drive
78	OC4/RD3	OC4	PWM, Heater module PWM drive
79	PMD12/IC5/RD12	RD12	Output, status LED-yellow, toggles for each bee entering hive
80	PMD13/CN19/RD13	RD13	Output, status LED-yellow, toggles for each bee leaving hive
81	PMWR/OC5/CN13/RD4	OC5	PWM, Hive cooling module fan drive
82	PMRD/CN14/RD5	RD5	Output, hive status LED-green, self test/diagnostic
83	PMD14/CN15/RD6	RD6	Input, PB switch for diagnostics, BUTTON_SW1_IO
84	PMD15/CN16/RD7	RD7	Input, PB switch for diagnostics, BUTTON_SW2_IO
85	VCAP/VDDCORE		Vddcore bypass caps
86	ENVREG	3.3v	P32_PWR
87	PMD11/RF0	RF0	Output, hive exterior light LED-red & optoisolator cell
88	PMD10/RF1	RF1	Output, hive interior light LED-red & optoisolator
89	PMD9/RG1	RG1	Output, hive status LED-yellow, self test/diagnostic
90	PMD8/RG0	RG0	Output, hive status LED-red, self test/diagnostic
91	TRCLK/RA6	TRCLK	Trace Clock, connector
92	TRD3/RA7	TRD3	Trace Data3, connector
93	PMD0/RE0	RE0	Output, LED-red & optoisolator for electric fence on/off
94	PMD1/RE1	RE1	Input/Output, manual override for electric fence, 3-position switch hi-lo-open

95	TRD2/RG14	TRD2	Trace Data2, connector
96	TRD1/RG12	TRD1	Trace Data1, connector
97	TRD0/RG13	TRD0	Trace Data0, connector
98	PMD2/RE2	RE2	Output, status LED-red & optoisolator, PIR Motion Alarm
99	PMD3/RE3	RE3	Output, status LED-red & optoisolator, Tilt Alarm
100	PMD4/RE4	RE4	Output, status LED-red & optoisolator, Tamper Alarm
Analog enable 3-4-5-8-9-10-11-12-13-14-15			

Figure 4 PIC32MX360F512 Pin & Function Assignments

(Continued on next page)

Hardware Overview

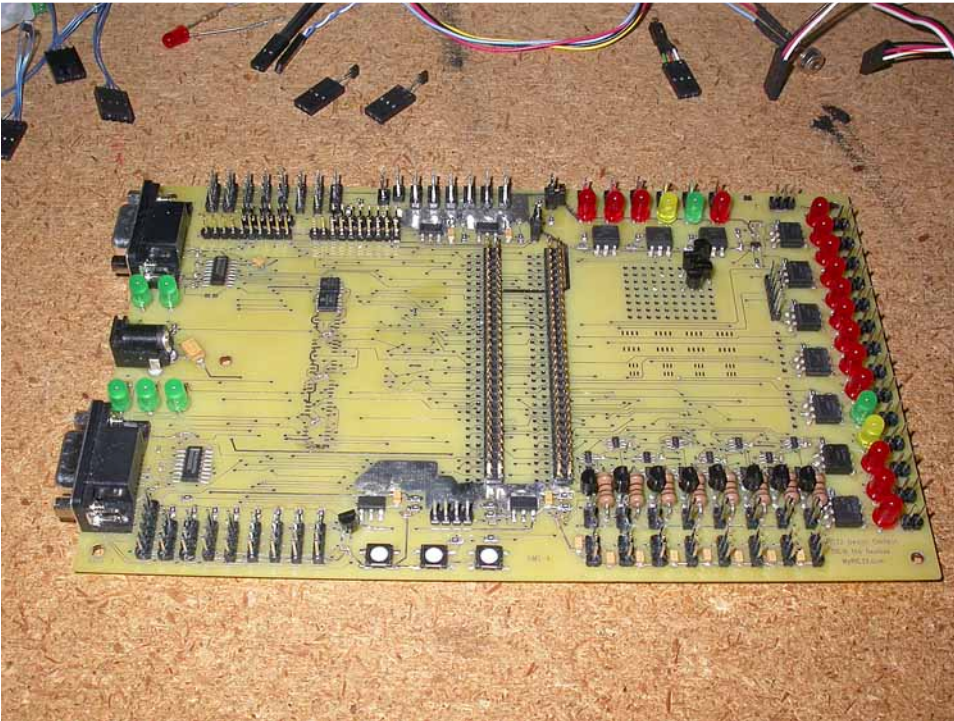


Figure 5 Populated prototype project board

Layout considerations included placement (top right corner of board) of the buffer amp for the humidity sensor to be placed far away from any heat sources on the board. The voltage regulators have extra ground plane copper poured around them to act as a heat sink.

Not all EEPROMs needed to be installed at this point in the software requirements. Two of eight possible I2C EEPROMs are visible to the left of the connectors, midway to the RS-232 connector board edge.

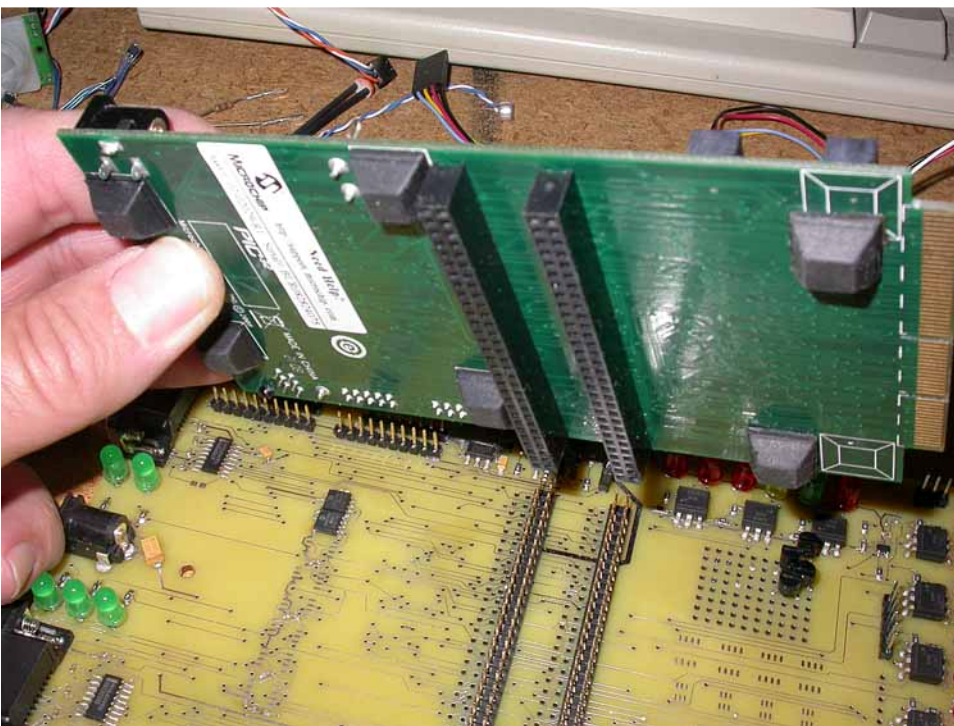


Figure 6 60-pin connectors allow stacking of the boards

The photo at left shows the two 60 pin connectors added to the bottom side of the PIC32 I/O expansion board. On the project contest prototype board notice the additional rows of holes next to the header pins. This allows for another set of connectors to be installed on the bottom side of the contest prototype board for yet more prototype boards to be stacked for additional functions not yet conceived. The neat thing is that these two 60 pin connectors allow access to all pins on the PIC32 processor for future expansion

Some extra prototyping area was provided on the contest prototype board for quick fixes. Even as I was laying out the board I was already planning some uses for it.

The really neat thing about this configuration is that I am able to use the tools available in my home lab to design, breadboard, and convert to a functioning product without a major redesign. The extremely fine pitch of the PIC32 processor package would otherwise make that just about impossible. My plan is to replace the PIC32 starter kit and expansion board with another board that just has the plug in PIC32 pim module (as used on the Explorer 16 Development Board) with necessary clock and Ethernet circuitry added. That little PIC32 pim module is more cost effective and doable to design with at the PCB level for home labs than trying to deal with the fine pitch parts. Altogether I am very pleased with the available hardware and tools that I have used to get this project up and running.

Schematic Details page 1- System Processor

A Microchip PIC32MX360F512 (U101) is used as the system controller. It services a number of hive sensors and controls and processes this data for presentation to the user via its embedded web server. It also monitors its own health in terms of system voltages and temperatures. It should be noted that sheet 1 represents what is provided by the PIC32 Starter Kit board. The only problem is that the 32 kHz crystal is really not supplied with the starter kit. A 32 kHz crystal and capacitors had to be installed for the RTCC to work. Also because the PCB traces are actually routed off the PIC32 Starter Kit board, I cut the traces at the crystal to eliminate the extra trace capacitance cause by this extra long routing. This then gave me the proper range of calibration of the 32 kHz crystal.

Schematic Details page 2- Power Distribution

A 12 volt power source was chosen because this is a pretty standard voltage to find in outdoor equipment. If needed a solar backup system would be easy to locate based on 12 volts. As little as 5 volts may be used to power the system. Therefore, 4 NiCad or NiMH batteries could be used to provide battery backup in power outages or remote locations.

VR201 drops the incoming unregulated 7.5 to 15 volts down to a regulated 5 volts for some of the systems sensors that require this voltage. It also provides a regulated source for three 3.3 volt regulators, VR203 provides 3.3 volts to the PIC32, VR204 provides 3.3 volts to many of the sensors and I/O expanders, and VR205 provides 3.3 volts to the PIC16F's that drive the frame temperature RFID subsystem. Each of these voltages are monitored by AN8(+12v), AN9(+5v), AN10(+3.3v), AN11(+3.3v), and AN12(+3.3v) on the PIC32.

Additionally on this page is the power switching transistor for the diagnostic test and future power reduction scheme of the LEDs used in the bee traffic counters. The LEDs will be pulse modulated to reduce power consumption.

A test switch is provided to allow a manual test and visual observation of LEDs for all voltage supplies. Only the LED for the raw +12 volts is powered on all the time, the others only light when the button is press to conserve power.

Schematic Details page 3- Ethernet & UART Interfaces

MOD301 is an Ethernet Interface Module AC164123. It is used to provide the main communications to the BMS. This same module lends itself to be used for both prototyping and production because it allows for placement of the RJ45 connector at any convenient location on the beehive, if installed on the end of a cable separate from the system controller board. Also experience shows that interfaces are generally the first thing to fail in harsh environments. Having the interface as a separate module allows for easy repair.

U302 a RS232 level shifter provides the UART2 interface which is used for a subsystem link (J301 DB-9 connector) to download ambient environmental conditions such as wind speed, inches of rain and outdoor temperature at the location of the beehives. The ambient environmental subsystem is not part of this PIC32 design contest.

During development, UART2 also provides a link for software debug.

U303 provides the UART1 interface for a GPS subsystem. Depending on the actual GPS model used it may or may not require this RS232 level shifter. The purpose of the GPS fills the need of identifying the location the beehive is located for research data and possible theft of the beehive. As part of this subsystem a compass could be implemented as well to

know the direction the beehive entrance is facing. The GPS subsystem is not part of this PIC32 design contest project.

Schematic Details page 4- Controller Environment Sensors

U401 is a MCP9801, a 12-bit digital temperature sensor that is located on the system controller board. This provides a reference temperature for inclusion along with other beehive data being logged if desired.

Four additional temperature sensors (U402 – U405) monitor the temperature at the base entrance inside of the hive, the top cover inside, the hive base temperature outside, and the hive top outside temperature. The idea here is it provides a well rounded view of the environment immediately in and around the beehive. Eight connectors are actually installed on the printed circuit board. This allows for expansion of up to eight temperature sensors, since their I2C addressing mode allows this.

S401 is a SS411A Hall Effect sensor. It is used as a lid open detect sensor. A bias magnet must also be used next to the hall sensors for correct operation. A magnet on the lid provides the trigger for this sensor. Another sensor S402 is used as a tilt indicator. A magnet is suspended from above the sensor; if the beehive tilts off center it will trigger the sensor. This is useful to know if other sensors are used to detect liquid levels such as feeding syrup. The need is to keep the beehive level to accurately measure liquid consumption. Also if beehive weight is implemented the beehive should be as level as possible. It's possible to provide automatic beehive leveling using the Aux I/O's described later in this document.

PIR1 located at the entrance (and optionally PIR2-PIR4 for additional direction location) are AMN11111 passive infrared motion sensors. These are used to detect the movement from a person or animal near the beehive.

Alarm conditions can be set according to the above sensors.

H406 is a HH-4031 humidity sensor used to record the internal beehive humidity. It is buffered by U407 a MCP6001 general purpose op amp. This data is logged along with the temperature data.

J401 is a configuration jumper for future electric fence control. The electric fence can be set to always on, always off, or set via the web server. A 3-state jumper configuration is used to detect which option the beekeeper selects, since only one of these options is allowed.

S404 and S405 are push button switches for diagnostic use. Currently they are programmed to increment the incoming and outgoing bee traffic counters.

Schematic Details page 5- Bee Traffic Counters

This Bee Traffic Counter subsystem is built into the basic BMS. A IR LED is positioned a bee's width across from an IR phototransistor to form a beam detection circuit. Two of these configurations are placed slightly shorter than a bee's length apart. This forms a detection circuit that differentiates which way the bee is traveling, in or out of the beehive. If the bee breaks beam A before breaking beam B, we know the direction of travel. If the bee is blocking beam B before breaking beam A, we know the bee is traveling the other way. In the initial design of this BMS, up to eight of these 'bee traffic tunnels' are designed in. Because this bee traffic subsystem is easily built around a single MCP23017 I2C I/O expander we can always add more groups of eight just by giving the additional MCP23017 another address on A0-A2. This subsystem could be located remotely from the PIC32 system controller at the beehive entrance. We need only provide power, ground, and two control lines to operate it.

Schematic Details pages 6, 7 & 8- Bee Scale Modules- advanced feature

The Bee Scale Module allows the bees to be weighed going in and out of the hive. As such it is physically located as part of the 'bee traffic tunnel' described above. We can know when a bee is on the scale when both IR pairs are blocked, and we know which way the bee was traveling at that point, in or out of the beehive. The scale works by driving a 2N2222 transistor (Q601) with a calibrated current through a coil (L601) as provided by a MCP4725 D to A converter (U602) to induce a magnetic field. Currently there are multiple configurations of this external sub-system being designed and evaluated. In the simple implementation the bee is walking on a magnetic strip of metal that is fixed at one end and free to dip at the other end. At the free end, the strip is bent at a 90 degree angle downward to form an 'L' shape. This metal

strip is so light that the weight of the bee and metal strip blocks an IR LED / phototransistor pair detection circuit (D801 & Q801). The coil is energized with just enough current to bring the metal strip back up and breaks the beam. With the calibrated current we will know the weight of the bee. This method also allows for easy and constant recalibration when no bee is on the scale, since the normal position un-energized can be built to exist in either the down or up position to block the IP position sensor. There would always be a small amount of calibrated bias current going through the coil to hold it just above the beam break position, which is signaled back through a MCP23017 (U801) I/O expander. In another implementation of this scale design there are small bias magnets used. One magnet 'floats' above the other, on which the bee being weighed is in contact with. Again the coil current is used to 'balance' the system. This method removes the drift problem with using a metal strip with some spring in it that would change over time.

Diode D601 provides back-EMF protection in the event of power loss while the coil is energized.

As in the Bee Traffic Counters, there are up to eight of these Bee Scale Modules in the initial design. A MCP23017 (U801) I/O expander is used for address enabling the DAC's and for balance detect of each bee scale.

Schematic Details page 9 & 10- Status & Control IO's & Interfaces

A light sensor CdS cell R901 is used to measure daylight hours. This helps to determine possible bee flight hours each day.

Beehive events are visually indicated via a number of LED's to aid the beekeeper in maintenance of the hive's many sensors. The status of all of these sensors is available via the web interface, but there is also a need for the beekeeper to identify and correct problems in the field.

Currently there are 20 beehive events that will produce a visual LED indication and an optoisolator output.

They are:

- Tamper alarm- if the lid is opened
- Hive movement- if the tilt sensor is triggered for any reason
- PIR motion alarm- triggered by one of PIR motion sensors
- Bee leaving hive- status LED toggles each time a bee leaves the hive
- Bee entering hive- status LED toggles each time a bee enters the hive
- Smoke module- enables the smoke module to produce smoke (optional subsystem, not included in this project)
- Rain module- enables an electric sprinkler valve to spray water on the hive. Makes bees think it's raining so they don't leave the hive. Useful to keep your bees in your hive if pesticide spraying is being done nearby.
- Entrance module servo power- enables/disables the power going to the entrance servos to conserve power and longevity of the servo. Entrance servo is used to close the beehive entrance during colder weather and reduce predator invasions.
- Electric fence module- in bear country an electric fence around the beehive is a must. This module is combined with the PIR motion sensors and reduces power requirements for the electric fence.
- Alarm, output can be used for anything.
- Sound module PWM drive- output can be buffered to drive any kind of transducer.
- Vibration module PWM drive-output can be buffered to drive a mechanical motor/vibrator device to shake the hive.
- Entrance module PWM servo drive- output can be used to directly drive a hobby type servo if the optoisolator is replaced with jumpers if needed. Optoisolator provides the greatest flexibility.
- Heater module PWM drive- output used to heat the beehive in cold climates as needed.
- Hive cooling module PWM fan drive- output used to power a small brushless fan for internal air circulation to cool the hive.
- Hive status module GREEN- provides a visual indication of hive status. An external light of another indicator can be driven from the optoisolator.
- Hive status module YELLOW- provides a visual indication of hive status. An external light of another indicator can be driven from the optoisolator.
- Hive status module RED- provides a visual indication of hive status. An external light of another indicator can be driven from the optoisolator.
- Hive exterior light module- provides a means to enable a light on or around the exterior of the hive.

- Hive interior light module- provides a means to enable a light on interior of the hive.

When an event is triggered an optoisolator is energized to provide isolated drive capability to the outside world. A 4N26/H11A1 (U901) is used for this purpose. This provides 1,500Vrms isolation and 30v/100mA current capability. Additional peripheral circuitry maybe required for each different application of this available output.

Since all of the above LED's and associated optoisolators are controlled by software, any of the functions can be changed to something else. Such as if an automatic patty feeder is needed, the optoisolator output can be used to drive that function. Any input sensing can be entered through the Aux5-8 I/O ports (U909). A typical usage for the hive status GREEN-YELLOW-RED modules might be to give visual indication of the beehive's internal temperature, too high or too low. In the current software implementation they are used to indicate email traffic being sent.

Schematic Details page 11- SPI Memory

U1105 – U1112 are 24LC512 I2C EEPROMs used to store the massive amount of data being collected. The beekeeper can download this data via the embedded web server at any time.

U1101 – U1104 are 25LC1024 SPI EEPROMs used to store the web pages.

Actual number of memory devices loaded on the PCB depends on the program requirements.

Schematic Details page 12- PICtail Plus pin mapping to PIC32 functions

J1201 is a 120 pin connector on the PICtail Plus I/O expansion board.

Schematic Details page 13- Bee Temperature Subsystems

The current BMS design does not include the RFID portion of the Bee Temperature Subsystem, although the non-contact IR bee temperature function may be included. The magnitude of the RFID part of the project makes it a project all its own. The hardware to interface to this as a future upgrade is provided in the basic BMS design, but at this time the RFID subsystem hardware is not complete. The schematic on page 13 shows where this part of the overall BMS project is headed. When this option is installed it will add 40 more PIC processors and 160 more MCP9801 temperature sensors! Talk about data collection :)

Schematic Details page 14 & 15- J10 & J11 net mapping to PIC32 functions

J141 and J151 map the nets found on J10 and J11 on the PIC32 PICtail Plus I/O expansion board. This is mainly provided so that the circuit board layout software has knowledge of the nets to map to the pins correctly.

Schematic Details page 16- Prototype area and Servo Power Connections

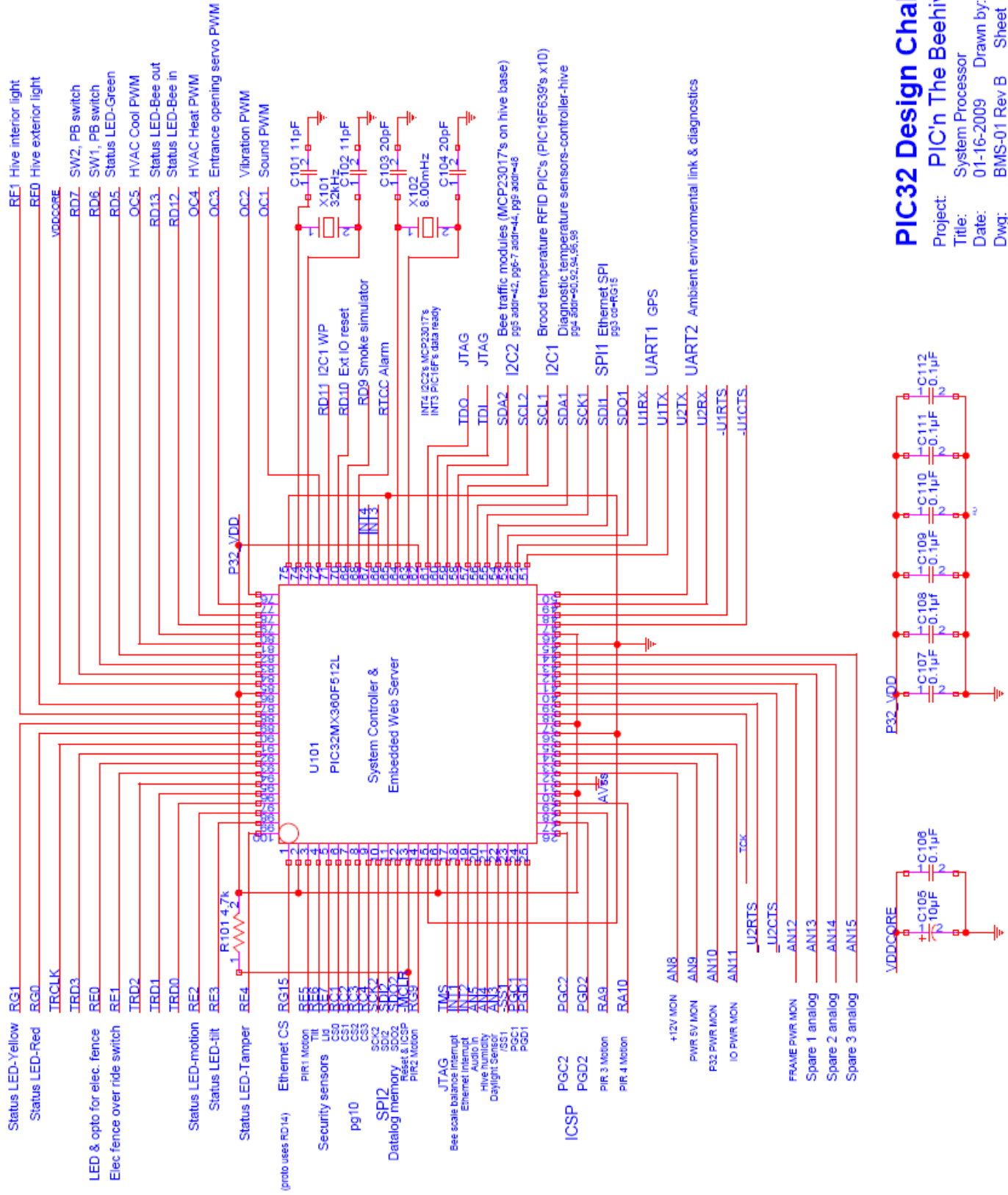
The servo power connectors are located on this sheet. The nets SP1 to SP8 are the switched servo power sources that come from PNP transistors on sheet 10. The PNP's on sheet 10 are driven by NPN's getting their drive signal from the I/O expansion port (U910) on sheet 9. This allows us to multiplex the power going to the servos using one PWM I/O.

Schematics

The following table lists the following schematics:

Schematic Page Number	Description
1	System Processor
2	Power Distribution
3	Ethernet & UART Interfaces
4	Controller Environment Sensors
5	Bee Traffic Counters
6	Bee Scale Modules 1-4
7	Bee Scale Modules 5-8
8	Bee Scale Module Addressing
9	Status & Control IO's & Interfaces 1
10	Status & Control IO's & Interfaces 2
11	SPI Memory
12	PICtail Plus pin mapping to PIC32 functions
13	Bee Temperature Subsystems
14	J10 & J11 Net Mapping Interface, page 1 of 2
15	J10 & J11 Net Mapping Interface, page 2 of 2
16	Proto Type Area & Servo Power Connections

Table 1 Schematic pages



PIC32 Design Challenge

Project: PIC'n The Beehive
 Title: System Processor
 Date: 01-16-2009 Drawn by: Carl
 Dwg: BMS-01 Rev B Sheet 1 of 16

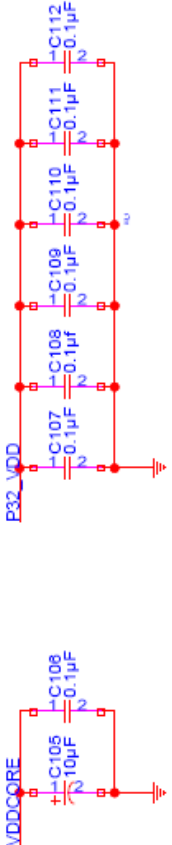


Figure 7 Schematic page 1

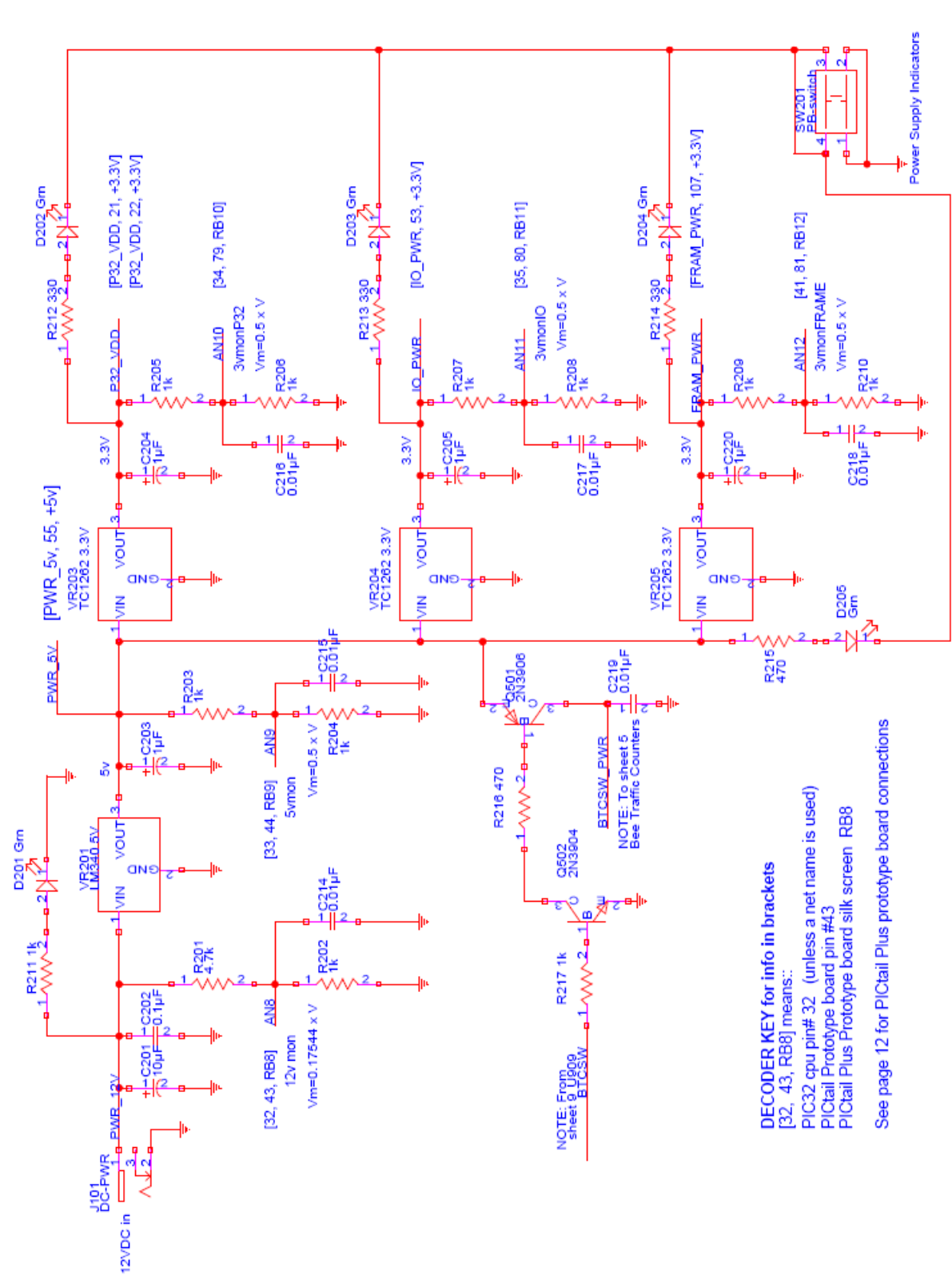
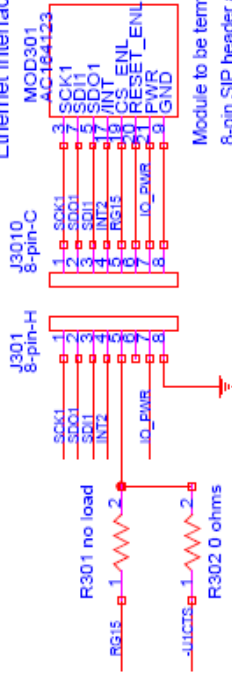


Figure 8 Schematic page 2

Ethernet Interface Module



NOTE: Alternet CS on proto RD14 /U1CTS

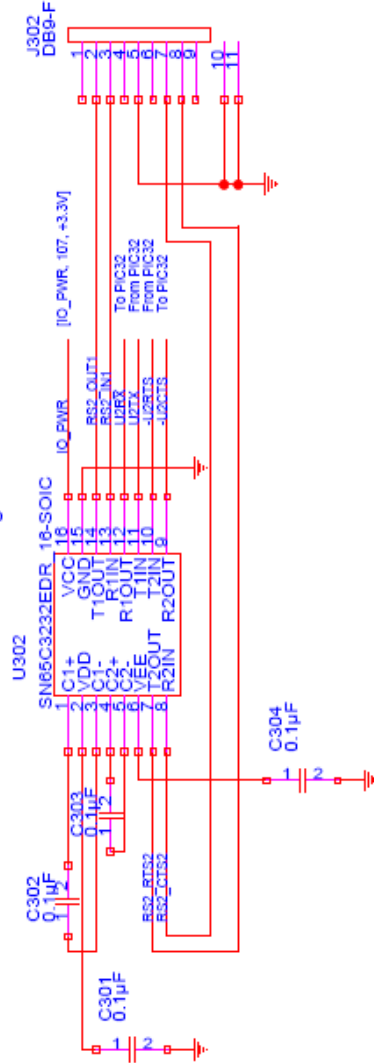
Module to be terminated to an 8-pin SIP header at the end of a cable soldered to the AC164123 Ethernet PiCtail Plus module.

DECODER KEY for info in brackets

[27, 76, RB7] means::

- PIC32 cpu pin# 27 (unless a net name is used)
 - PiCtail Prototype board pin # 76
 - PiCtail Plus Prototype board silk screen RB7
- See page 12 for complete pinout details

Ambient Environmental Link and Diagnostic use UART2 Interface

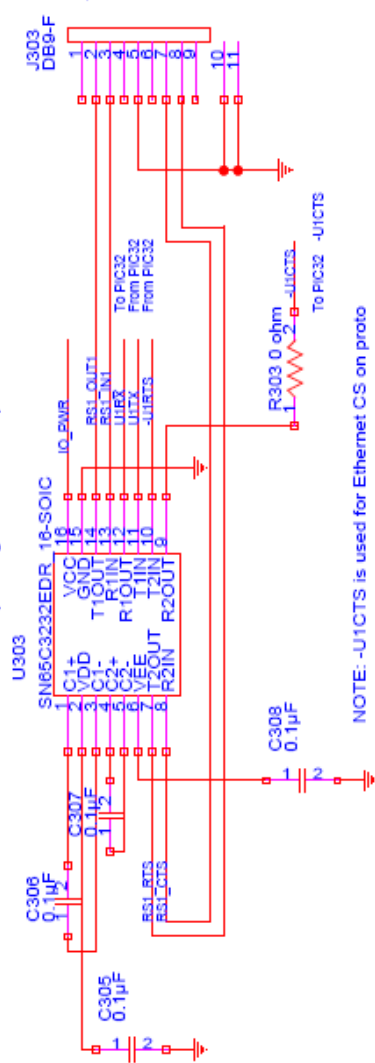


- U2RX [49, 34, RF4]
- U2CTS [30, 20, RD3]
- U2CTS [40, 31, RF12]

<<<This RD3 is incorrect silk screen on board. It should be RF13

GPS Module UART 1 Interface

Possible no-load depending on GPS IO requirement



- U1RX [52, 2, RF2]
- U1CTS [51, 4, RF3]
- U1CTS [46, 20, RD15]
- U1CTS [47, 19, RD14]

NOTE: -U1CTS is used for Ethernet CS on proto

PIC32 Design Challenge

Project: PIC'n The Beehive
 Title: Ethernet & UART Interfaces
 Date: 01-16-2009 Drawn by: Carl
 Dwg: BMS-03 Rev B Sheet 3 of 16

Figure 9 Schematic page 3

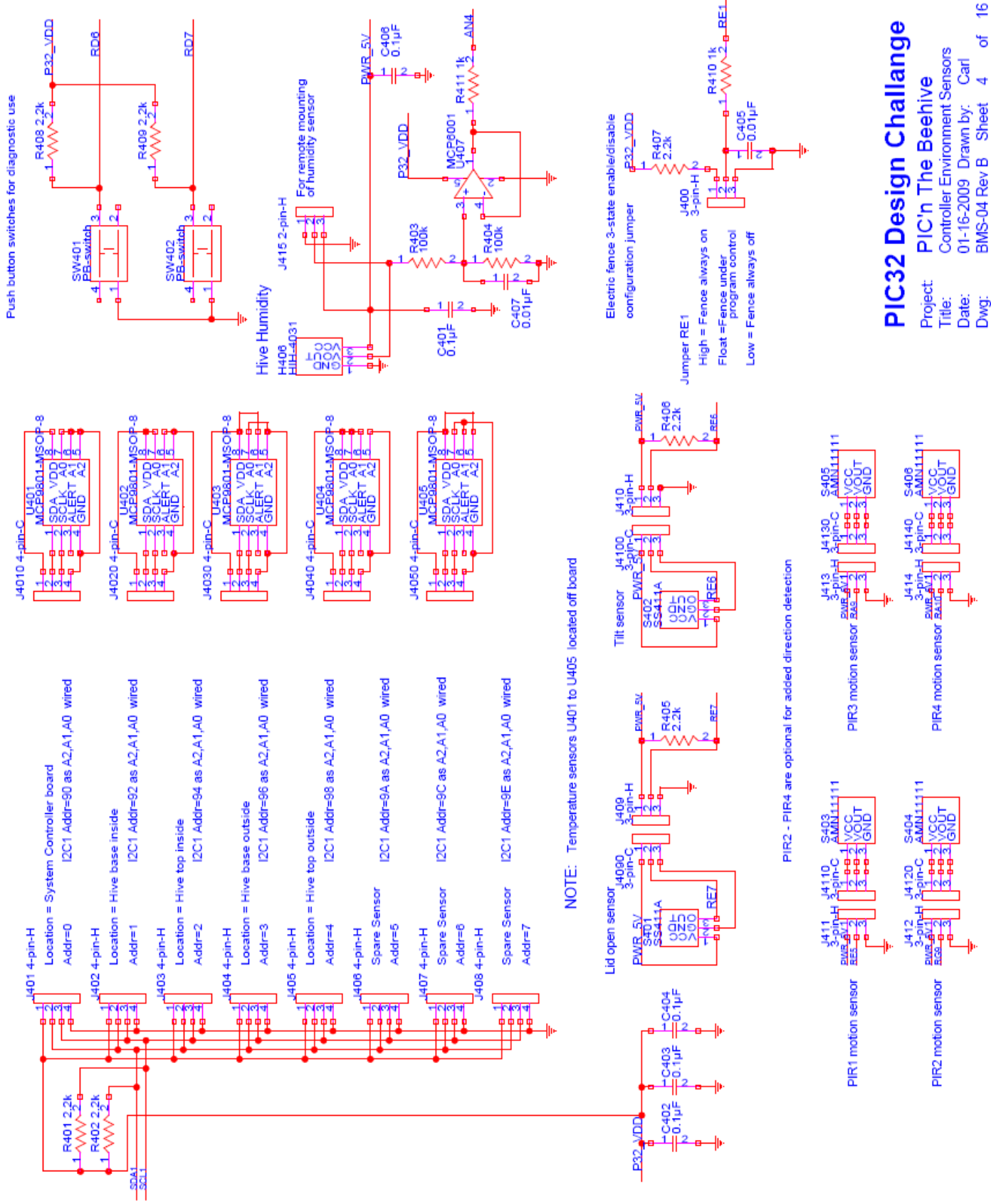
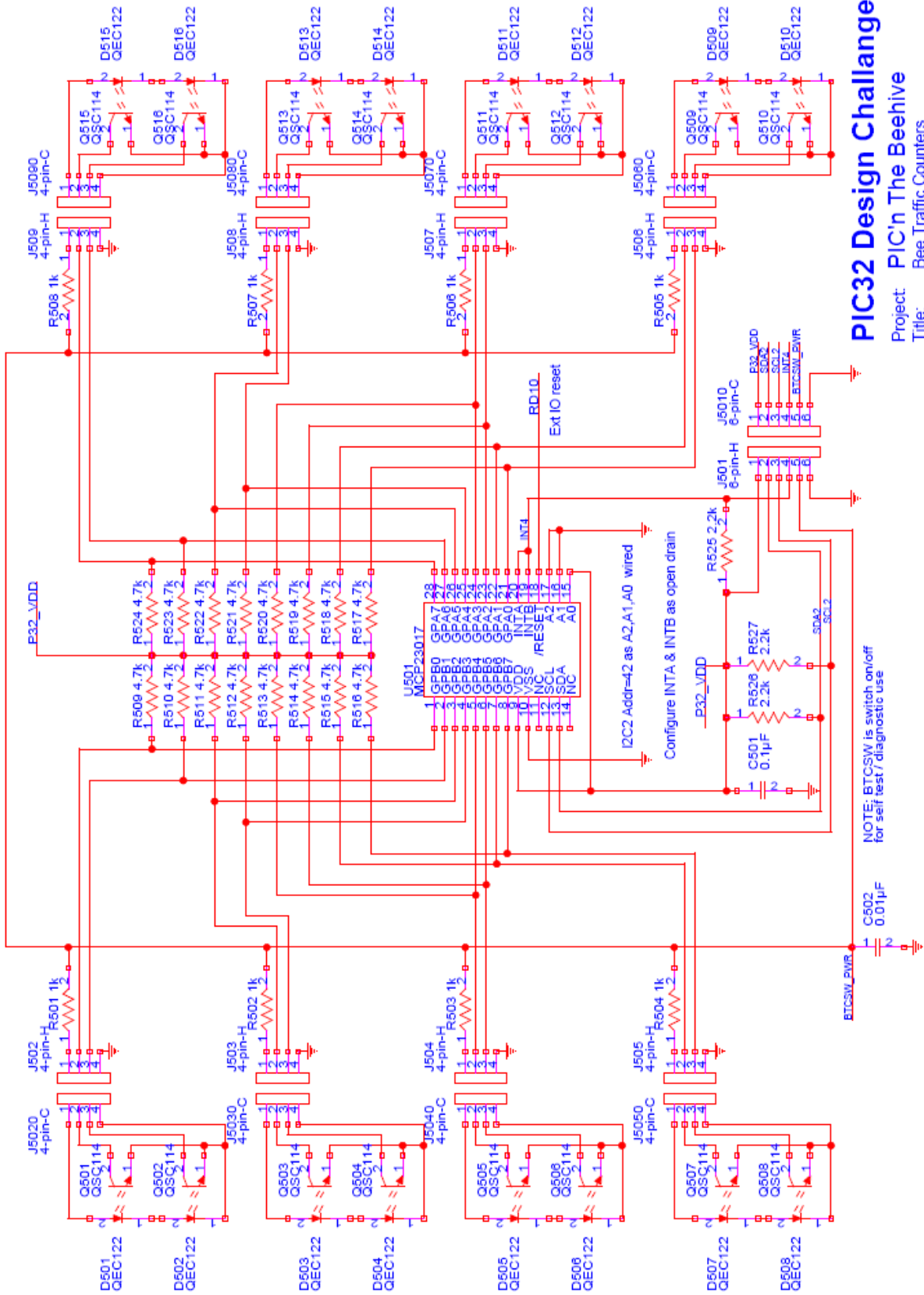


Figure 10 Schematic page 4

PIC32 Design Challenge

Project: PIC'n The Beehive
 Title: Controller Environment Sensors
 Date: 01-16-2009 Drawn by: Carl
 Dwg: BMS-04 Rev B Sheet 4 of 16



PIC32 Design Challenge
 Project: PIC'n The Beehive
 Title: Bee Traffic Counters
 Date: 01-16-2009
 Dwg: BMS-05 Rev B

Sheet 5 of 16
 Drawn by: Carl

Figure 11 Schematic page 5

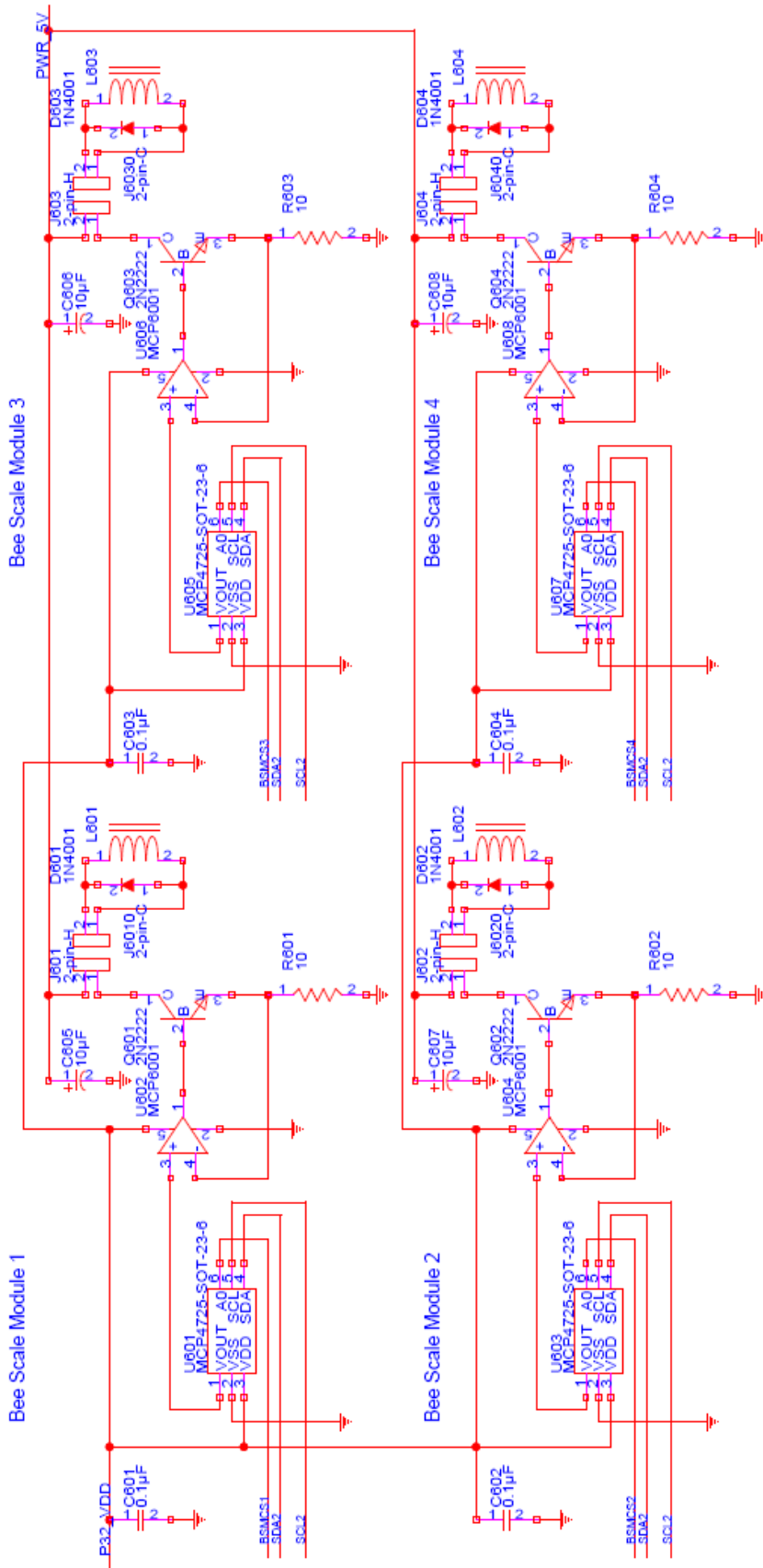
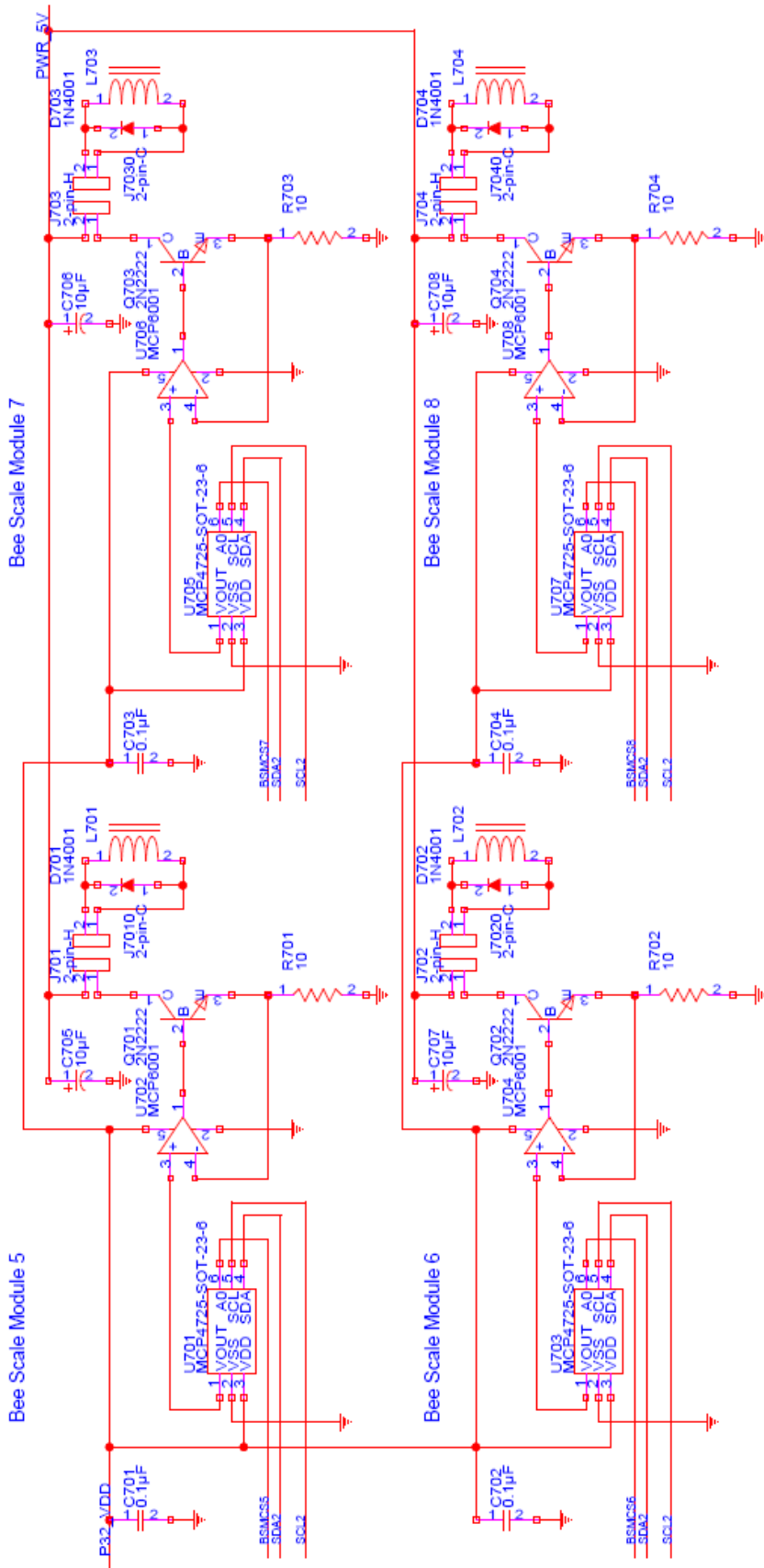


Figure 12 Schematic page 6

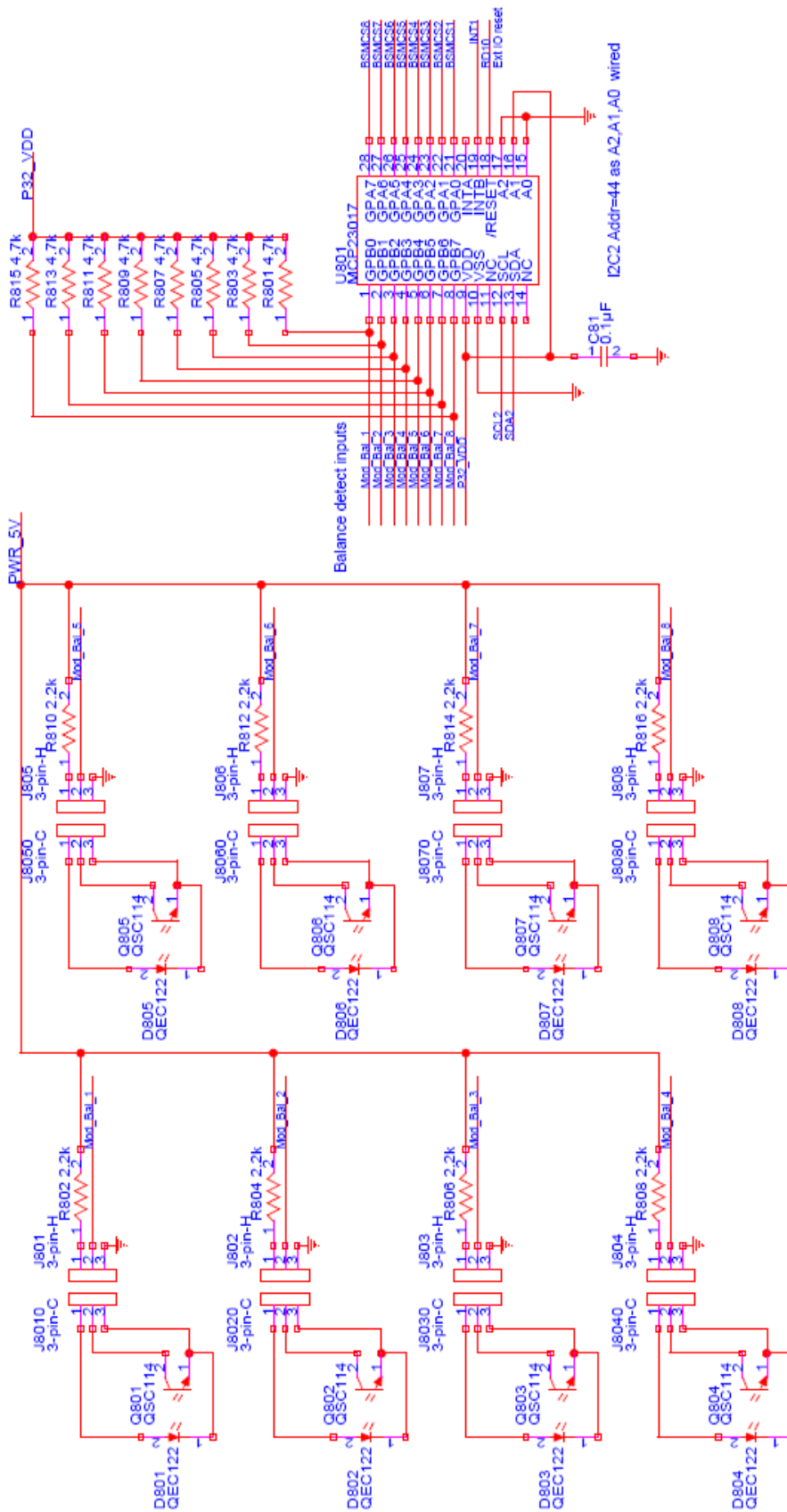
PIC32 Design Challenge

Project: PIC'n The Beehive
 Title: Bee Scale Modules 1 to 4
 Date: 01-16-2009 Drawn by: Carl
 Dwg: BMS-06 Rev B Sheet 6 of 16



Project: PIC'n The Beehive
 Title: Bee Scale Modules 5 to 8
 Date: 01-16-2009 Drawn by: Carl
 Dwg: BMS-07 Rev B Sheet 7 of 16

Figure 13 Schematic page 7



PIC32 Design Challenge

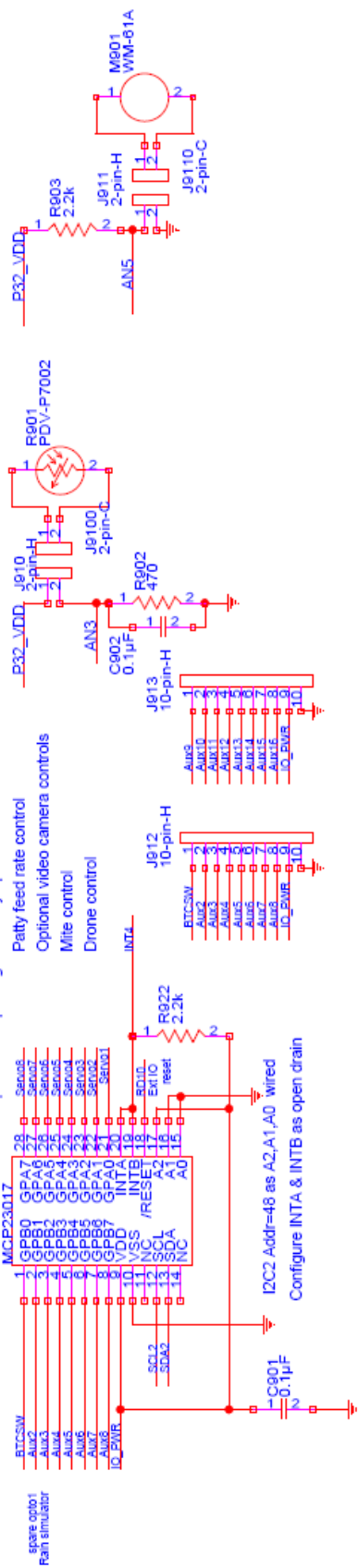
Project: PIC'n The Beehive
 Title: Bee Scale Module Addressing
 Date: 01-16-2009 Drawn by: Carl
 Dwg: BMS-08 Rev B Sheet 8 of 16

Figure 14 Schematic page 8

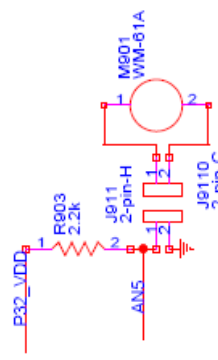
NOTE: BTCSSW is a control to enable/disable the power to the Bee Traffic Counter LEDs. Ref. sheet 2

Auxiliary IO's & future use

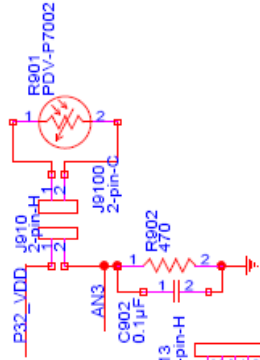
- Hive leveling motor control
- Feed syrup level
- Patty feed rate control
- Optional video camera controls
- Mite control
- Drone control



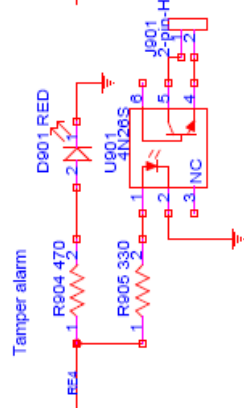
Microphone



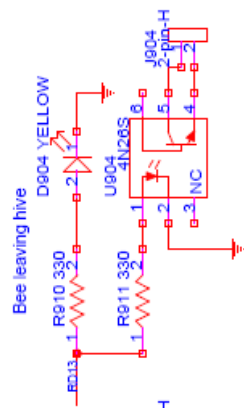
Daylight Sensor



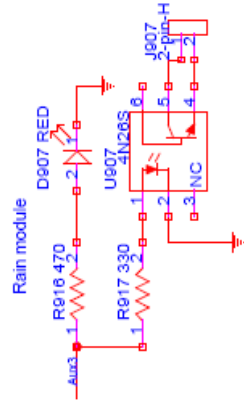
Tamper alarm



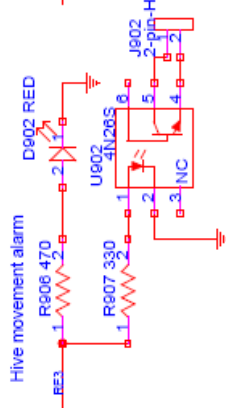
Bee leaving hive



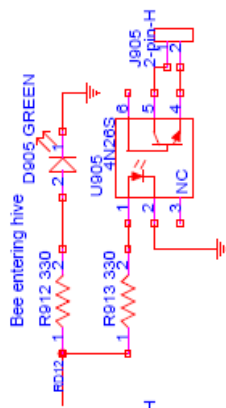
Rain module



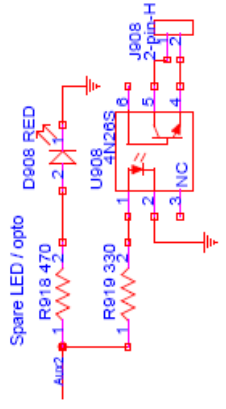
Hive movement alarm



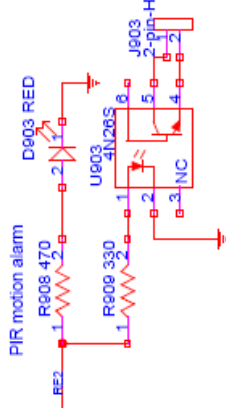
Bee entering hive



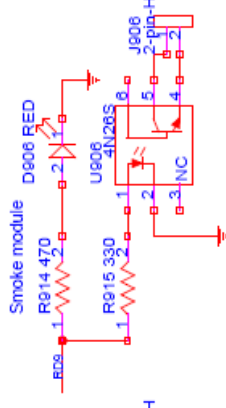
Spare LED / opto



PIR motion alarm



Smoke module



Electric fence module

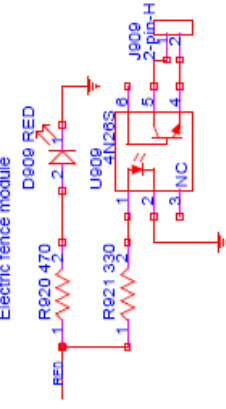


Figure 15 Schematic page 9

PIC32 Design Challenge

Project: PIC'n The Beehive
 Title: Status & Control, IO's & Interfaces 1
 Date: 01-16-2009 Drawn by: Carl
 Dwg: BMS-09 Rev B Sheet 9 of 16

NOTE: Optoisolators may be loaded as needed for use.

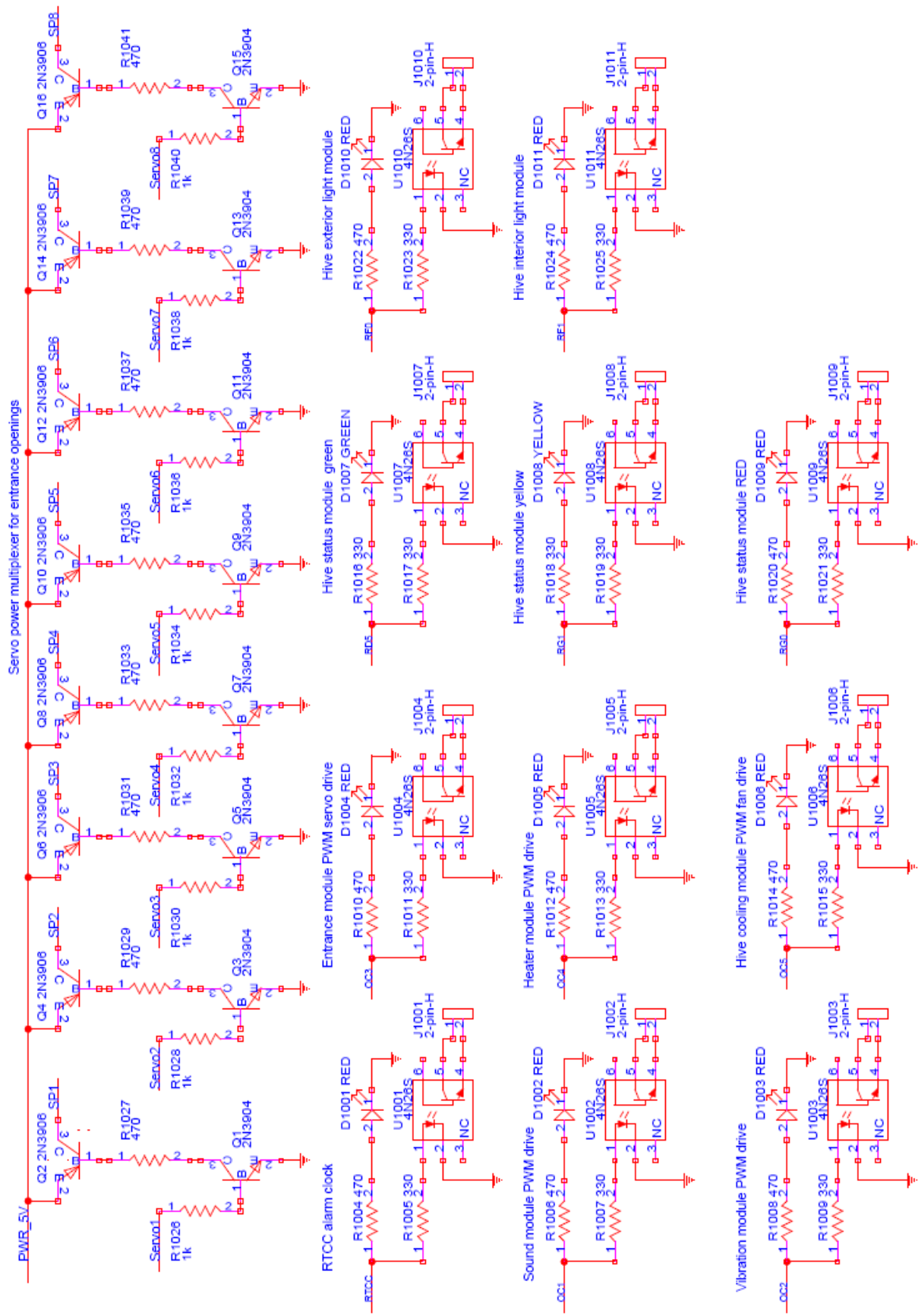
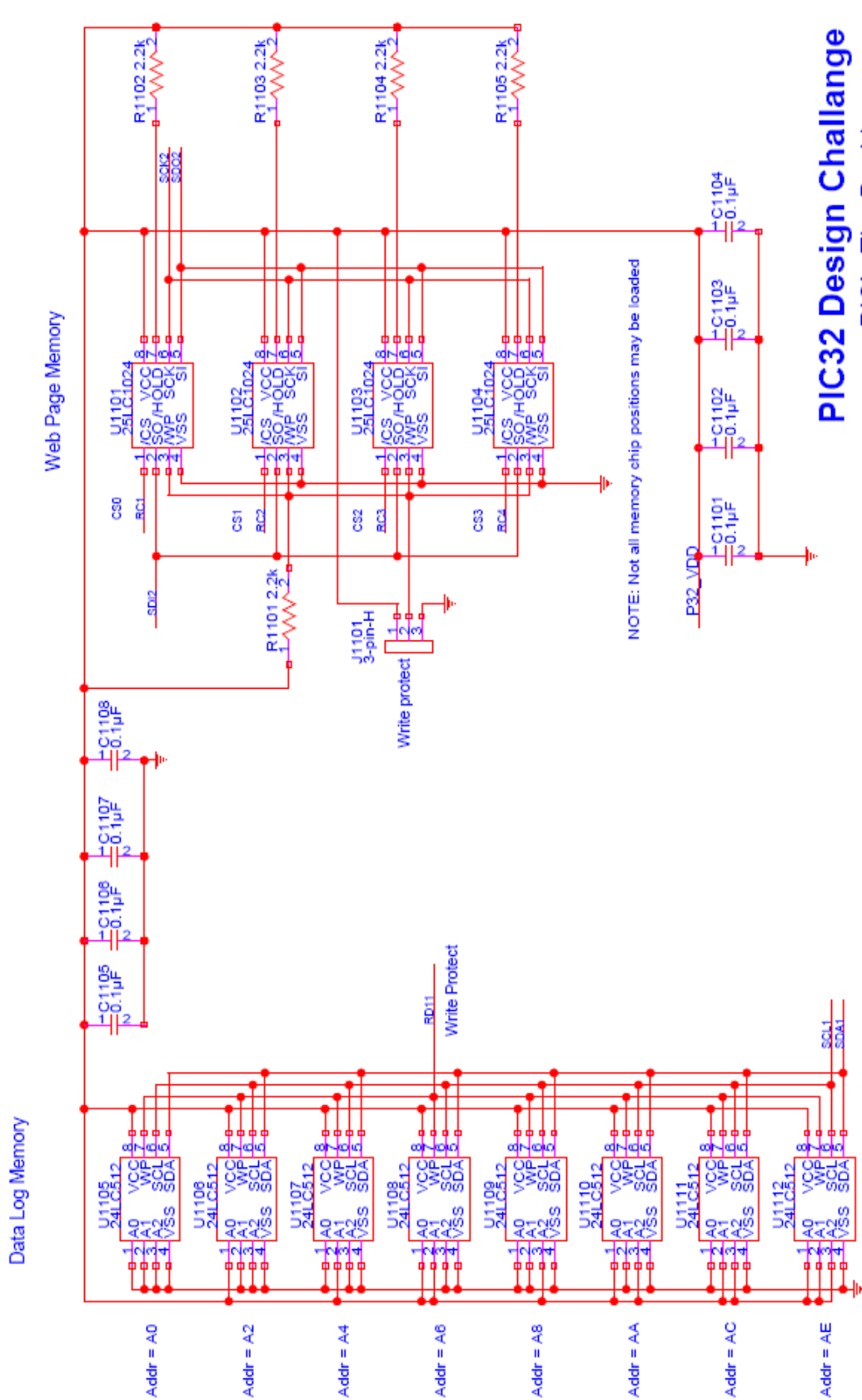


Figure 16 Schematic page 10

PIC32 Design Challenge

Project: PIC'n The Beehive
 Title: Status & Control IO's & Interfaces 2
 Date: 01-16-2009 Drawn by: Carl
 Dwg: BMS-10 Rev B Sheet 10 of 16

NOTE: Optoisolators may be loaded as needed for use.

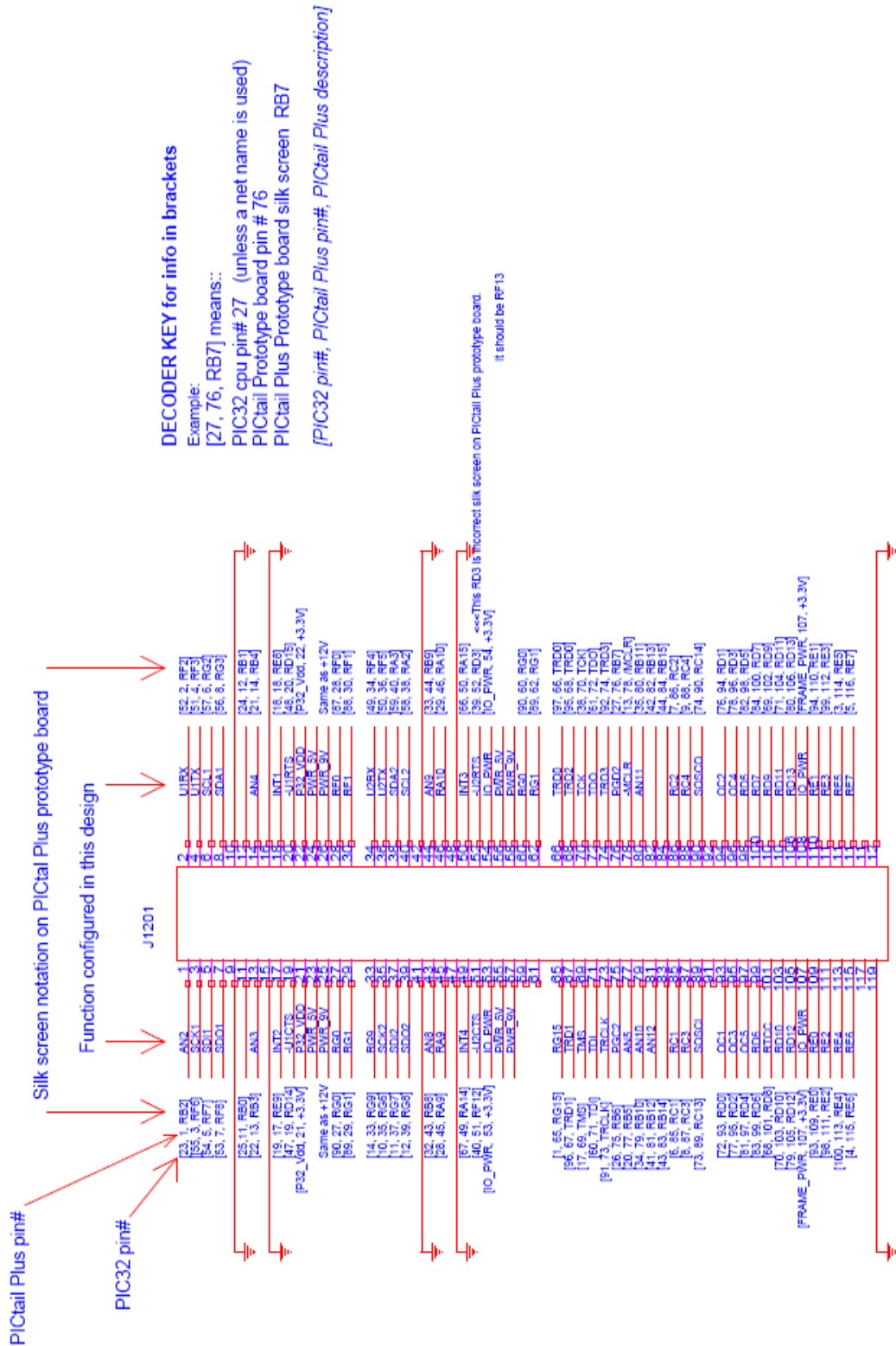


PIC32 Design Challenge

Project: PIC'n The Beehive
 Title: Web Page & Data Log Memory
 Date: 01-16-2009 Drawn by: Carl
 Dwg: BMS-11 Rev B Sheet 11 of 16

Figure 17 Schematic page 11

PIC32 Starter Kit used with I/O Expansion board DM320002.
 All connections to the PIC32 Starter Kit are made through the PICtail Plus expansion connector.
 A PICtail Plus prototype board AC164126 is used to make these connections.
 The net names indicate how the PIC32 pins are used function wise.



DECODER KEY for info in brackets
 Example:
 [27, 76, RB7] means::
 PIC32 cpu pin# 27 (unless a net name is used)
 PICtail Prototype board pin # 76
 PICtail Plus Prototype board silk screen RB7

[PIC32 pin#, PICtail Plus pin#, PICtail Plus description]

It should be REF13

PIC32 Design Challenge

Project: PIC'n The Beehive
 Title: PICtail Plus pin mapping to PIC32 function
 Date: 01-16-2009 Drawn by: Carl
 Dwg: BMS-12 Rev B Sheet 12 of 16

Figure 18 Schematic page 12

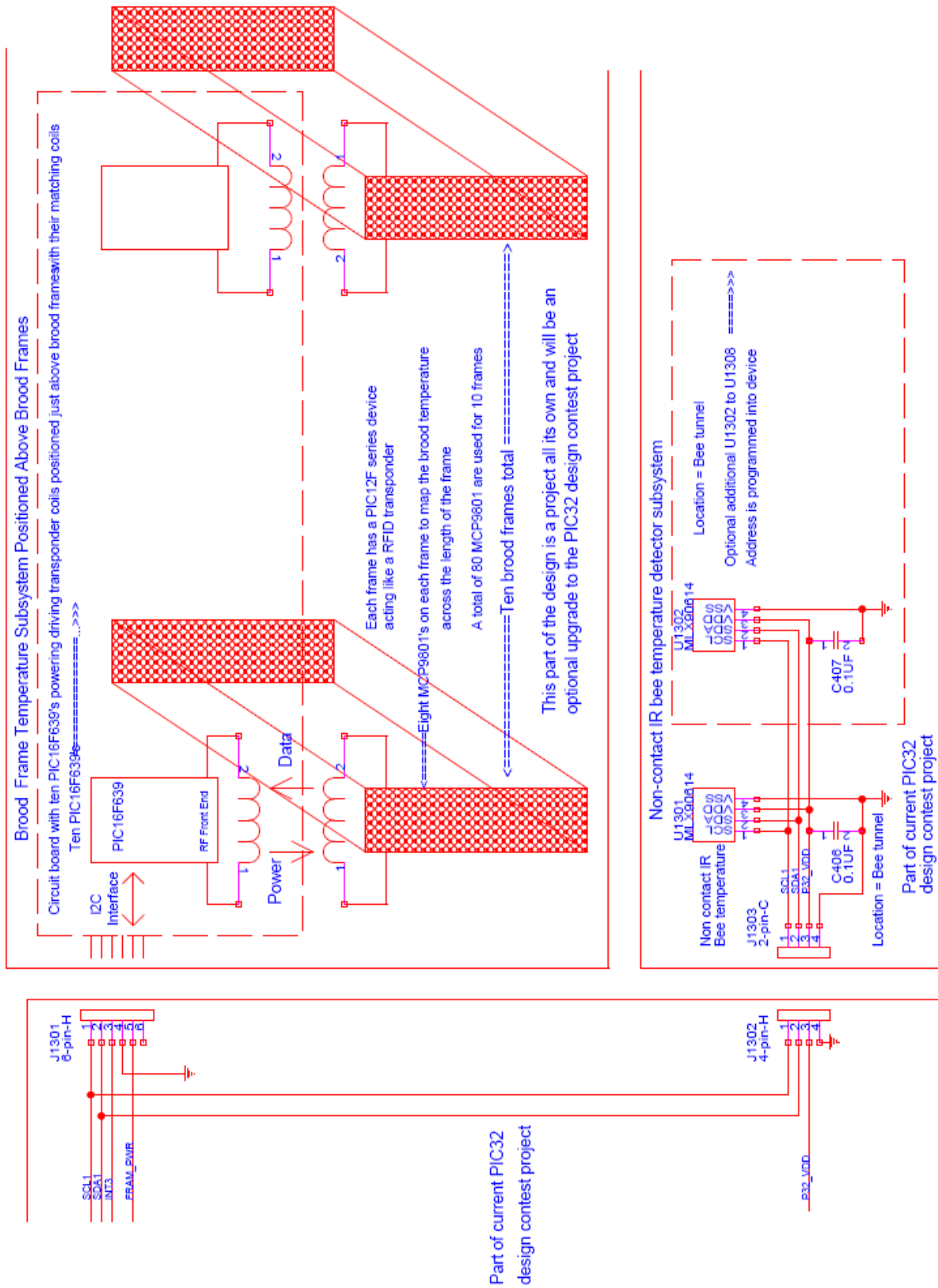
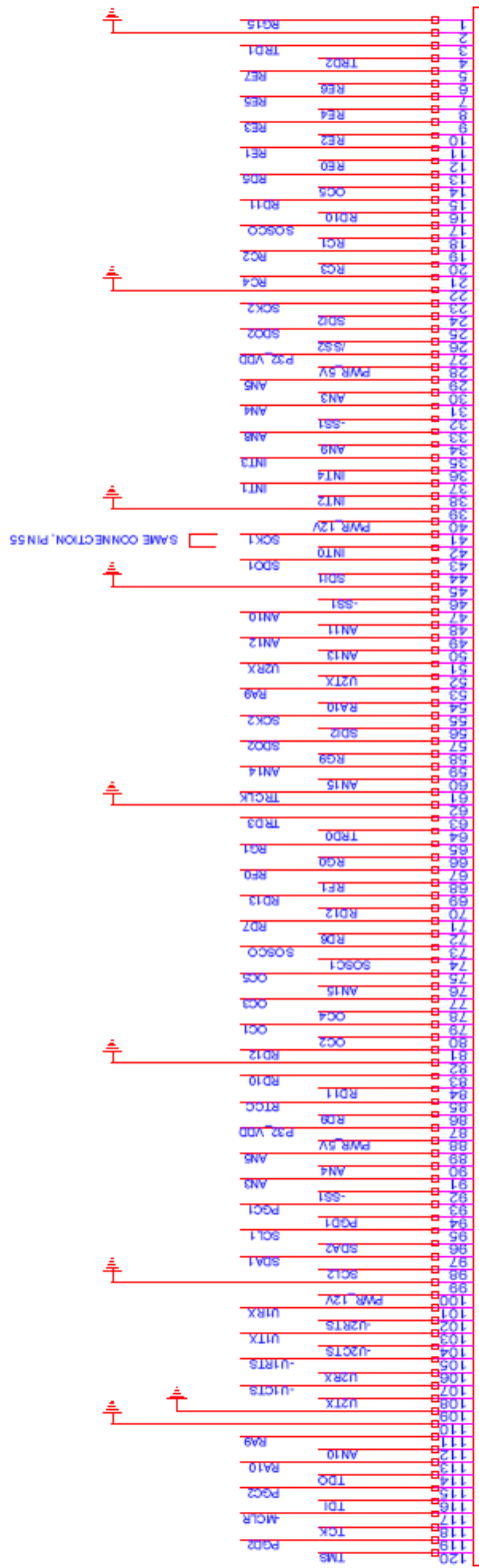


Figure 19 Schematic page 13



J141
PIC32-expansion

1 of 2 connector pairs to interface to PIC32 expansion board

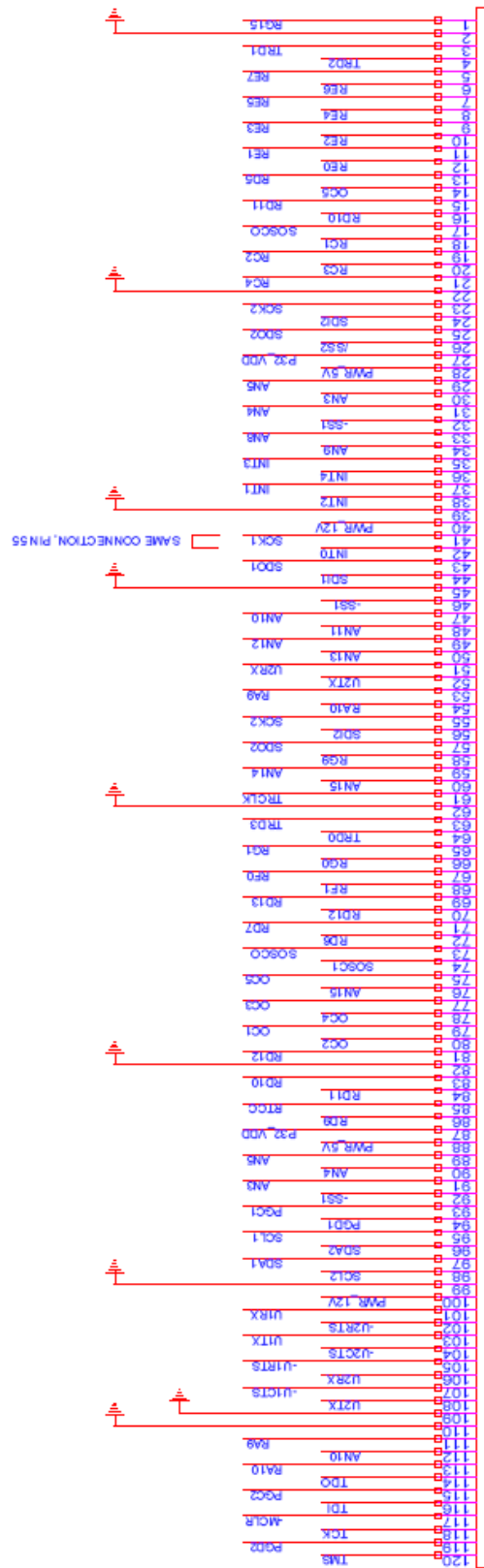
NOTE:
This page is used to map the BMS project net names to the PIC32 expansion board's two 60 pin connector headers. The net names listed here are the name given to the net as used in this design.

The two 60 pin connectors are combined as one 120 pin connector for this purpose. J10 are pins 1-60, and J11 are pins 61-120.

PIC32 Design Challenge

Project: PIC'n The Beehive
 Title: J10 & J11 Net Mapping Interface
 Date: 01-16-2009 Drawn by: Carl
 Dwg: BMS-14 Rev B Sheet 14 of 16

Figure 20 Schematic page 14



J151
PIC32-expansion

2 of 2 connector pairs, use for additional module expansion stacking

NOTE:

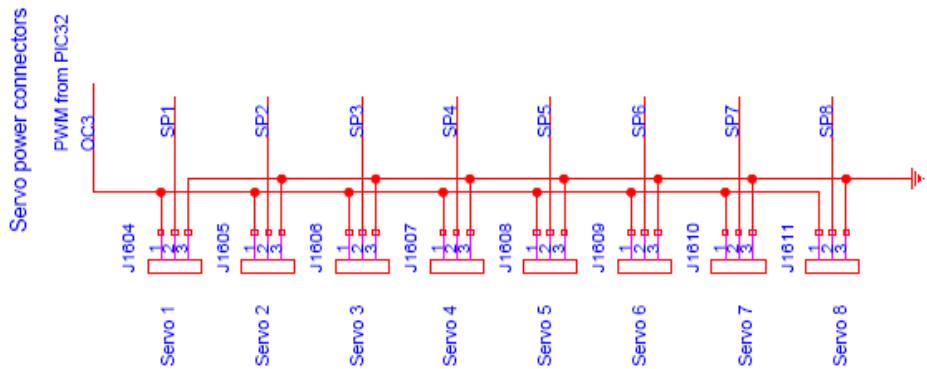
This page is used to map the BMS project net names to the PIC32 expansion board's two 60 pin connector headers. The net names listed here are the name given to the net as used in this design.

The two 60 pin connectors are combined as one 120 pin connector for this purpose. J10 are pins 1-60, and J11 are pins 61-120.

PIC32 Design Challenge

Project: PIC'n The Beehive
 Title: J10 & J11 Net Mapping Interface
 Date: 01-16-2009 Drawn by: Carl
 Dwg: BMS-15 Rev B Sheet 15 of 16

Figure 21 Schematic page 15



Spare foot prints for breadboard area

PIC32 Design Challenge

Project: PIC'n The Beehive
 Title: Proto Type Area, Servo Pwr Connectors
 Date: 01-16-2009 Drawn by: Carl
 Dwg: BMS-16 Rev B Sheet 16 of 16

Figure 22 Schematic page 16

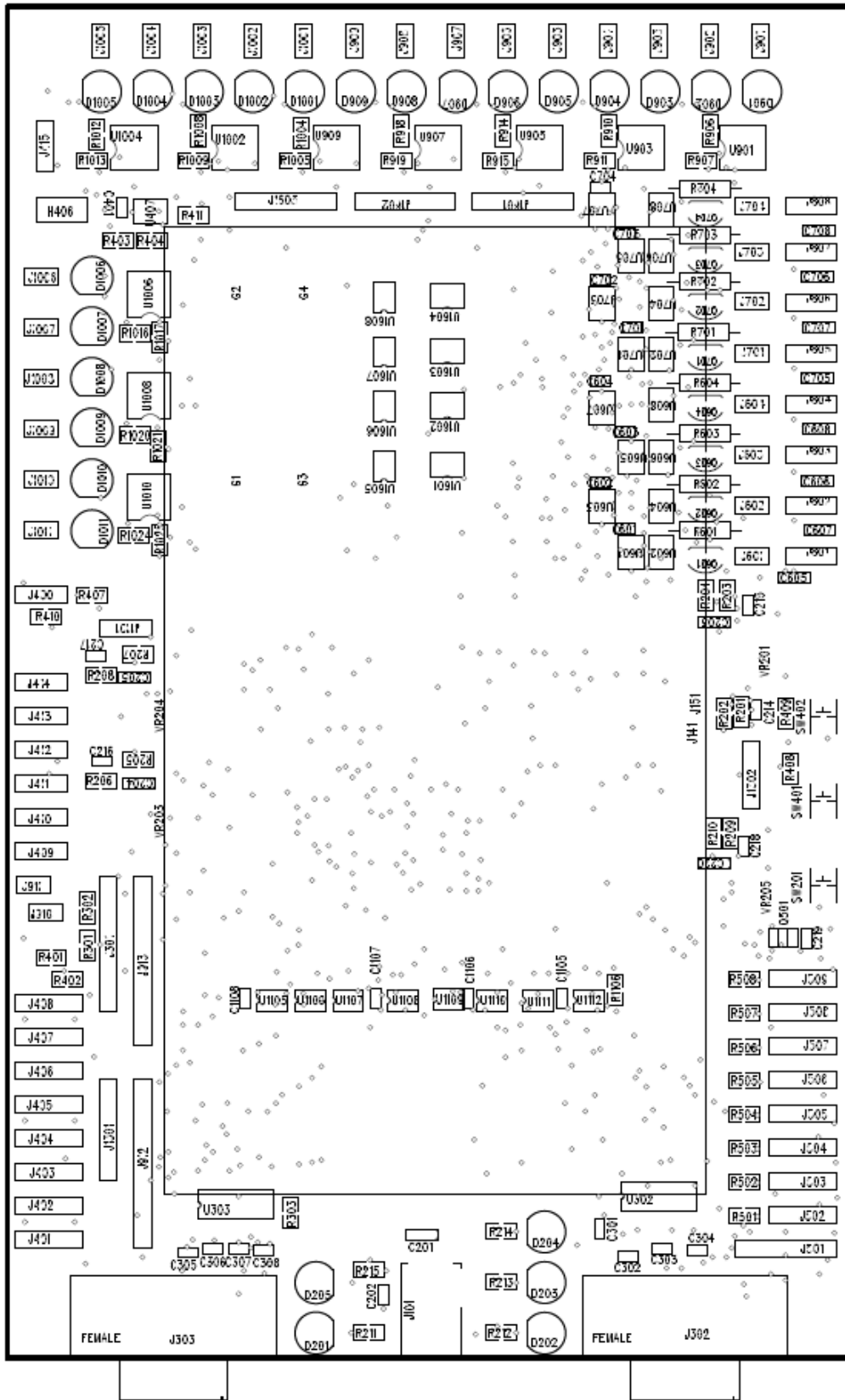


Figure 24 Prototype board topside view loading

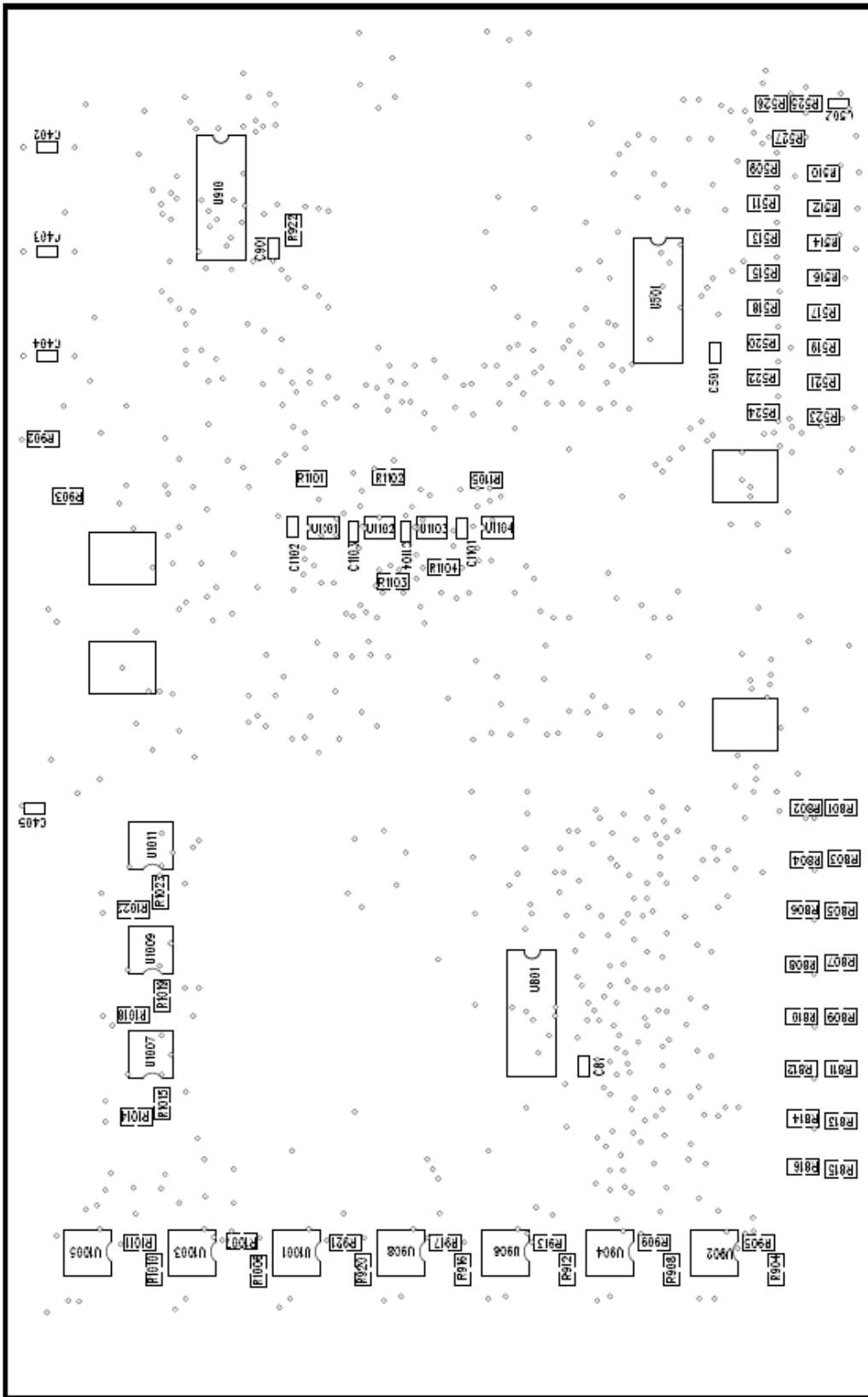


Figure 25 Prototype board bottom side view loading

Software Design

The PIC32 monitors the beehive's sensors, controls the hive's functions, and analyzes the data for presentation via the user request through its embedded web server. The software was patterned around the Microchip TCPIP Demo application for the PIC32. The workings of this software are based on the state machine approach to performing tasks. A measured time through one pass of the main loop is typically less than 1mS depending on the activity. For this application this response is well within the needs of all functions. If a need arises for faster response then all one needs to do is insert additional calls to that state machine to check that function in the main loop.

Software Block Diagram

PIC'n The Beehive

Software Block Diagram

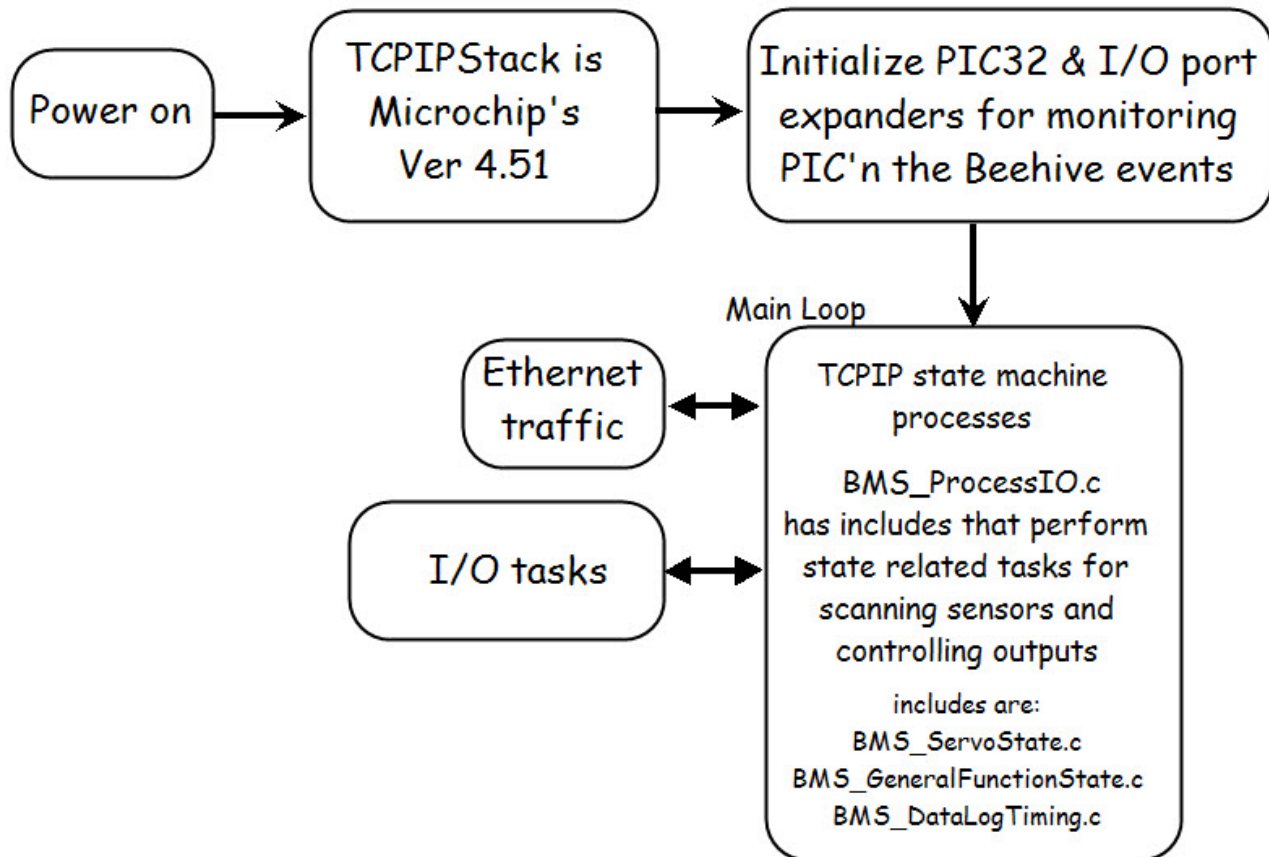


Figure 26 Software Block Diagram

Software Design – Embedded web server user interface

Index Web Page

This current version of software has all the basic functions working with the latest graph configuration. Screen shots of the embedded web server pages and descriptions are as follows:

HOME page: index.htm

Title: Hive Stats at a Glance

Description: Two graphs show the relative performance between two selected days for a function and for each of two functions. Days are presented in different colors side by side and functions are presented one above the other. The horizontal axis of each graph is based on the 24 hour clock format. The vertical axis is a relative percentage of the maximum value of data on the graph. Current hive date and time are displayed near the top left corner of the page, all pages display this time. Setting of the PIC32's real time clock/calendar (RTCC) is done on the Utilities web page, shown later.

Figure 27 below is showing a typical view of the home web page. The functions 'T-xxx' are temperatures recorded for various positions inside and outside of the hive. The graphs give a good visual image of what's going on in the hive, while the data analysis summary gives an accurate understanding of the actual temperature values recorded.

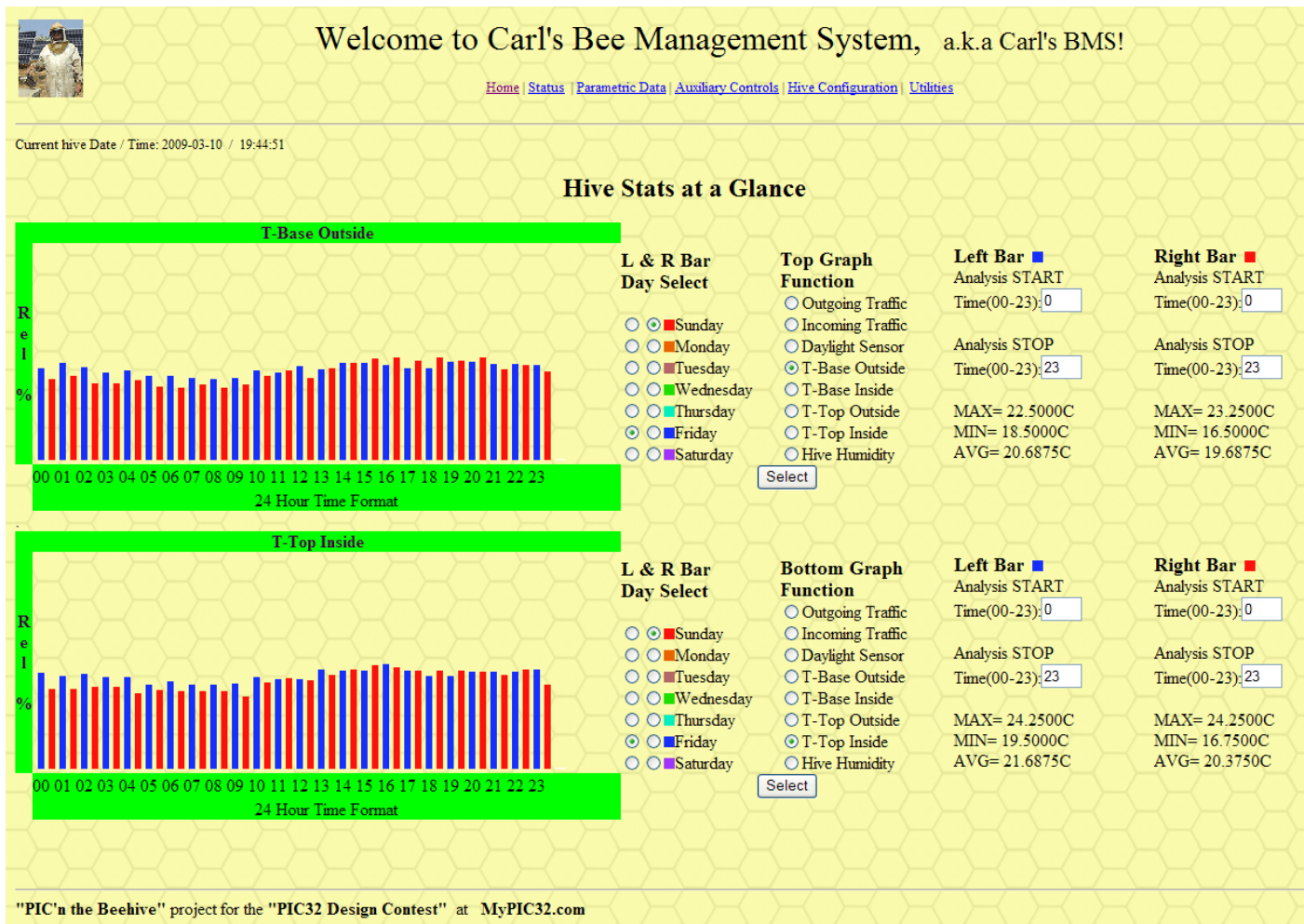


Figure 27 Home web page. Top function 'T-Base Outside', 2 different days selected. Bottom function 'T-Top Inside', same 2 days selected as in top graph.

Figure 28 shows another way to look at the graphs. This could be to plot the same day, while looking at different functions on the top and bottom.

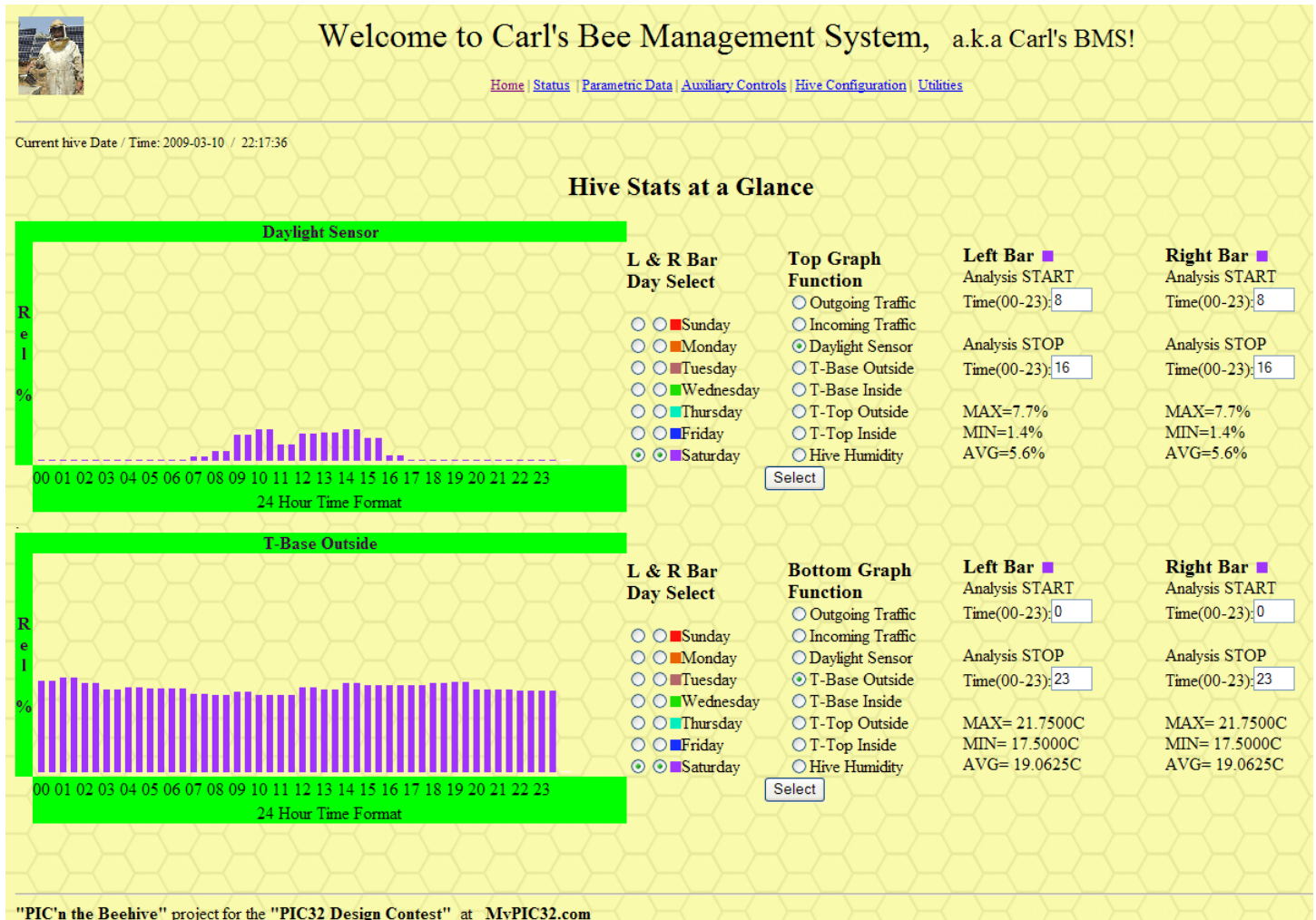


Figure 28 Home web page. Top function 'Daylight Sensor', 1 day selected. Bottom function 'T-Base Outside', same day selected as in the top graph. Hours for data analysis selected as 8 to 16 in top graph.

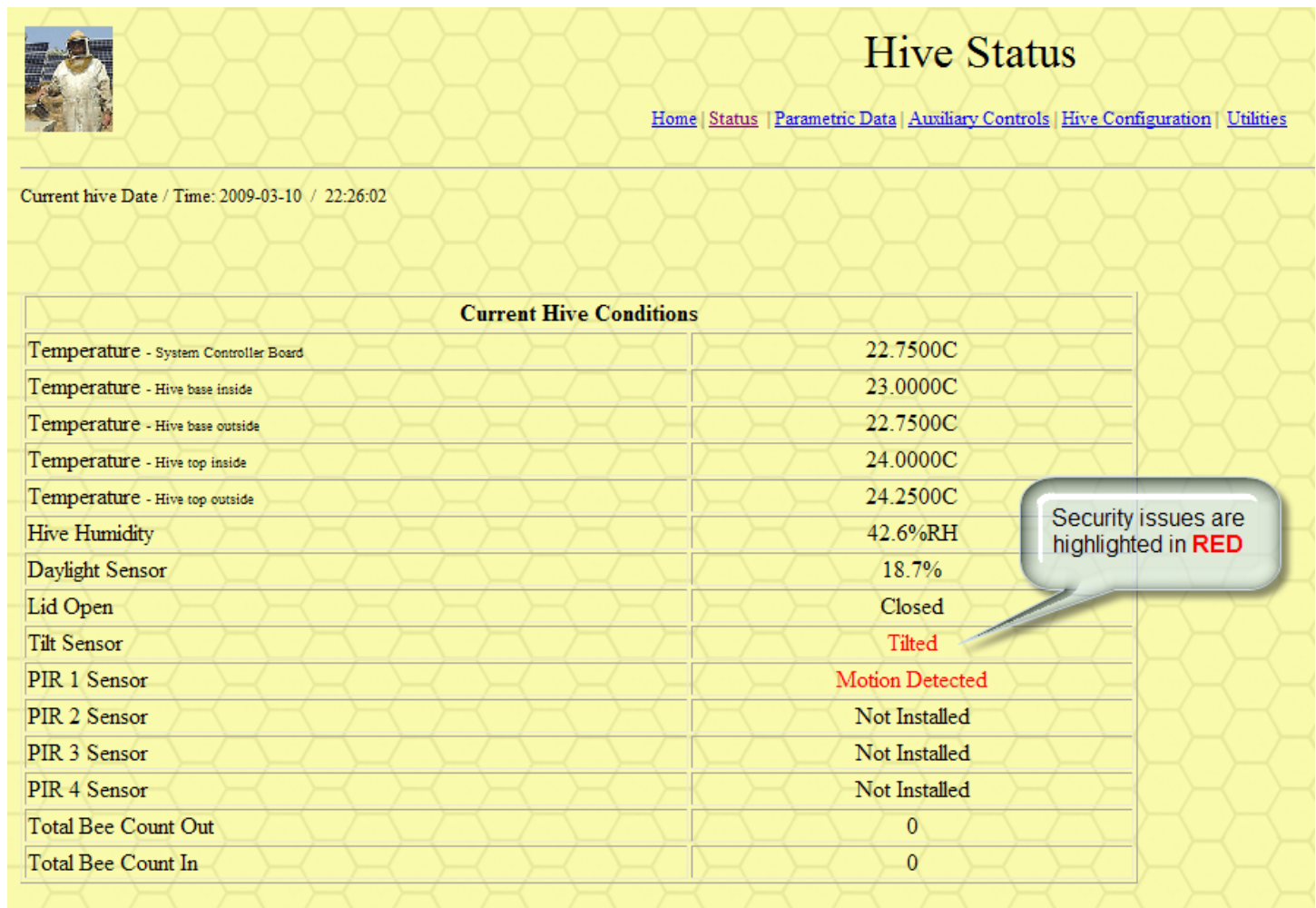
(Continued on next page)

Status Web Page

STATUS page: status.htm

Title: Hive Status

Description: Hive status or current hive conditions of the hive's 'basic functions' for environmental and security sensors are displayed. This displays the numeric values of the sensors. Security problems are highlighted in **RED** type face. Total bee traffic count for the period being data logged is shown. Configuration of the security sensors is performed under the 'Hive Configuration' menu described later.



Hive Status

[Home](#) | [Status](#) | [Parametric Data](#) | [Auxiliary Controls](#) | [Hive Configuration](#) | [Utilities](#)

Current hive Date / Time: 2009-03-10 / 22:26:02

Current Hive Conditions	
Temperature - System Controller Board	22.7500C
Temperature - Hive base inside	23.0000C
Temperature - Hive base outside	22.7500C
Temperature - Hive top inside	24.0000C
Temperature - Hive top outside	24.2500C
Hive Humidity	42.6%RH
Daylight Sensor	18.7%
Lid Open	Closed
Tilt Sensor	Tilted
PIR 1 Sensor	Motion Detected
PIR 2 Sensor	Not Installed
PIR 3 Sensor	Not Installed
PIR 4 Sensor	Not Installed
Total Bee Count Out	0
Total Bee Count In	0

Security issues are highlighted in **RED**

Figure 29 Hive Status web page.

(Continued on next page)

Parametric Data Web Page

PARAMETRIC DATA page: param.htm

Title: Hive Parametric Data

Description: This page is similar to the previous status page, but includes controller parameters to verify the system is operating in normal range. It does not include hive statistics such as bee traffic counts. Additional information will be added to show the performance of the systems sensors, a useful indication before a sensor fails completely.

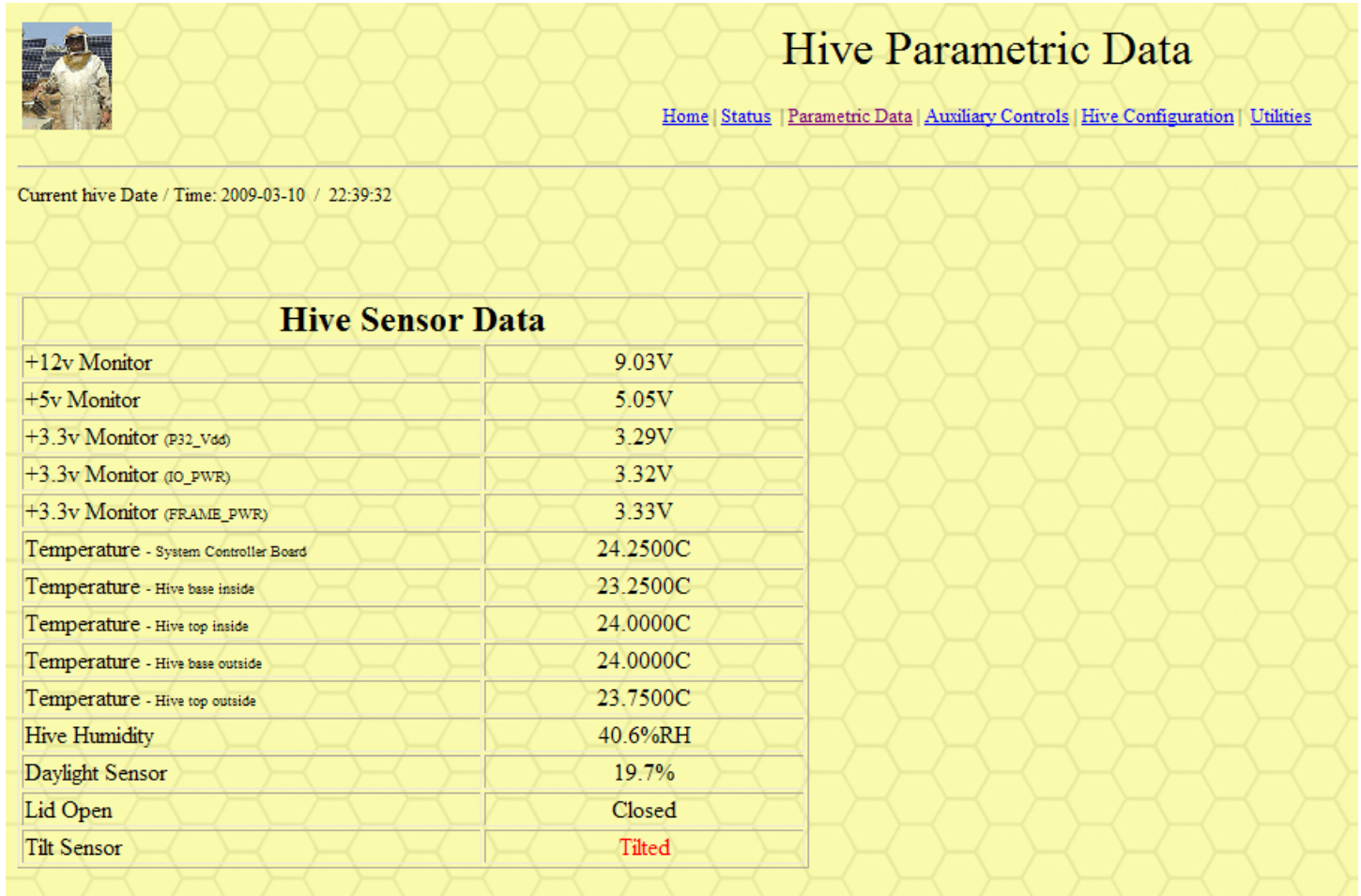


Figure 30 Hive Parametric Data web page.

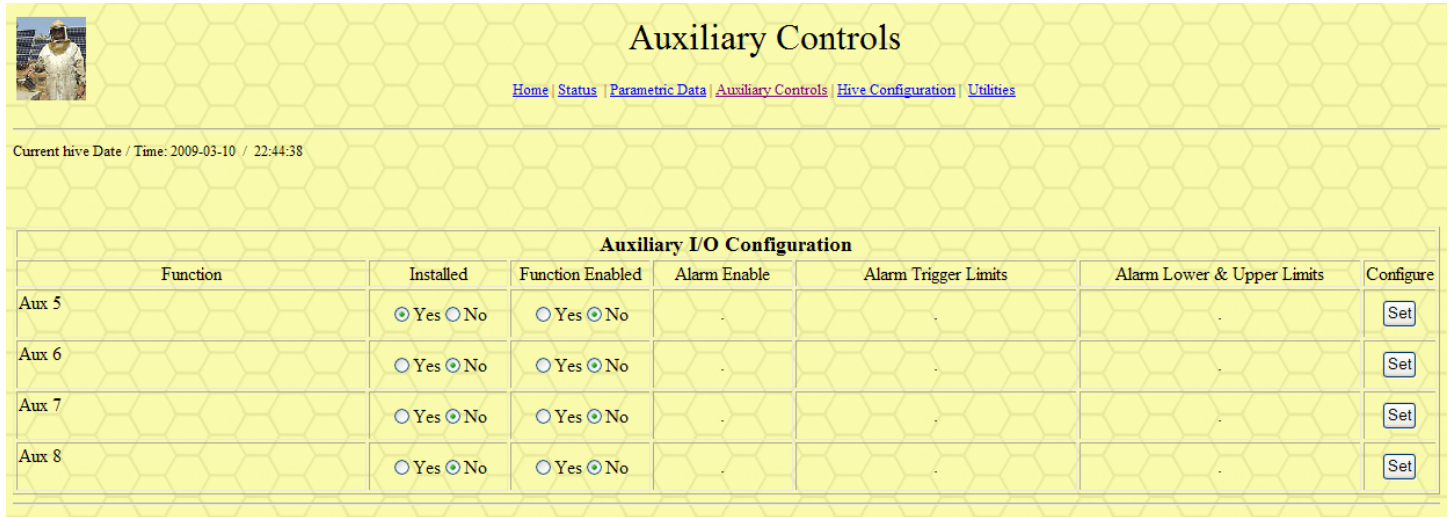
(Continued on next page)

Auxiliary Controls Web Page

AUXILIARY CONTROLS page: auxiliary.htm

Title: Auxiliary I/O Configuration

Description: A simple configuration for the extra outputs available to control anything. The intent here is to remotely select and control video cameras located in or around the hive. A cell phone could be auto dialed with one of these outputs as well. The design is flexible enough to allow the addition of more I/O port expanders to implement many more functions if needed. Such functions as an automated hive leveling mechanism could be implemented using both inputs and outputs from one expander.



Auxiliary I/O Configuration						
Function	Installed	Function Enabled	Alarm Enable	Alarm Trigger Limits	Alarm Lower & Upper Limits	Configure
Aux 5	<input checked="" type="radio"/> Yes <input type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	.	.	.	<input type="button" value="Set"/>
Aux 6	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	.	.	.	<input type="button" value="Set"/>
Aux 7	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	.	.	.	<input type="button" value="Set"/>
Aux 8	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	.	.	.	<input type="button" value="Set"/>

Figure 31 Auxiliary Controls web page.

These and all other control outputs have visual LED indication and optoisolators to buffer the system controller from the outside world.

(Continued on next page)

Hive Configuration- Security Web Page- Sub Menu

Hive Configuration page: security1.htm (1st of 5 pages in sub-menu)

Title: Security Configuration

Description: There are 5 pages that allow the user to configure the system software on an individual basis. It allows the user to input a number of parameters to describe a function as follows:

- 1) If function is installed
- 2) If the function is currently selected as 'enabled' or 'disabled'
- 3) If the function will trigger an alarm indication
- 4) If alarmed, then what alarm trigger limits will be used

The software within the program can be modified to accommodate any desired response. Some functions are interpreted as on/off for the yes/no inputs under 'Function Enabled'. An example of this the 'Electric Fence' would be remotely turned on or off by this selection. As with all output drive functions, an optical isolator provides the electrical switching for this function.

Function	Installed	Function Enabled	Email Alarm Repeat Wait Time	Alarm Trigger Limits	Alarm Lower & Upper Limits	Configure
Lid Open Sensor	<input checked="" type="radio"/> Yes <input type="radio"/> No	<input checked="" type="radio"/> Yes <input type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No 60 sec	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
Tilt Sensor	<input checked="" type="radio"/> Yes <input type="radio"/> No	<input checked="" type="radio"/> Yes <input type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No 60 sec	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
PIR 1 Sensor	<input checked="" type="radio"/> Yes <input type="radio"/> No	<input checked="" type="radio"/> Yes <input type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No 60 sec	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
PIR 2 Sensor	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No 60 sec	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
PIR 3 Sensor	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No 60 sec	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
PIR 4 Sensor	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No 60 sec	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
Electric Fence	<input checked="" type="radio"/> Yes <input type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No				<input type="button" value="Set"/>

Figure 32 Hive Configuration- Security web page.

(Continued on next page)

Hive Configuration- Environmental Web Page – Sub Menu

Hive Configuration page: enviro1.htm (2nd of 5 pages in sub-menu)

Title: Hive Environmental Configuration

Description: This page allows the user to configure the hive's basic environment features. Items such as the hive's entrance openings, heating, cooling are controlled. Additionally there are controls for producing sound, and vibration. The heating, cooling, sound and vibration are driven from PIC32 output pins that can be configured in the system software to be digital I/O or pulse width modulated (PWM). This allows for the required proportional control or wave form output. An interior light control is available if required for such options as inside the hive video. An exterior light control is also available if required for such a need as drawing the bees outside of the hive at night. A rain simulator control output is available if the need is to keep the bees inside the hive. Such an example would be the case if pesticide was being sprayed in the hive's area. Since bees don't fly in the rain, a sprinkler connected to an electric irrigation valve would provide the means to keep them inside the hive. While the lights are currently controlled from the PIC32's digital I/O pins, on the BMS_HardwareConfig.h page this could be reconfigured in software to use one of the PWM output pins to have a dimmer control on the lights if needed.



The screenshot shows a web page titled "Hive Configuration" with a yellow honeycomb background. It includes a navigation menu with links for Home, Status, Parametric Data, Auxiliary Controls, Hive Configuration, and Utilities. Below the menu, there is a timestamp: "Current hive Date / Time: 2009-03-10 / 23:05:56". A secondary navigation menu includes Security, Hive Environmental, Bee Traffic, Bee Scale, and Controller. The main content is a table titled "Hive Environmental Configuration" with columns for Function, Installed, Function Enabled, Alarm Enable, Alarm Trigger Limits, Alarm Lower & Upper Limits, and Configure. The table lists various functions such as Entrance Servo 1-4, Hive Cooling, Hive Heating, Sound Simulator, Vibration Simulator, Rain Simulator, Smoke Simulator, Hive Interior Light, and Hive Exterior Light. Each function has radio buttons for 'Yes' and 'No' under the 'Installed' and 'Function Enabled' columns. The 'Alarm Enable' column also has radio buttons for 'Yes' and 'No'. The 'Alarm Trigger Limits' column has radio buttons for 'Lower', 'Upper', 'Both', and 'Not Used'. The 'Alarm Lower & Upper Limits' column has input fields for 'Upper Limit' and 'Lower Limit'. The 'Configure' column has a 'Set' button for each row.

Function	Installed	Function Enabled	Alarm Enable	Alarm Trigger Limits	Alarm Lower & Upper Limits	Configure
Entrance Servo 1	<input checked="" type="radio"/> Yes <input type="radio"/> No	<input checked="" type="radio"/> Yes <input type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
Entrance Servo 2	<input checked="" type="radio"/> Yes <input type="radio"/> No	<input checked="" type="radio"/> Yes <input type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
Entrance Servo 3	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
Entrance Servo 4	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
Hive Cooling	<input checked="" type="radio"/> Yes <input type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
Hive Heating	<input checked="" type="radio"/> Yes <input type="radio"/> No	<input checked="" type="radio"/> Yes <input type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
Sound Simulator	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No				<input type="button" value="Set"/>
Vibration Simulator	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No				<input type="button" value="Set"/>
Rain Simulator	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No				<input type="button" value="Set"/>
Smoke Simulator	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No				<input type="button" value="Set"/>
Hive Interior Light	<input checked="" type="radio"/> Yes <input type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
Hive Exterior Light	<input checked="" type="radio"/> Yes <input type="radio"/> No	<input checked="" type="radio"/> Yes <input type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>

Figure 33 Hive Configuration- Environmental Controls web page.

Currently all of the above 'Installed' and 'Function Enabled' inputs allow for the LED indicator and optical isolator outputs to be driven.

Hive Configuration- Bee Traffic Web Page- Sub Menu

Hive Configuration page: traffic1.htm (3rd of 5 pages in sub-menu)

Title: Bee Traffic Configuration

Description: This page configures the software to recognize the installed and enabled number of bee traffic counters. The required software and hardware I/O is already on the prototype board for eight modules. It would just need to have the exterior hardware plugged in and be programmed to enable it on this web page.

Bee Traffic Configuration						
Function	Installed	Function Enabled	Alarm Enable	Alarm Trigger Limits	Alarm Lower & Upper Limits	Configure
Bee Traffic Counter 1	<input checked="" type="radio"/> Yes <input type="radio"/> No	<input checked="" type="radio"/> Yes <input type="radio"/> No	<input checked="" type="radio"/> Yes <input type="radio"/> No	<input type="radio"/> Lower <input type="radio"/> Upper <input checked="" type="radio"/> Both <input type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
Bee Traffic Counter 2	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
Bee Traffic Counter 3	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
Bee Traffic Counter 4	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
Bee Traffic Counter 5	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
Bee Traffic Counter 6	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
Bee Traffic Counter 7	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
Bee Traffic Counter 8	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>

Figure 34 Hive Configuration- Bee Traffic Configuration web page.

(Continued on next page)

Hive Configuration- Bee Scale Web Page- Sub Menu

Hive Configuration page: scale1.htm (4th of 5 pages in sub-menu)

Title: Bee Scale Configuration

Description: This page configures the system software for the number of installed bee weighing scales. The hardware for eight scales is included on the prototype board. The technique used is a precision method of controlling the current through a coil. In this application the PIC32 drives a digital to analog converter (DAC) via an I2C bus, there is one DAC for each of the eight scales on the prototype board. The voltage output from the DAC is used to drive a voltage to current converter. This current is then driven through a coil that pulls a metal strip up slightly. An optical IR pair detects the mechanical limit at which to stop applying more current. At this point the exact amount of current is known and correlates to a calibrated amount of weight. This bee scale function is an advance feature of the project. As such it is not currently being implemented in this release of the software. I wanted to get all of the underlying basic features functionally implemented, before spending more time fine tuning the software in this advanced feature. It would be possible to configure the use of upper and lower limits in the system software to give an indication of 'out-of-normal' values, this being used to log a possible fault condition with that scale.

Bee Scale Configuration						
Function	Installed	Function Enabled	Alarm Enable	Alarm Trigger Limits	Alarm Lower & Upper Limits	Configure
Bee scale 1	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
Bee scale 2	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
Bee scale 3	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
Bee scale 4	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
Bee scale 5	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
Bee scale 6	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
Bee scale 7	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
Bee scale 8	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>

Figure 35 Hive Configuration- Bee Scale Configuration web page.

(Continued on next page)

Hive Configuration- Controller Web Page- Sub Menu

Hive Configuration page: cont1.htm (5th of 5 pages in sub-menu)

Title: Hive Controller Configuration

Description: This is a minor page to describe the functions that are controller related. The intent is to be able to program the system software to set an alarm condition based on the parameter selected. The alarm will cause a system status light to be lit. Currently the green status LED is used to indicate the 'heart beat' for the TCP/IP stack. If it blinks, the system software is up and running through the state machines. The yellow LED status is used to indicate a security alarm or status alert email is currently being transmitted. The red LED status is currently being used to indicate that a data log email is being transmitted. In Figure 9 below, the +12 volt supply has been configured to send an email alarm if the voltage drops below 7.5 volts or goes above 15 volts. For illustration purposes the repeat time out value has been set a small value of 60 seconds. A more realistic value might be on the order of 1 hour or more, so as not to flood the email box.

Hive Controller Configuration						
Function	Installed	Function Enabled	Alarm Enable	Alarm Trigger Limits	Alarm Lower & Upper Limits	Configure
+12v power supply monitor	<input checked="" type="radio"/> Yes <input type="radio"/> No	<input checked="" type="radio"/> Yes <input type="radio"/> No	<input checked="" type="radio"/> Yes <input type="radio"/> No 60 sec	<input type="radio"/> Lower <input type="radio"/> Upper <input checked="" type="radio"/> Both <input type="radio"/> Not Used	Upper Limit: 14500 Lower Limit: 7500 Enter as mV, i.e. 9.15v ==> 9150	<input type="button" value="Set"/>
Hive Status RED Light	<input checked="" type="radio"/> Yes <input type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
Hive Status YELLOW Light	<input checked="" type="radio"/> Yes <input type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>
Hive Status GREEN Light	<input checked="" type="radio"/> Yes <input type="radio"/> No	<input checked="" type="radio"/> Yes <input type="radio"/> No	<input type="radio"/> Yes <input checked="" type="radio"/> No	<input type="radio"/> Lower <input type="radio"/> Upper <input type="radio"/> Both <input checked="" type="radio"/> Not Used	Upper Limit: <input type="text"/> Lower Limit: <input type="text"/>	<input type="button" value="Set"/>

Figure 36 Hive Configuration- Hive Controller Configuration web page.

(Continued on next page)

Utilities Web Page

System Utilities page: util.htm

Title: Clock Setup, Warm Restart Setup, Data Log Memory (Save/Restore), Data Logging

Description: Ref Figure 10 on next page, there are several system related utilities to help maintain it. The PIC32's RTCC is set on this page. Time is entered based on a 24-hour clock format.

Email alerts and data logging are configured here as well. It is possible to have the system ignore these alarm conditions by selecting 'DISABLE emails'. A good example of using this would be when working on the hive to avoid sending emails that don't really need to be sent. If enabled, the alerts are set immediately when the condition occurs. The data logging emails are sent every night at midnight for that day's data collected.

In the case when the beekeeper needs to have some current information about hive conditions, an immediate data log email could be requested. The beekeeper could select any or all the days desired from the last 7 days.

Also included are two functions to save the PIC32's RAM to 24LC512 EEPROM's. This allows for a 'warm restart' without having to enter all the configuration data again. The data log memory can also be selected to save its current status to EEPROM as well. This would be done just prior to powering the system down. Then on restart this page can be selected and the 'RESTORE Data Log Memory' executed to restore the system to the same state it was in when shut down. Currently with all of the functions described on the previous pages, about six weeks of data can be stored in one 24LC512 memory device. The prototype board currently allows for up to 8 of these devices installed. One device is reserved for controller functions and calibration data. With seven devices possible, there can be something over 42 weeks' of data that can be stored locally on the system board. Currently the software is configured to keep a rolling 7 days of data in both RAM and EEPROM. Additional options are being added to allow the maximum use of storage installed.

The ability to clear the 'Repeat Wait Time' counters was provided as a means to test program flow and alarm conditions. This clears the repeat email timers so that the next email is set immediately if the error condition continues to exist.

The Real Time Clock Calendar calibration value can also be set on this page. This is used to fine tune or adjust the RTCC oscillator. Positive values will slow the clock down, and negative values will speed the clock up.

(See web page image on next page)

System Utilities



[Home](#) | [Status](#) | [Parametric Data](#) | [Auxiliary Controls](#) | [Hive Configuration](#) | [Utilities](#)

Current hive Date / Time: 2009-03-10 / 23:53:47

Clock Setup						
Current Time =						
year(yyyy)	month(1-12)	mday(1-31)	wday(0-6)	hour(00-23)	min(0-59)	sec(0-59)
<input type="text" value="2009"/>	<input type="text" value="03"/>	<input type="text" value="10"/>	<input type="text" value="3"/>	<input type="text" value="23"/>	<input type="text" value="53"/>	<input type="text" value="47"/>
<input type="button" value="Set"/>						

Email Alerts, Status and Data Logging (Enable/Disable)								
Email Alerts & Status:	<input type="radio"/> ENABLE emails	<input checked="" type="radio"/> DISABLE emails						
Email Data Logging:	<input type="radio"/> ENABLE emails	<input checked="" type="radio"/> DISABLE emails						
Email Log Data Now:	<input type="checkbox"/> Sun	<input type="checkbox"/> Mon	<input type="checkbox"/> Tue	<input type="checkbox"/> Wed	<input type="checkbox"/> Thu	<input type="checkbox"/> Fri	<input type="checkbox"/> Sat	<input type="button" value="Execute"/>

Warm Restart Setup		
<input type="radio"/> SAVE CURRENT Configuration	<input type="radio"/> RESTORE System Configuration	<input type="button" value="Execute"/>

Data Log Memory (Save/Restore)		
<input type="radio"/> SAVE CURRENT Data Log Memory	<input type="radio"/> RESTORE Data Log Memory	<input type="button" value="Execute"/>

Repeat Wait Time Reset		
<input type="radio"/> RESET all Alarm Wait Times	<input type="radio"/> Ignore selection	<input type="button" value="Execute"/>

RTCC Calibration Value	
<input type="text" value="0"/> Range: -512 to +511	<input type="button" value="Set"/>

Figure 37 System Utilities web page.

(Continued on next page)

Email Data Log Output

The following is a sample of the email that is sent either when the user requests it on demand or as an automated request at the end of each day. The data is easily imported into a spread sheet for further analysis if needed.

Current Folder: INBOX

[Compose](#) [Addresses](#) [Folders](#) [Options](#) [Search](#) [Help](#) [Fetch](#) [Calendar](#)

[Message List](#) | [Delete](#) [Previous](#) | [Next](#) [Forward](#) | [Forwa](#)

Subject: HIVE-1_DATA_LOG
From: "HIVE-1"
To: hive1@microchip.com
Priority: Normal
Options: [View Full Header](#) | [View Printable Version](#) | [View Message details](#)

Out_Traffic data log for: Fri | Sent 2009-03-14 / 11:48:57,
2, 0,

In_Traffic data log for: Fri | Sent 2009-03-14 / 11:48:57,
2, 0,

Daylight data log for: Fri | Sent 2009-03-14 / 11:48:58,
212, 217, 59, 14, 0, 0, 0, 7, 15, 86, 127, 62, 47, 203, 64, 63, 23, 4, 0, 0, 0, 71,
76, 74,

T-Base_Outside data log for: Fri | Sent 2009-03-14 / 11:48:58,
5392, 5376, 5392, 5120, 5136, 5008, 4944, 4880, 4960, 4864, 5072, 5024, 5312, 5360,
5536, 5520, 5552, 5472, 5584, 5472, 5520, 5600, 5488, 5632,

T-Base_Inside data log for: Fri | Sent 2009-03-14 / 11:48:58,
5536, 5520, 5296, 5328, 5392, 5040, 5136, 5136, 5072, 5008, 5168, 5328, 5440, 5472,
5632, 5840, 5920, 5776, 5664, 5696, 5728, 5744, 5744, 5584,

T-Top_Outside data log for: Fri | Sent 2009-03-14 / 11:48:59,
5632, 5616, 5376, 5376, 5216, 5040, 5056, 5168, 4992, 5072, 5136, 5392, 5328, 5360,
5680, 5728, 5856, 5776, 5584, 5488, 5824, 5632, 5696, 5792,

T-Top_Inside data log for: Fri | Sent 2009-03-14 / 11:49:00,
5616, 5568, 5552, 5408, 5376, 5200, 5072, 4992, 5136, 5296, 5360, 5344, 5296, 5664,
5776, 5808, 6080, 6016, 5680, 5760, 5872, 5696, 5872, 5824,

Hive_Humidity data log for: Fri | Sent 2009-03-14 / 11:49:00,
372, 373, 371, 366, 366, 364, 364, 364, 362, 361, 370, 367, 367, 372, 372, 373, 372,
369, 366, 367, 367, 370, 374, 373,

[Download this as a file](#)

Figure 38 Email data log sample output

PIC32 System Controller Software Overview

The starting point for this program was the Microchip TCPIP stack demo 'TCPIP Demo App'. Initially I played around with the demo and hardware to learn the basics of a web server, since this was my first try at designing a web server. Also I had never written a web page application in HTML. The short of it was I was on a steep learning curve.

The PIC'n the Beehive started with writing small code segments to understand the needs of all the hardware devices. This included routines for the UART, I2C, SPI and I/O expansion ports, with each device having their own special quirk. As development proceeded I tried to standardize on a procedure call to the hardware item's lower level routines. I'm sure there is still some clean up to do to make this consistent throughout the program. The application development was done by modifying the 'TCPIP Demo App', adding necessary code modules where needed. Because of this you will see extra code left in, that at this point in time it does not need to be there. However, I did not remove the code because as development of advance features proceeds some of these 'extra' modules will be used. Better to leave them in now than try to restore them later.

Software Modules

MainDemo.c

The 'TCPIP Demo App' module as supplied from Microchip already initializes and starts the TCPIP stack. Additional code was added to initialize the required hardware modules for the BMS. It is in this section that the MCP23017 16-bit I/O expanders are reset. A side note here that initially I was thinking I could get away without a hardware reset on these devices, but during development I was experiencing 'lock' ups of these expanders, even though I had written a routine to determine which 'bank' these devices were currently configured in before trying to configure them. The simple and reliable solution was to use a PIC32 I/O pin (MCP23017_Reset_io) for the hardware reset to the three MCP23017's. The routine (InitializeBeeTrafficCounters for the MCP23017 used for the bee traffic counter first opens the I2C port and then initializes the expander. Due to a significant library bug that makes the I2C2 port non-functional in the PIC32 some work around code was added that is not needed for the I2C1 port. So any device that uses the second I2C port needs to have the work around code. The nice thing is that when the bug gets fixed, the code will still work as is. A hardware feature was designed in for self testing the bee traffic counters. This works by reading the IR detectors on U501 with and without the IR emitters powered (BTC SW_io) on. This insures that both the emitter and detector function and that the bee tunnel is clear. This self test code was used at initial board bring up, but is not in the current revision. I will likely add it back in the final version.

The next thing that occurs during the initialization sequence is that a jumper to ground, if present, on AUX4 is read as a low level. The purpose of this jumper is to determine if we are to generate test data for the demo mode. If present, the TestModeBMS flag is set TRUE. I used this extensively for test and debug during program development. While it started out as 'test' maybe it should be more appropriately called 'demo' now.

Next a couple of variables are set for the web page graph display. This could be about anywhere, but it is here.

The next clever hardware thing was to drive eight servos from just one PIC32 PWM pin. I used a MCP23017 expander as a means to enable the power to the entrance servos one at a time. Since in this application there is no need for speed in operating this feature, this method can be used. Another important fact is that we can cut power consumption with the servos off, and they only turn on if required to reposition. Later on in the software, each pass through the servo state machine, the status is check for each servo for possible change, and it only changes one at a time. Also important in this application is that once the servo positions the entrance opening, power to the servo does not need to be maintained to hold the position. If it did, with each servo drawing a maximum of 200mA this would be a chunk of current (1.6A) to deal with. *(One hardware glitch was that the MCP23017s did not function as I was expecting when pulled up to 5 volts as an open collector, too much leakage. So I ended up using a NPN and PNP combination to implement high side drive to the servos. This part of circuit had not yet been finalized when I was laying the PCB out, so that's what the breadboard area*

was used for. This same circuit was used for the IR emitter on/off mentioned above in the bee traffic tunnels. In that case I was originally going to use a p-channel FET because of the low power requirement, but just decided to go with the same circuit used here. In hind sight I could change this to just one NPN for a low side switch on the IR emitters if I redo the PCB board layout. Current software would work with either configuration.) Now with the servo powered and configured the ServoBusyFlag is set FALSE.

Just for good measure several I/O pins used for status and control outputs are set low. Later they will be set according to the 'warm start' data that was previously saved in EEPROM. The PIC32's real time clock calendar (RTCC) is set with a default time. This time is later set by the user in the 'Utility' web page.

The include file 'InitDataLogInterval.c' is next executed that sets up some data log flags. Currently these values are set to data log at 0, 15, 30 and 45 minutes on the RTCC.

Currently only the servos are using the PIC32's I/O pin as a PWM. The other uses for heating, cooling, sound, vibration are currently set as standard on/off mode. They are however connected to and configurable as PWM outputs if needed.

Next the routine 'InitializeBarGraphArray' is run to clear some data space as good measure and check the 'TestModeBMS' flag for simulated data generation.

The 'INTEnableSystemMultiVectoredInt' is executed to configure the interrupts. Some PIC18CXX code, which was left in from the original Microchip demo software is skipped over. An include file 'BMS_ADC_scan_setup.c' is executed to setup the ADC to scan the channels used in the BMS application. Next is the UART setup which completes the 'InitializeBoard' routine. Following that, the 'EEPROM_to_RAM_Copy' routine is called to set the software to the last 'saved' configuration so that we don't have to re-enter all of the configuration information for our hive's installation. This is done by default on every power up. It can be executed on demand by the user if so desired (Utility web page). This might be required if a mistake was made while entering new configuration information. Currently one 24LC512 is reserved for system configuration information, although less than 1k byte is being used. The second 24LC512 is used for data logging. Up to eight of these devices are possible to install in the current configuration, but only two are presently installed.

The main routine continues with some additional things like setting some details up for the TCPIP stack and checking if some buttons are pressed for board setup. This part of the 'TCPIP Demo App' TCPIP stack code has not been modified.

The next part of the code executes in a while(1) loop continuously. Each pass through a timer is check to determine if the "I'm alive" green LED should be toggled. At this point I have added checks for the two push buttons being pressed. If SW1 is pressed it will increment the bee incoming traffic counter (BeeInCounter). If SW2 is pressed it will increment the bee outgoing traffic counter (BeeOutCounter).

Another include file 'BMS_ProcessIO.c' is executed. The 'CheckBeeCounters' routine scans the IR detector pairs in the bee traffic tunnels to see if a bee is present and which direction it is going, and increments the appropriate bee traffic counter. Additional include files sequence through state machine routines for the various functions to scan. The 'BMS_ServoState.c' code actually scans for 8 servos, even though the current hardware is wired for two. The web server page that supports this function only shows 4 just to get the point across. I didn't want to clutter up the screen at this time. The 'BMS_GeneralFunctionState.c' scans through all of the user selectable functions to see if any change has been requested through the web server state machine. The last include file 'BMS_DataLogTiming.c' checks to see if data logging is scheduled to be done. At this point current time is retrieved from the PIC32 RTCC. A check is made to see if we have the 'TestModeBMS' flag set. If set, this will change the rate at which we data log. The point of the 'TestModeBMS' option is to collect and display useful data at a faster rate for testing, debug and demo of the application. The change involves using seconds for minutes and hours for days substitution. Since we need to count only up to 23 hours on the chart, the minutes are set to reset back to 0 at 24, instead of counting on to 59 minutes. The days only need to count 0-6, so at 7 hours we reset it back to 0. This method gives us some variability in the actual sensor data we are measuring quicker. The 'DataLogHive' routine is called to scan the sensors and record the data to the memory array 'DataLogDayArray[LogDay][DayHour][Reading].cFunc'. The array has the following dimensions `DataLogDayArray[7][24][Reading].function_record`. So 1 week of data is stored in the PIC32's RAM. This makes it fast

when scanning data for the web server graphs. This concludes the main loop, which repeats over and over checking all state machines.

Web Server code

There are two main routines that were added for the BMS web server application. The 'CustomApp.c' code has an include file added at the end that is 'BMS_CustomApp.c'. This contains all of the HTTPPrint files for dynamic variable callback functions being requested from the web pages. *(When I started this application I didn't have everything figured out quite yet. But as the development went on I got cute about using arrays to pass the dynamic data back to the browser. When doing the graphs this worked out really well. At one point I stopped reducing this down, because no one would ever figure out what was going on to learn from it.)*

The other addition is in the 'HTTP_IO_RESULT HTTPExecuteGet' function. I have added a number of include files that relate to each web page to evaluate what data is being configured or requested and processing that request. The 'BMS_HTTPApp_index.c' include file is the one that configures the graphs for display. To display the vertical bars on the web page I used the HTML ability of sizing a gif image. The basic image which is of the color I select is 2 pixels wide by 5 pixels high. When the image is sent to the browser it is scaled as necessary for the calculated height of the bar. The actual data going to the web browser is done in the HTTPPrint routines above.

A HTML code line sent to the browser might have

```
<IMG SRC="red2x5.gif" HEIGHT=76 WIDTH="6">
```

The gif image file is named red2x5.gif; the actual height the bar would be displayed at is 76 with a width of 6.

The actual code line that you see in the source code being compiled would be something like

```
<IMG SRC=~sdoday(0)~ HEIGHT=~sdobr(0)~ WIDTH="6">
```

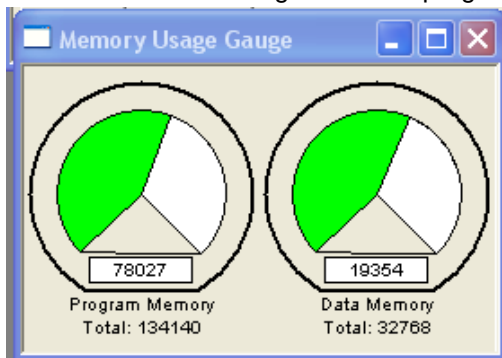
The ~sdoday(0)~ is a dynamic variable callback to get the gif file name 'red2x5.gif' and the ~sdobr(0)~ gets the number 76. The number inside the () is the item we wish to get the data for. It might be a day of the week of the bar number (hour) that we are graphing.

Web Page HTML code

There are eleven web pages that can be served up. Not having any previous experience writing HTML code I just dug my heels in and started. I used just a HTML text editor for writing the code. If you can read HTML code then I think the pages are self explanatory.

The memory array called FunctionModule[FunctionModuleSize] contains records of information about the hive and the sensors installed. This array configures the program 'on-the-fly' to do what is programmed. On the Utility.htm page this information can be saved to EEPROM. It is used for the 'warm restart' when powering the system up.

At the time of this writing the PIC32 program memory gauge was noted as follows:



This would indicate that there is a lot of room yet for the advanced features I want to include in this project.

Prototype Hardware Photos

Main Board

The following are photos of some of the prototype hardware developed in this project.

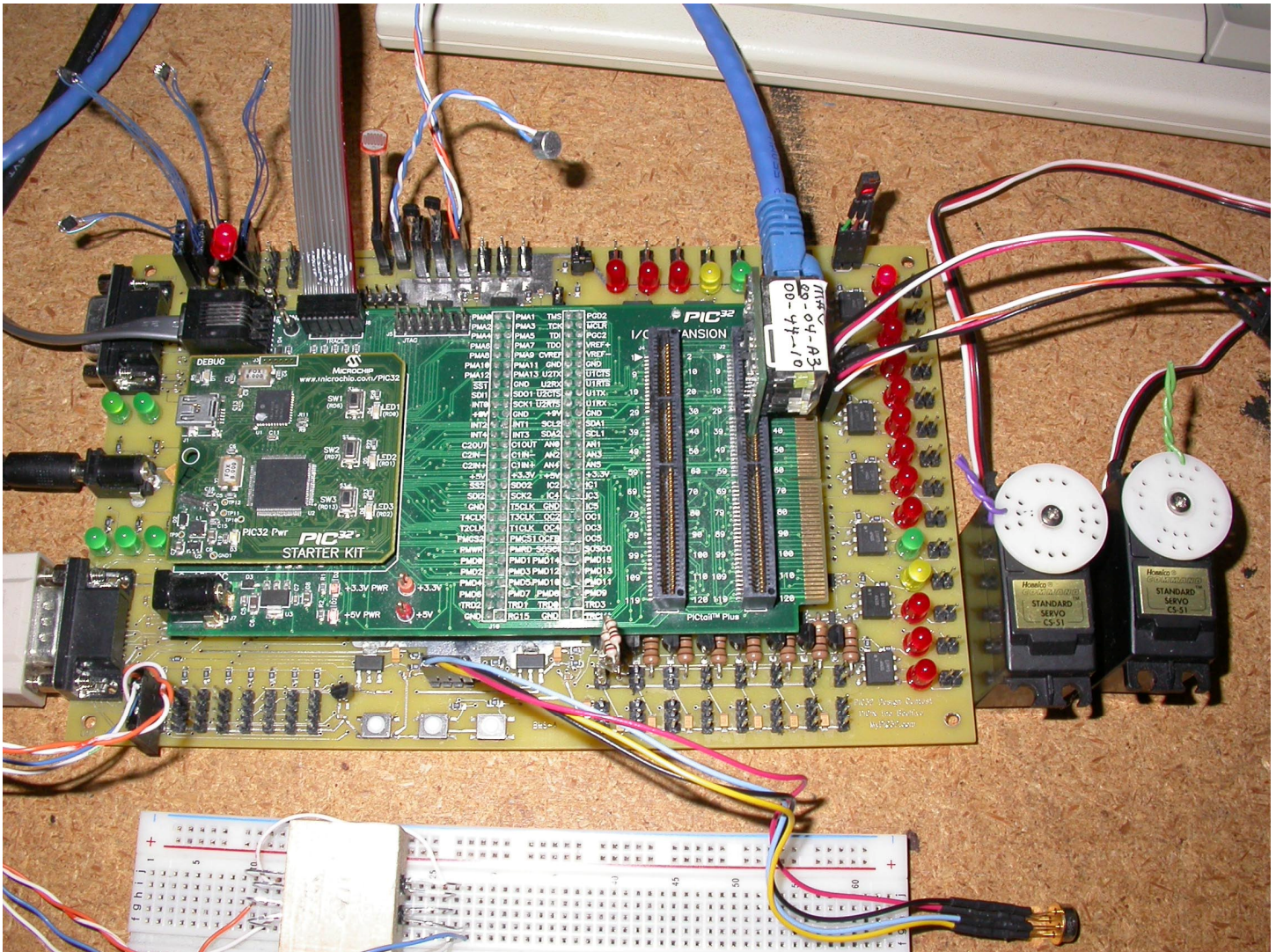


Figure 39 Photo of prototype board and early bee tunnel in foreground. Non-contact IR bee temperature sensor can be observed on the end of a 4-wire cable lying across the bread board.

(Continued on next page)

Bee Tunnels

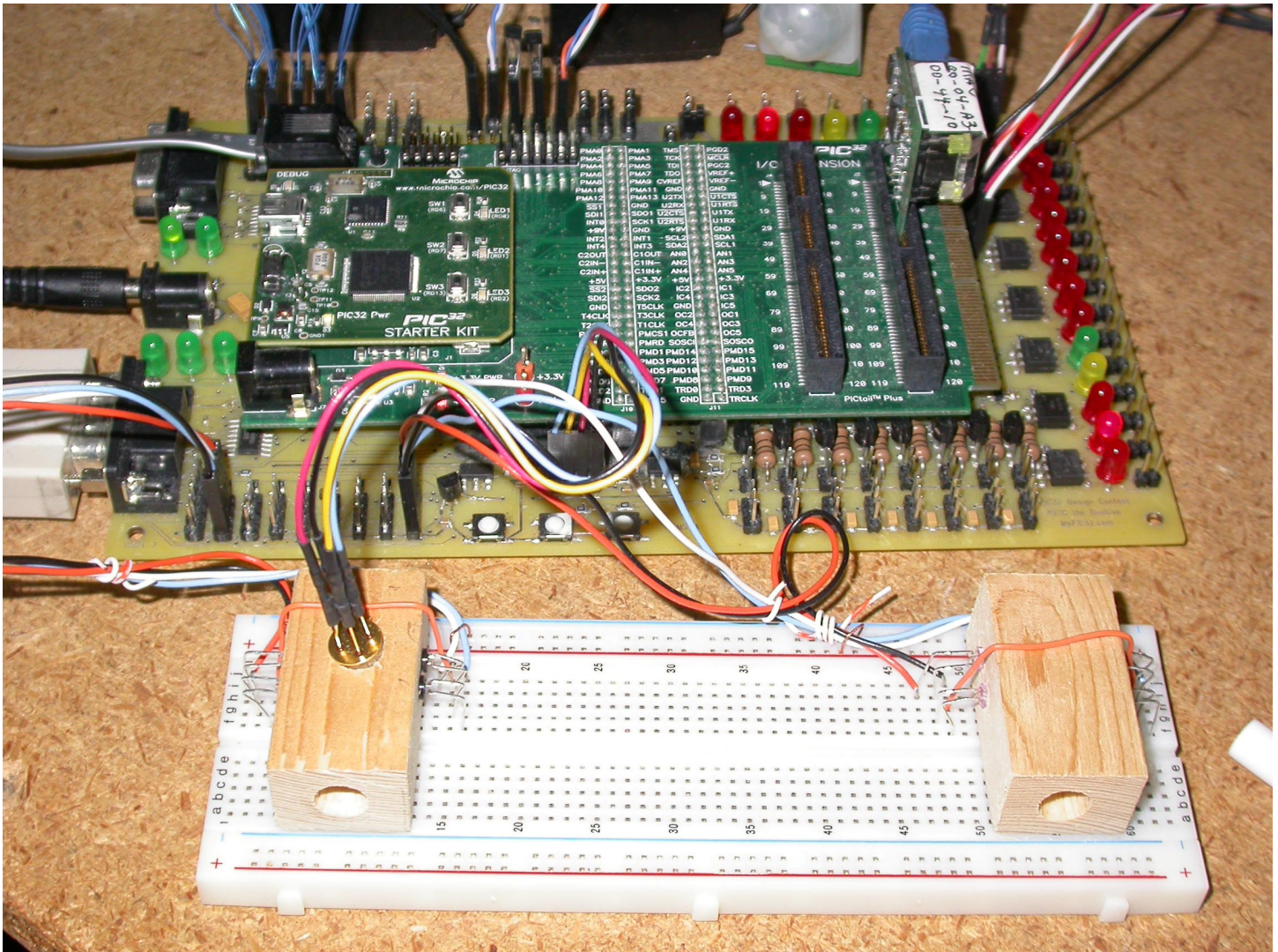


Figure 40 A later version of the bee tunnel is seen here with the non-contact IR temperature sensor mounted on the bee tunnel on the left. Not shown are additional versions with the entrance servos and the bee scales configured on them.

Closing Statement

If you are not into bees, I think you will still find this project helpful in many other applications. Any task requiring data acquisition, analysis and web page presentation should find this project useful for ideas. You should be able to find helpful techniques for both hardware and software solutions in applying the Microchip PIC32 processor development tools.

As I stated in the beginning of this project report, I am in the process of setting up a web site CarlTheBeekeeper.com to post future updates after the contest ends. I will be including much more material that I was not able to post here. I think the possibilities are enormous as to the potential use, functionality, small foot print and extremely low cost the PIC32 offers as a solution to real time data acquisition and analysis. This PIC'n the Beehive project has demonstrated how much can be accomplished with the Microchip PIC32 and not even come close to running out of processor resources.

Bio- Carl the Beekeeper



I'm a retired engineer scientist after working 31 years at a major computer company. A few years before retiring I became interested in beekeeping. I have a passion for electronics and to use that knowledge to develop gadgets to do interesting tasks. I thought this contest was the perfect opportunity to give my idea of a high-tech beehive a try. I have been playing and designing with 8 bit micros since the 8008 first came on the scene over 30 years ago. I seem to be busier now doing special projects than I was working a regular job, but I love what I do.