EnviroMonitor Users Manual

Ver 1.0
Jun 8 2008

The Reliable Glacial Monitoring Solution

Windmill Publishing House

Environmental
Monitoring System
With Missile Defense

EnviroMonitor 188S

TABLE OF CONTENTS

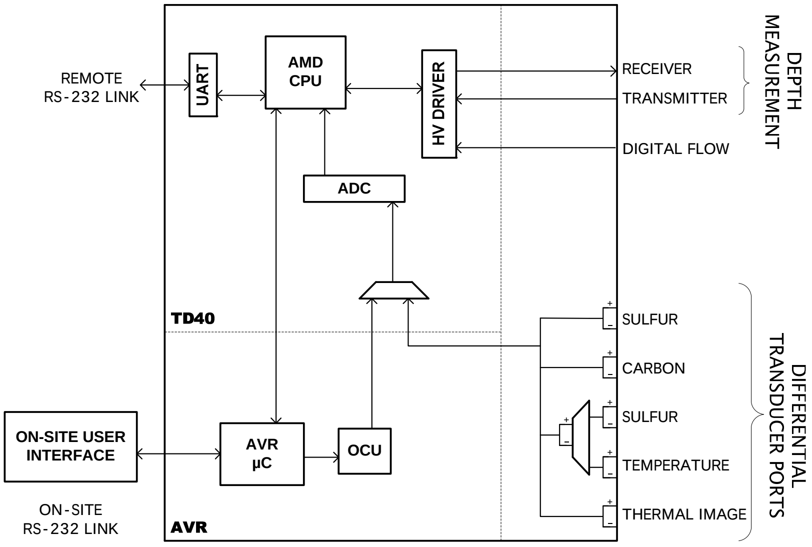
Environmental Monitoring System With Missile Defense

EnviroMonitor 188S

FEATURES

4 Differential Input Ports (0-5V)
1024-Bit Resolution
1 Digital Measurement Port
Frequency Range 10-10kHz
1 Receiver/Transmitter Temporal Measurement Pair
Freq. Range 35-1.75kHz
User-Settable Measurement Configurations
Onboard Calibration Unit For Remote Diagnosis and Testing
Lower Power Consumption in Sleep Mode (<5mW)
Buffered I/O for Reliable Operation
Operation from 8.5V-12V
On-Site Operation Terminal
Remote Operation Terminal
Extensive Error Logging with Reports to both Terminals
Reliable CPU Core (Tern TD40)
Flexible/Expandable Design

FUNCTIONAL BLOCK DIAGRAM



PRODUCT DESCRIPTION

The EnviroMonitor is a high-reliability monitoring system built upon the Tern TD40 platform. It features several measurement capabilities tailored to the monitoring of glacial environments; however it is designed for flexibility additional functionality can be added at the user's request. To accommodate the remote-deployment nature of such system as the EnviroMonitor, both local and remote interfaces are provided that are both reliable and efficient.

The standard model utilizes four differential sensor inputs that allow for basic environmental monitoring. A temporal transmitter/receiver pair is also incorporated into the unit, and is designed to accurately measure ultrasonic pulses used in glacial depth measurements. Additional inputs include a differential thermal imaging unit, and a digital transducer input to utilize the fluid dynamic field's latest digital transducer technology.

RELIABILITY AND OPERATION

The primary focus in the design of a system such as the EnviroMonitor is to provide a high-reliability monitoring system for deployment in remote locations with extreme weather conditions. As such several features have been added in addition to the fundamental monitoring core. These features focus on software reliability, remote device calibration and checking, and extensive diagnosis capabilities. Use of this device is intended to require minimal maintenance, and this design philosophy has been incorporated into most of the peripheral features, in addition to the operating system core.
TCP/IP or UDP, are included in the development of this protocol.

An onboard calibration unit (OCU) is also employed with the unit, allowing the remote to quickly debug and potential errors associated with the device, and to calibrate the system remotely if necessary. This unit provides simulation for all of the peripheral measurement ports, and is designed to test the full range of each input.

GENERAL DESCRIPTION AND THEORY

The EnviroMonitor is intended for use as a standalone device, accessible via remote serial connection. A local interface is also provided for on-site measurement, calibration and simulation during installation and maintenance. The primary measurements incorporated into the device include:

- Ambient Temperature
- Glacial Run-off Flow
- Ambient Carbon and Sulfur Levels
- Thermal Activity
- Digital Flow Rates

The data acquisition has been designed as a firm-time system, but all measurements may be upgraded to real-time constraints if necessary. The primary purpose of this design choice was to keep the overall system's processor consumption within moderate levels to maximize the reliability of the operating system's performance.

For the remote connection, a simple and reliable protocol was developed with the intention of facilitating easy user interfacing. Many aspects of higher level protocols, such as

USE CASES

The following use cases have been established for the current model:

Remote User Use Cases

The following use cases have established for the remote user, and comprise the following use diagram.

Set Limits
- User set range for warning and alarm states
Take Measurements
-User can take different desirable measure
Display
– User can display the collected and computed data
Calibration
– Compute collected data
Log Data
- User can request to log data collected

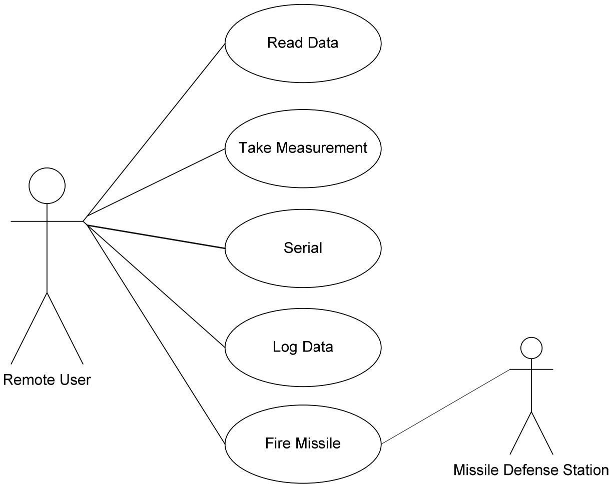


Figure 1: *Remote User Use Cases*

Local User Use Cases

The following use cases have established for the local user, and comprise the following use diagram.

Read Data
– User can read transmitted data
Take Measurements
- User can request measurements
Serial
– User can request serial transition
Log Data
- User can request data logging
Fire Missile
– User can fire missile at Missile Defense Station

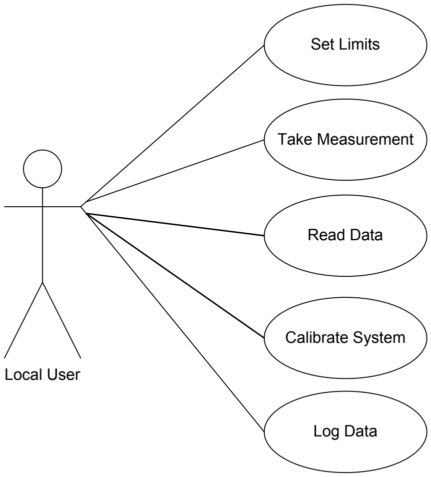


Figure 2: *Local User Established Use Cases*

OPERATING PLATFORM

The µC/OS platform was utilized for the development of the system. This system was chosen for its combination of low-cost and high reliability. It allows for safe integration of the several functional components of the measurement system in a simple and firm-time manner.

This operating environment also allows for dynamic scheduling, event timeouts, and extensive error logging capabilities. See 'System Architecture' for more information on these topics.

The µC/OS architecture finally presents a simple and highly deterministic structure, with all source code provided to the designer. This allows the designer complete control of the hardware, and the ability to meet the demands of any real-time constraints imposed upon the system. Specific constraints for the current model include:

Hard Time
– Thermal Image Processing
– Glacial Measurement Timing
– Digital Flow Timing
Firm Time
– Serial Response
– Sample Rates
Soft Time
– Local Interface Response

REMOTE INTERFACE

An RS-232 link was incorporated into the system design for the remote user interface. It allows for measurement readings, data logging, DAQ control, measurement calibration, CPU diagnostics, and error reports if necessary. This link is accompanied with a simple transmission protocol for the remote user to gain access. Sample code for the user interface is also provided. Please see the 'Remote Protocol' for more information.

SERIAL PROTOCOL

A serial communication protocol is devised for remote command and data transmission. The Serial I/O uses an asynchronous serial protocol with a transfer rate of 9600 bit/s. Messaging between system and console consists of ASCII characters with checksum for error detection. The protocol consists of an exchange of Information and Control Frames.

LOCAL INTERFACE

A local interface has been developed for onsite installation, calibration, and maintenance. The system also presents to the local user all of the current system measurement values. The local interface includes for the local user

LCD Display
- For display of current measurements.

Keypad
- For system interaction and calibration.

Visual Status Indicators
- For visual indication of the environmental status.

Audio Alarm Indicator
- For audible alarm condition annunciation.

Local RS-232 Link
- For system and OCU configuration.

MAINTENANCE CONSIDERATIONS

Extensive use of error logging was incorporated into the EnviroMonitor to allow for exact determination of faulty operation. This is intended to reduce the amount of downtime for the system in the event of an error. It also allows for remote diagnosis of the system error, allowing for optimization of the maintenance team and accompanied equipment, saving the user significant amounts of time and money when deploying a response team.

Additionally, an onboard calibration unit (OCU) has been incorporated into the design, allowing for both local and remote testing of the system hardware. This is primarily to provide the user with the ability to debug faulty transducers, a serious issue in harsh arctic environment where accuracy and lifespan are difficult to simultaneously maintain.

In line with the OCU, the system has also been provided additional flexibility with its A/D conversion parameters; all of which are flexible and customizable for the user. This allows for flexibility in transducer selection for the user after product deployment. The current model is shipped with standard settings for the initial customer transducer specifications; see ‘Components Listing’ for more detail.

EXPANDABILITY

The current hardware configuration of the EnviroMonitor utilizes the highly-customizable Tern TD40 platform. This platform includes several I/O ports of many different interfaces; at the time of development only 45% of the possible ports are in use. As such, at the users request the current model can be adapted to a virtually limitless array of potential applications.

Additionally, for the software development, the current model is shipped with a 32KB ROM unit. The unit is easily upgradeable to a 512KB unit, and any additional design features or communication links can easily be accommodated. The µC/OS task environment also easily supports this expandability, and its pre-emptive, deterministic capabilities allow for more complex designs to still accurately meet their timing needs.

MEASUREMENT SYSTEM

The current model incorporates four methods of measurement:

1. Differential Analog Measurement	4 Ports
a. Analog Transducers	
2. Differential Spectral Analyzer	1 Port
a. Thermal Imaging Unit	
3. Timed Transmitter/Response Unit	1 Port
a. Depth Measurement Unit	
4. Digital Transducer Unit	1 Port
a. Flow Transducer Unit	

Analog Measurement Ports

The analog ports interface with the TD40's onboard A/D, the LTC2543. It is a 12-bit, switched-capacitor, successive approximation A/D. It communicates via SPI with the TD40 CPU, and has a total of eleven channels. Four channels are provided with the current model. It allows for the following ranges of operation:

Maximum Range

5V Maximum Input Voltage

Default Range

The following default ranges have been implemented on the EnviroMonitor per user specification. All settings can be modified at run time through the local interface.

Temperature Port

- The temperature port is configured for an input of 0-200mV maximum reading.

Flow Port

- The temperature port is configured for an input of 0-50mV maximum reading.

Carbon

- The temperature port is configured for an input of 0-100mV maximum reading.

Sulfur

- The sulfur port is configured for an input of 0-100mV maximum reading.

Range Considerations

The LTC2543 will be damaged by applying an input of more than +5V. If necessary, the gain interface, as will be detailed next, can be configured to attenuate the inputted signal to a maximum input voltage of ±32V. This feature is not shipped with the current model.

The current model is shipped to report measurements above the set A/D limits as the maximum limit, and this feature should be considered in the user design.

Interfacing Considerations

The TD40 platform employs the following comparator interface (LM324) for the A/D ports, and its input impedance is important to consider when interfacing a device. It is as follows:

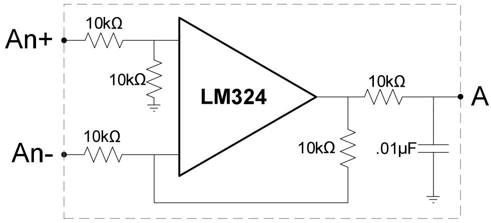


Figure 3: *TD40 A/D input configuration.*

The variable gain amplifier depicted Figure 3 is designed to allow variable gain of the inputted signal to maximize the input resolution, up to 100dB.The current model is deployed with unity gain; however future models can employ higher resolution gain at the user's request.

Also to note, an analog multiplexer is utilized for the reading of the carbon and sulfur values into one A/D port. This is accomplished with a CD4043 analog multiplexer.

Expandability Considerations:

The current TD40 platform provides a 160µs default A/D access time. Thus hard-time sampling events, such as the current FFT spectral analysis, are limited in bandwidth by this read-time constraint (approximately 2kHz maximum). The cost of upgrading this system to accommodate higher bandwidth then should be considered for future extensions.

Differential Spectral Analyzer Port

This port utilizes one of the LTC2543's A/D ports, and also is channeled through an LM324 configuration. The only distinguishing characteristic for this port in the current model is that it is configured through software for exclusive FFT data-acquisition. This also can be customized per user request.

Transmitter/Receiver Port

The EnviroMonitor includes a port to provide a digital stimulus and response that can accurately measure the ping response time of a transmitter/receiver pair. The device has a timing precision of ±50µs, and is calibrated to output a 1ms pulse upon request

Operation

The timing is measured via a timer onboard the AMD CPU. The timer used is the internal 16-Bit Timer 2, and it is calibrated to a 50µs clock period. The count is started at the rising edge of the waveform, and is halted at the return ping's rising edge.

The transmitter takes advantage of the µC/OS environment by dynamically 'halting' the measurement task attributed to this measurement, and waiting for a response. If a response fails, the system will retry five times before reporting an error to the OS. Please see 'System Architecture – Glacial Depth Task' for more information on this event.

It is important to note that the EnviroMonitor does not include the transducer circuitry; rather it only includes the 0-5V digital stimulus and reception ports. Sample circuits for the transmitter/ receiver pair have been included to illustrate design of such devices; please see the ‘Resources’ section.

Interfacing Considerations

The output of the glacial stimulus is through the TD40’s onboard High Voltage Driver, the ULN2003 Darlington Configuration. This device should be accommodated into any transducer designs, and a schematic is as follows, courtesy of the TD40 documentation:

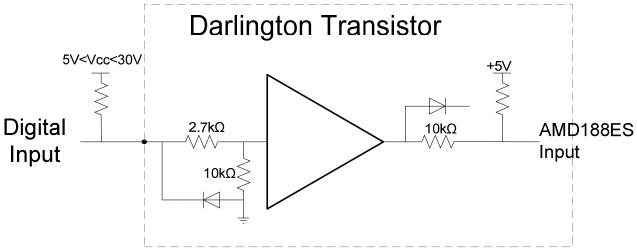


Figure 4: *Internal configuration of the glacial ping input port. Pictured is one of the ULN2003A drivers.*

It is important to note in Figure 4 that the external pull-up resistor is required by the user and is not part of the current model. This capability allows the user flexibility in deploying the current model with a wider range of potential receivers.

The transmitter output is also interfaced through the same configurations, and a pull-up resistor is also required for this port.

Recommended Pull-Up Resistor

A 1.2kΩ is recommended for this port.

Recognized Logic Levels

The ULN2003A configuration yields a low-level recognition of .8V, and a high recognition higher than 3V and less than 30V.

Digital Transducer Port

The digital transducer port also utilizes the Darlington buffer described in Figure 8. It is a standard interfaced, designed to accommodate the following wireless transducer:

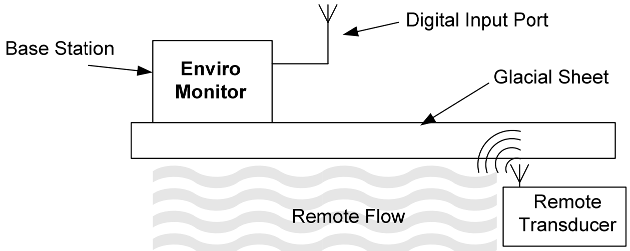


Figure 5: *Sample application of the digital transducer port.*

REMOTE COMMUNICATION

The current model uses the SCC2691 UART for the RS-232 serial interface. The baud rate generator is directly operated using an 8MHz crystal providing a default 9600 bit/s rate. It incorporates several features including full-duplex asynchronous receive/transmit; 8-bit data transfer, hardware flow control, and interrupts for transmit and receive.

REMOTE INTERFACE

External systems can interface with the EnviroMonitor via the RS-232 serial port. To reduce hardware dependency on the serial interface, the system is designed with simplicity in the hardware setup and software protocol. The system uses a defined information frame for sending data and a control frame for control purposes. All information sent and received must be in ASCII characters.

To ensure correct data is transferred, a simple handshake scheme is employed. For every message transmitted, the receiver must acknowledge with an ACK or NAK message indicating the message was received correctly or incorrectly, respectively. The ACK or NAK is determined by computing the message checksum and length.

To interface with the system properly, the external system must configure its serial port to the following settings specifying the baud rate, data bits, stop bits, parity check, and flow control.

– BAUD Rate	9600 bit/s
– Data Bits	8-bit
– Stop Bits	1-bit
– Parity Check	none
– Flow Control	none

Flow control is issued by the system using a single ASCII character XOFF (`0x13`) or XON (`0x10`) to the external system. The serial contains a 75-character in-buffer and out-buffer. When the buffer size reaches 70 characters, an XOFF is sent to the transmitting device, but the system can still receive at most 5 more characters. After which, an XOFF is sent and no data is accepted when the buffers reach 75 characters. When the buffer drops to 50 characters an XON is transmitted, and the system will continue to accept more characters. The following are valid ASCII characters:

- XOFF (`0x13`), XON (`0x10`)
- 0-9, A-F
- I, S, P, D, L, M, W
- start header (`0x01`)
- end (`0x0A`)
- space (`0x20`)

REMOTE PROTOCOL

The following describes the remote protocol. The user must conform to the following specifications for proper transmission. This part is broken into two sections. Section one describes the Information Frame and the second section describes the Control Frame.

The following activity diagram then is established for the remote protocol:

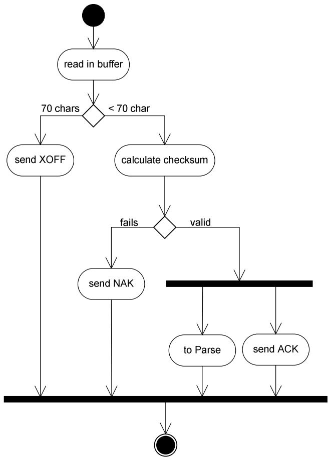


Figure 6: *Activity Diagram for the remote serial protocol input*

Information Frame (I-Frame) Format

The message includes ASCII characters (0-9, A-Z) and three control characters (start header, space, and return carriage).

{ [start][length][body][checksum][end] }

Start and End

A message begins with a start character (0x01) and ends with an end character (0x0A).

Length and Checksum

The correctness of the message is verified by calculating the checksum and length of the message. The length is sent as four ASCII characters that includes Start and End frames with variable-sized body information.

The checksum is sent as two ASCII characters that correspond to hexadecimal XORing from the start character to the body last character. Excluded from the checksum frame are the two checksum characters and the end character.

Body (Data)

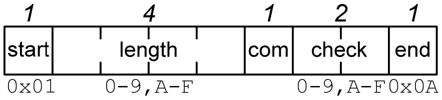
The body comprises of a command character followed by the measured data information. A maximum of 128 bytes is allowed for the body information.

Command Format

The command format consists of a character in the I-Frame body indicating the command to be performed by the system.

Type	Command Description
I	Initializes the communication between system and the user-interface application.
S	Starts the measurement tasks by directing the hardware to detect sensor signals.
P	Indicates the STOP mode. Terminates any measurement tasks
D	Enables data logging.
L	Disables data logging.
M	Requests the most recent measurements o be transmitted to the serial output.

A typical *command* message is formatted as follows. The top number indicates the number of bytes, and valid ASCII characters are shown below the frames.

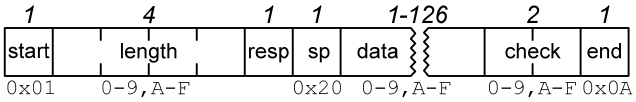


Response Format

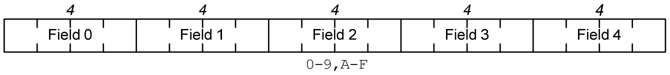
The response format consists of a character in the I-Frame body indicating the response to a command.

Type	Response Description
M	Returns the measured data upon receiving the M *command*.
W	Returns the warning measured data after a warn event or an alarm event occurred.

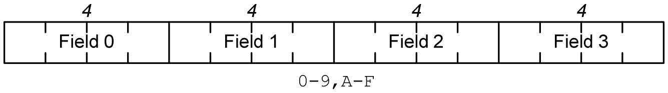
A typical *response* message is formatted as follows:



`M` *response* frames are further divided as follows:



`W` *response* frames are further divided as follows:



`M` and `W` responses share the following data formatting:

Field 0:	Temperature measured data.
Field 1:	Flow rate measured data.
Field 2:	Carbon level measured data.
Field 3:	Sulfur level measured data.
Field 4:	Thermal image measured data.

Control Frame (C-Frame) Format

Proper transmission is ensured with a handshake scheme that is implemented using Control Frames.

{ [start][type][sequence number][end] }

- Every information frame transmitted from system or console must be acknowledged (ACK `0x06`) or negative acknowledged (NAK `0x15`).
- Every information frame must be ACK/NAK before the next message transmission.
- ACK frame indicates the transmitted message was received correctly (length and checksum match).

- NAK frame indicates the transmitted message was NOT received correctly (length and/or checksum did not match).
- If a NAK is received, the last information frame must be resent up to 3 consecutive times. On the fourth retry and a NAK is received, the receiver must generate a link-down error, and the sender must re-initialize the serial-link with an I-information frame.
- Upon startup, the user can initialize the serial-link with an I-informational frame. An ACK indicates the link is established.

A typical *ACK* or *NAK* is formatted as follows:

1	*1*	*1*	*1*
start	hand shake	Seq-ctl	end
0x01	0x06 0x15	0-7	0x0A

Start: Start of frame (0x01).
Handshake: Control frame type (ACK/NAK).
Sequence Number: The sequence number is modulo 8 with ASCII characters between 0-7.

ONBOARD CALIBRATION UNIT

The onboard calibration unit is primarily designed for local accessibility. It allows the user to configure system limits used for the primary measurement warning unit. For the Measurement system, it allows for configuration of the following parameters for each of the four analog measurements:

Maximum ADC Count	Per Channel
Minimum ADC Count	Per Channel
System Warning Level	Per Channel
System Alarm Level	Per Channel
Digital Flow Conversion Rate	Digital Flow

The system also provides simulations for all of the system measurements, with the exception of the thermal imaging sensor. This is accomplished via a 2-to-1 multiplexer bank, with optional pre-calibrated simulations. They are as follows:

Analog Measurement Simulation

The analog simulations provided utilize pre-calibrated analog potentiometers, with resistance ranges calibrated to meet the user specification for transducer ranges on the current model. These potentiometers were implemented as indicated in Figure 7:

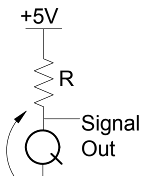


Figure 7: *Analog Simulation*

Glacial Depth Simulation

The glacial simulation is accomplished via a response algorithm implemented on the AVR µC. The algorithm has the following user settable parameters (italics denote default settings, followed by the input range):

- Glacial Depth [*3950m*; 30m-5km]
- Transmission Velocity [*3500m/s*; const]
- Glacial Melt Rate [*100m/min*; 1-1000m/min]

The hardware mechanism on the AVR used to implement this functionality is a high-priority maskable interrupt. The subroutine for this event is a simple variable-delay mechanism that has a resolution of ±5ms. The subroutine concludes by returning a 1ms ping at the conclusion of the simulated delay. Please see the 'System Source Code' for detailed implementation.

Digital Flow Simulation

The digital flow simulation is a simple variable-frequency digital signal generated by the AVR. It has the following user settable parameters:

- Transducer Frequency [*1kHz*;1Hz,1kHz]
- Transducer Conv. Rate [*333*;1-999]

Thermal Imaging Simulation (Not Included)

The thermal imaging simulation imposed too much overhead and cost, and thus was not included in the current model. However the following describes its theory, if calibration is necessary.

Recommended Test Apparatus

A simple function generator can be utilized to test the thermal imaging apparatus. To interface with the port, the following range is recommended:

Amplitude:	2.25V
Offset:	1.125V
Frequency:	30-1.75kHz

LOCAL INTERFACE

The local interface is comprised represented by the following Schematic:

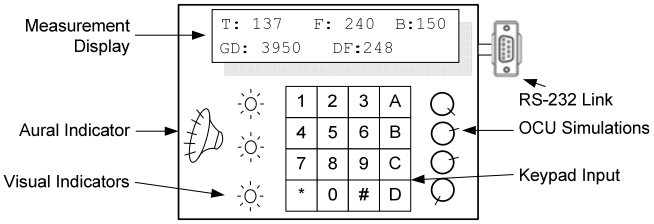


Figure 8: *Local Interface Schematic*

This interface is comprised of the following components:

Measurement Display
This unit is a 16x2 LCD Character Display, displaying all of the current system parameters.

Aural Indicator
A speaker is included to provide aural indication of an alarm state.

Visual Indicator
A set of green, yellow and red LEDs are provided to indicate the current system state as indicated by the current measurements, and user inputted limits.

RS-232 Link
A DE-9 connection is provided for the RS-232 Local Interface Link.

OCU Simulations
Simulation knobs are provided for local OCU operation.

Keypad Input
A keypad is provided to adjust the system settings, and the OCU calibration.

RS-232 Interface
The following is an example of the supplied user interface:

The local interface is intended to provide all functionality necessary to install, calibrate, and maintenance the system. It is important to note that the design is simplistic and optimized to reduce cost and complexity. The user should then note then that an arbitrary amount of features and complexity can be accommodated upon request in future versions.

EXPLANATION OF WINDMILL MECHANICS

The following illustrates the proper implementation of a windmill, taken from 17th century industrial Holland:

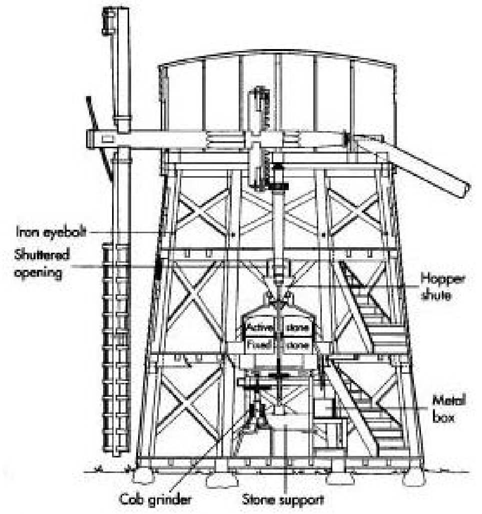


Figure 9: *Windmill illustration*

Notice the skillful inclusion of the iron eyebolt, in addition of the bronze hopper.

SYSTEM CHARACTERISTICS

The following section details the performance behavior of the EnviroMonitor. Due to the rapid-development employed for the current model; several statistics are only hypothesized and have not been full investigated for accuracy.

Thermal Imaging Sampling Accuracy

The following data represents the accuracy of the FFT port with respect to sample frequency. This data then presents both upper and lower bounds on the current model's performance.

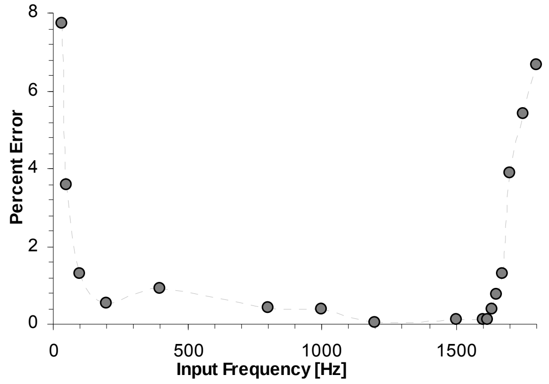


Figure 10: *Thermal measurement accuracy vs. frequency*

Task Execution Times

The following is a depiction of the task execution times for the EnviroMonitor. Task execution times were not taken under normal system operation for the current model; please see the 'Open Issues' section for more detail.

Table 1: *Task execution times*

Task	Execution Time
Analog Measurement Task	3.68ms
Thermal Task	146ms
Digital Flow Task	
Glacial Task	
Computation Task	370ms
Warning Task	202.5µs
Alarm Handler Task	
Warning Handler Task	
Status Task	
Display Task	1.65s
Command Task	
Serial Parse Task	
Serial Com Task	
User Handler Task	

Remote Communication Link Throughput

The following presents the hypothesized relationship between the embedded serial link with the remote station, and the current CPU consumption.

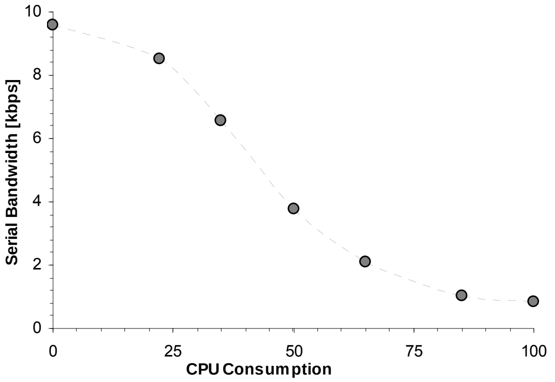


Figure 11: *Serial Bandwidth considerations with respect to CPU consumption.*

User Interface Response Time

The following depicts the hypothesized relation between CPU consumption and the user interface response time. Future models are intended to report this quantity, and attempt to minimize its affects.

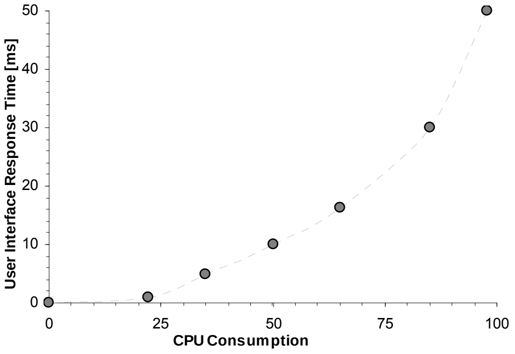


Figure 12: *User interface response time.*

Glacial Depth Measurement Accuracy

The following is a presentation of the hypothesized accuracy of the glacial measurement accuracy. It is partially based upon qualitative preliminary results' however further testing is required.

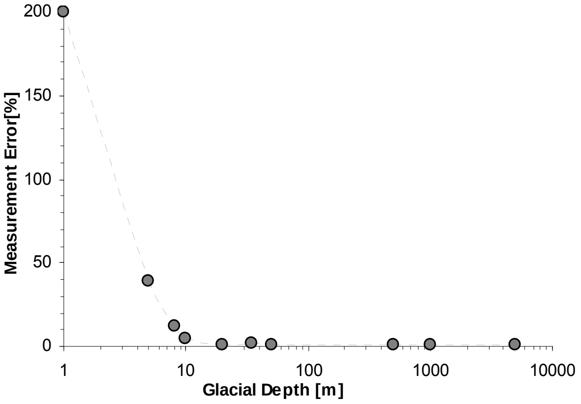


Figure 13: *Glacial depth accuracy vs. depth.*

LINEARITY OF THE ANALOG TRANSDUCERS

The following is a data set containing the linearity of the supplied measurement transducers, in addition to linear-fits of each device's response within their linear region:

Temperature Transducer

The following describes the behavior of the current model's temperature transducer:

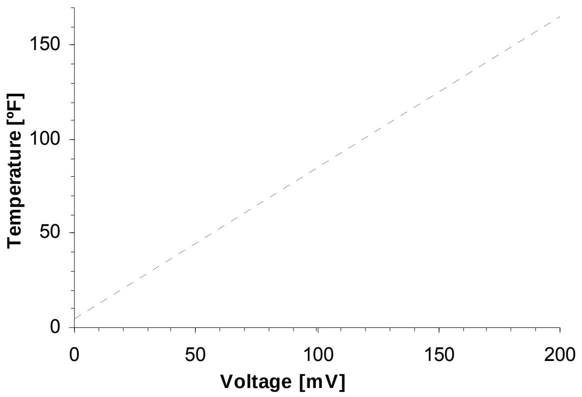


Figure 14: *Temperature transducer response*

Range of Operation

Maximum:	200mV
Minimum:	0V

Linear Transducer Fit

$$Temperature = 5 + .8 \cdot \text{Voltage}$$

Flow Rate Transducer

The following describes the behavior of the current model's temperature transducer:

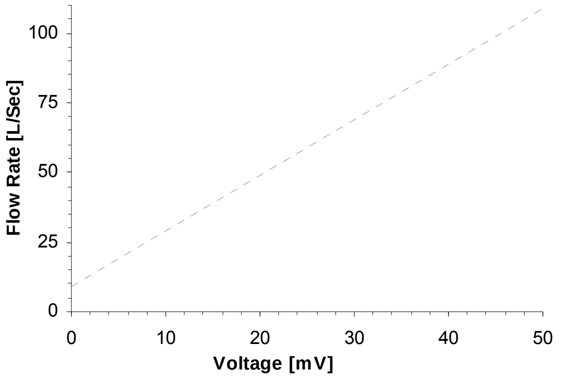


Figure 15: *Temperature transducer response*

Range of Operation

Maximum:	50mV
Minimum:	0V

Linear Transducer Fit

$$FlowRate = 9 + 2 \cdot \text{Voltage}$$

Carbon Level Transducer

The following describes the behavior of the current model's temperature transducer:

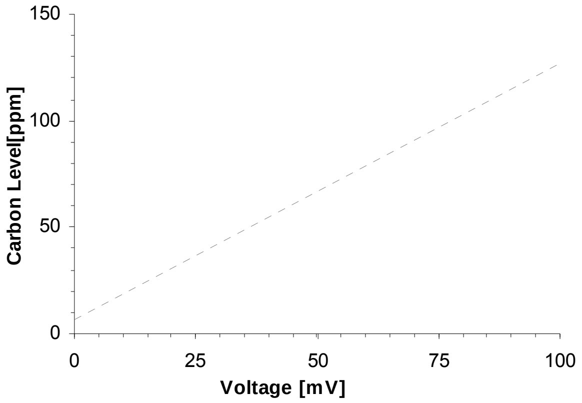


Figure 16: *Temperature transducer response*

Range of Operation

Maximum:	100mV
Minimum:	0V

Linear Transducer Fit

$$CarbonLevel = 7 + 1.2 \cdot \text{Voltage}$$

Sulfur Transducer

The following describes the behavior of the current model's temperature transducer:

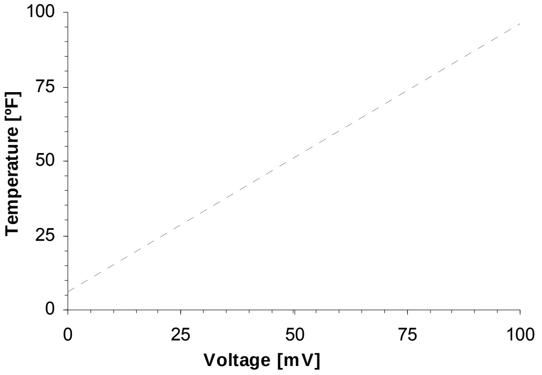


Figure 17: *Temperature transducer response*

Range of Operation

Maximum:	100mV
Minimum:	0V

Linear Transducer Fit

$$SulfurLevel = 6 + .9 \cdot \text{Voltage}$$

ABSOLUTE MAXIMUM ELECTRICAL RATINGS*

Operating Temperature	-55ºC to +125ºC
Input Voltage on Glacial Port	-30V to +30V
Input Voltage on Digital Flow Port	0V to +5.0V
ADC Input Voltage	0V to +5V
Maximum Operating Voltage	30.0V
DC Current Per ADC Pin Pin	50mA
DC Current for the Glacial Port	200mA
DC Current VCC and GND Pins	1.5A

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS

Parameter	Conditions	Min	Typ	Max	Unit
ANALOG INPUT PORTS					
Voltage	Std.	0		5	V
THERMAL IMAGING PORT					
Voltage		0		5	
Frequency		30		1750	Hz
GLACIAL OUTPUT PORT					
Voltage		-50	5	50	V
Current				350	mA
Pulse Duration		0.500	1	4	ms
GLACIAL INPUT PORT					
Voltage		0	5	30	V
Current				350	mA
Return Timeout		4.5	5	5.5	sec
Logic Low			0.8		V
Logic High			3		V

PIN DESCRIPTION

The following is a Pinout of the EnviroMonitor:

Table 2: *Pinout descriptions*

Pin No.	Mnemonic	Function
A	Remote Serial Link	DE-9 Connection for Remote Access
B	Local Interface Connection	DE-25 Connection to the Local Interface
14	Vcc	Supply Voltage
15	Gnd	Device Common
1	Glacial Output Port	GO
2	Glacial Input Port	GI
3	Digital Flow Port	DF
4	Temperature Port (+)	T+
5	Temperature Port (-)	T-
6	Flow Port (+)	F+
7	Flow Port (-)	F-
8	Carbon Level Port (+)	CL+
9	Carbon Level Port (-)	CL-
10	Sulfur Level Port (+)	SL+
11	Sulfur Level Port (-)	SL-
12	Thermal Imaging Port (+)	TH+
13	Thermal Imaging Port (-)	TH-

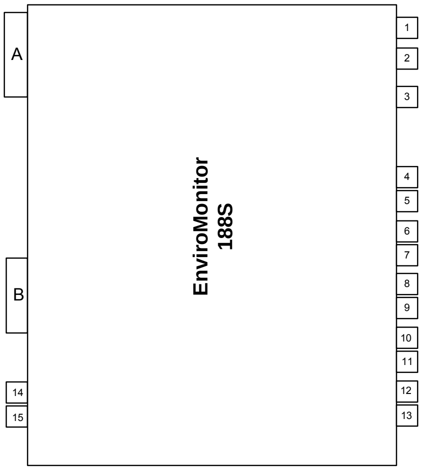


Figure 18: *Pinout of the EnviroMonitor*

Table 3: *Timing specification for the Timer 0/uCOS software timer*

C Source Code	Assembly	Opcode	Clocks
//Enter ISR	push bp	PUSH r16	10
	push dx	PUSH r16	10
	push ax	PUSH r16	10
	mov bp,sp	MOV r16,r16	13
	mov dx, 0FF36h	MOV r16,imm16	4
	mov ax, 0000Fh	MOV r16,imm16	4
	out dx,ax	OUT DX, AX	11
call_OSIntEnter	_call	CALL FAR	31
OS_ENTER_CRITICAL()		CLI	2
OSIntNesting++;		INC m8	15
OS_EXIT_CRITICAL()		SLI	2
	ret	RET	30
OSTaskCtr[OSTCBCur->OSTCBPrio]++	mov bx, [_ostcbcur]	MOV r16, 16	13
	mov al, [bx+0x05]	MOV AL, moffs8	8
	mov ah,0	MOV AL, moffs8	8
	add ax,ax	ADD r16, r16	3
	mov bx,ax	MOV r16,r16	13
	inc [bx+_ostackctrs]	INC m16	19
outportb(0xff22,0x0008)	mov dx, 0FF22h	MOV r16,imm16	4
	mov ax, 0008h	MOV r16,imm16	4
	out dx, ax	OUT DX, AX	11
call _OSIntExit	_call	CALL FAR	31
	routine'		*207 552*
	ret	RET	30
//Leave the ISR	pop ax	POP r16	14
	pop dx	POP r16	14
	pop bp	POP r16	14
	iret	iret	28
			560 908

14.0-22.7µs

Table 4:*Timing Specification for the digital flow response*

C Source Code	Assembly	Opcode	Clocks
//Enter ISR	push bp	PUSH r16	10
	push dx	PUSH r16	10
	push ax	PUSH r16	10
	mov bp,sp	MOV r16,r16	13
	mov dx, 0FF36h	MOV r16,imm16	4
	mov ax, 0000Fh	MOV r16,imm16	4
	out dx,ax	OUT DX, AX	11
call_OSIntEnter	_call	CALL FAR	31
OS_ENTER_CRITICAL()		CLI	2
OSIntNesting++;		INC m8	15
OS_EXIT_CRITICAL()		SLI	2
	ret	RET	30
digFlowCounts[ISRCount] = pingCount	mov bx, [0x009e]	MOV r16, m16	13
	mov cl, 0x02	MOV r8,imm8	3
	shl bx, cl		
	mov dx, [0x0058]	MOV r16, m16	13
	mov ax, [_pingcount]	mov r16, m16	13
	mov [bx+0x0064],dx	MOV moff16, AX	13
	mov [bx+_digflowcounts],ax	MOV moff16. AX?	13
if(ISRCount == 5)	cmp word_ptr[0x009e],0x05		
	jnz #os_files#495		
	mov al, 0x14	MOV r8,imm16	4
	push ax	PUSH r16	14
OSTaskResume(DIG_FLOW_PRIORITY)	call_	CALL	30
	Routine		*430 981*
	ret	RET	31
	pop cx	POP r16	14
outportb(INT1CON,INT1_MASK)	mov dx, 0xff3a	MOV r16, imm16	4
	mov al, 0x0f	MOV r16, imm16	4
	out dx, al	OUT DX, AL	7
ISRCount = (ISRCount+1)%6)	mov ax, [0x009e]	MOV r16, m16	13
	inc ax	INC r16	3
	mov bx, 0x0006	MOV r16, imm16	4
	cwd	CWD	4
	idiv bx	IDIV r16	53 61
	mov [0x009e],dx	MOV m16, r16	16
pingCount = 0	mov word_ptr [0x0058],0x00	MOV m16,imm16	13
	mov word_ptr [_pingcount],0	MOV m16,imm16	13
outportb(0xff22,0x000D)	mov dx, 0FF22h	MOV r16,imm16	4
	mov ax, 000Dh	MOV r16,imm16	4
	out dx, ax	OUT DX, AX	11
call _OSIntExit	_call	CALL FAR	31
	routine'		*207 552*
	ret		30
//Leave the ISR	pop ax	POP r16	14
	pop dx	POP r16	14
	pop bp	POP r16	14
	iret	iret	28
			1783 2128

44.6-53.2µs

Table 5: *Timing specification for the RTS interrupt*

Source Code	Assembly	Opcide	Clocks
//Enter the ISR	push bp	PUSH r16	10
	push dx	PUSH r16	10
	push ax	PUSH r16	10
	mov bp,sp	MOV r16,r16	3
	mov dx, 0FF36h	MOV r16,imm16	4
	mov ax, 0000Fh	MOV r16,imm16	4
	out dx,ax	OUT DX, AX	11
call_OSIntEnter	_call	CALL FAR	31
OS_ENTER_CRITICAL()		CLI	2
OSIntNesting++;		INC m8	15
OS_EXIT_CRITICAL()		SLI	2
	ret	RET	30
	push 0010h	PUSH imm16	14
call _OSTaskResume	_call	CALL FAR	31
	routine'		*430 981*
	ret	RET	30
	mov dx, 0FF22h	MOV r16,imm16	4
	mov ax, 000Bh	MOV r16,imm16	4
	out dx, ax	OUT DX, AX	11
call _OSIntExit	_call	CALL FAR	31
	routine'		*207 552*
	ret		30
//Leave the ISR	pop ax	POP r16	14
	pop dx	POP r16	14
	pop bp	POP r16	14
	iret	iret	28
			994 1890

24.9-47.3 µs

Table 6: *Timing specification for the Timer 2 interrupt*

C Source Code	Assembly	Opcode	Clocks
//Enter ISR	push bp	PUSH r16	10
	push dx	PUSH r16	10
	push ax	PUSH r16	10
	mov bp,sp	MOV r16,r16	13
call_OSIntEnter	_call	CALL FAR	31
OS_ENTER_CRITICAL()		CLI	2
OSIntNesting++;		INC m8	15
OS_EXIT_CRITICAL()		SLI	2
	ret	RET	30
(ULONG) pingCount++	add word_ptr[_pingCount],1	ADC m16,imm8	20
	adc word_ptr[0x0058],0	ADC m16,imm8	20
outportb(0xff22,0x0008)	mov dx, 0FF22h	MOV r16,imm16	4
	mov ax, 0008h	MOV r16,imm16	4
	out dx, ax	OUT DX, AX	11
call _OSIntExit	_call	CALL FAR	31
	routine'		*207 552*
	ret	RET	30
//Leave the ISR	pop ax	POP r16	14
	pop dx	POP r16	14
	pop bp	POP r16	14
	iret	iret	28
			520 865

13.0-21.6 µs

Table 7: *Summary of timing.*

Critical Section	Min Time (µs)	Max Time (µs)
Timer0	14.0	22.7
Timer2	13.0	21.6
RTS	24.9	47.3
FlowRespond	44.6	53.2

OPEN ISSUES

The open issues can be analyzed in four different aspects. This section is discussed by first detailing the issues that arose, the reasons why these issues arose, and the attempts to solve the problems. It should be noted that; however; the lab *should* work with an extra day for integration. As a reminder, some task specifications are incomplete due to non-operating tasks. The aspects will be discussed in the following order:

1. System Integration
2. OS Message Dispatching
3. Serial Communication
4. Thermal Imaging

System Integration

One of the more challenging aspect was integrating the new OS design, the previous tasks (lab 4), and the new tasks required in lab 5. On the previous project, the lab worked relatively well although there were some issues with the system freezing. Nevertheless, the tasks functioned properly. Integrating the new tasks (serial, parse, command, and thermal imaging) was attempted in a span of two days. This gives little margin for error and little debugging time. While the tasks performed individually rather well, it is this new integration that made these *new* individual task unstable.

The whole system was finally integrated and all tasks were scheduled to run. This does not imply, unfortunately, that every task would run when properly. Lab 4 tasks worked as expected, but there were problems in getting the messages to go from:

Command → Parse

This issue was not resolvable in time for the demo.

Severity	Solution	Time	Est. Cost
High	Re-code	5-10 hrs	$700

OS Message Dispatch

In order to study in depth of µC/OS and its capability, a new task interaction method was devised. It is essentially a new OS adapted from µC/OS source code. Lab 5 was designed from the scratch to use messages via mailboxes and queues, rather than the conventional flags. This was not easy work.

There are certain programming errors during the debugging process that resulted not so much in the logic but in the programming. Certain messages were either posted or pended to the wrong mailboxes and queues. This took considerable time in tracing the source. By using the debugger, considerable effort was made to ensure all previous tasks worked properly (they did). One problem was the dispatching of task messages (i.e. in determining which task is needed to run next). This revolves around the Command Task, in which the appropriate messages failed to call Parse Task correctly. The final cause(s) for this is still unknown.

Severity	Solution	Time	Est. Cost
Critical	Re-code	10-15 hrs	$600

Serial Communication

The heart of this lab is the serial input/output, parse, and command tasks. Due to the complexity of the messaging method, it was difficult to integrate all of the three components in approximately 1.5 days. The issue involves this operation sequence:

1. Serial receives I-Frame, returns ACK.
2. Parse receives message body from Serial.
3. Parse sends correct command to Command Task.

The issues arise on the return path where Command Task *post* the message to a mailbox such that Parse Task can *pend* the respective message. While the posting works properly, the Parse Task is unable to *pend* the message. Because of this, the return path stops before reaching the Parse Task (it is skipped as the debugger is stepped).

Severity	Solution	Time	Est. Cost
High	Re-code	5-10 hrs	$700

Thermal Imaging

Testing shows the thermal imaging task samples and FFT the properly as an individual task. But the integration of the task into the OS proved difficult. The task was *tested* and *worked* properly *before* the demo. For unknown cause, the thermal imaging task produced illegal opcode during the demo. An opcode trap implies a non-existent opcode was used. The most likely cause is improperly used pointers within the system (or thermal task).

Severity	Solution	Time	Est. Cost
Medium	Re-code	3-8 hrs	$560

EXAMPLE CODES

Interfacing with the serial communication is programming language independent. However, the software and hardware should implement the RS-232 serial protocol. To demonstrate the serial interface, a functional Java application with source code and necessary libraries are included. This application can be loaded into any computer with a serial com port.

There are two aspects to be considered: sending to the serial interface and receiving from the serial interface. To test the Java application, a Java SDK 1.4 or greater is needed. A recommendation is the Eclipse SDK development platform.

The example code has three files:

```
PortOpen.java
SimpleGUI.java
MainTest.java
```

Of particular importance are `SimpleGUI` and `PortOpen`. These two files need to be edited accordingly. The file `SimpleGUI` provides the GUI for the remote console whereas the `PortOpen` provides the Serial Transport Layer.

SimpleGUI

A text area is used to display sent and received information. Text can be entered into the area by calling the methods:

```
textArea0.enterDataNewLine(String)
textArea0.enterDataNoNewLine(String)
```

The SimpleGUI currently has four buttons.

Initialize System	Initializes the system by sending an I-Frame.
Measure On/Off	Turns on measurement tasks by sending an *S* command, and off by a *P* command.
Data Logging On/Off	Turns off any measurement by sending a *D* command, and off by an *L* command.
Current Measurement	Gets the current measurements analog measurements and thermal imaging by an *M* command.

When any one of the buttons is pressed, an *ActionEvent* is generated. This event is handled by overriding the method:

```
     void actionPerformed(ActionEvent anEvent)
```

To determine which button was pressed and to send the proper command, `PortOpen` has two methods that are called within the `actionPerformed(…)` method.

```
// tells button that was pressed
aPort.setButtons(String)

// tells there is data to send out
aPort.setSendData(boolean boo)
```

OpenPort

The core of the application lies in the `OpenPort` file. This class has three methods of interest:

```
// checks the received message for error
// builds the IFrame for transmission
void converse()

// parses the received data for command
boolean parseData(char[])

// display values or execute the command
void displayAndCommand(char [])
```

The ***`converse()`*** method provides core the sending and receiving of the message. It has two primary sections.

Receiving

Receiving data from the system is in `if(receiveFlag)` block. The input data is stored as a `String` in `scannedInput` variable. This string is first converted into a Char array using `scannedInput.toCharArray()`. A more detailed explanation is provided in the source code.

```
Check the first character in array for the start char
Check if NAK received
If NAK received 3 consecutive times, send I-Frame to reinitialize system
        If yes, send the last command
Check if ACK received
        If yes, set ackFlag true, nakCount false
Check if valid response received
    If yes, send char array to parseData()
            If checksum & length okay, pass to

    displayAndCommand()
            If not okay, send NAK
    If no, send NAK
Set receiveFlag false
```

Sending

To simplify the process, the checksum and length are predetermined, since the system only supports the mentioned commands. The sending message length for this model is always 9 bytes (ASCII characters). Pre-computed command strings are available in the `OpenPort.java` source code.

Sending data is in the `if(sendFlag)` block. The variable `sendFlag` is set true by the user if a command I-Frame is to be sent out. Before data can be sent, the `ackFlag` must be true from a previous receive.

The String `readyOutput` contains the serial formatted message. To send the message, set:

```
readyOutput = commandStringIFrame
os.print(readyOutput
```

RESOURCES

The following series of examples are intended to provide a broad reference to the user when interfacing the EnviroMonitor, and are intended towards all levels of user proficiency.

It is the intention of the designer to provide a sufficient body of resources such that the user will have significant flexibility and knowledge at hand when deploying the system.

Ultrasound Example Circuits

The following example circuit has been provided as an example for conducting ultrasound measurements. Such a circuit would be useful for interfacing with the glacial monitoring port. The designs are provided courtesy of Dave Johnson, of discoverrights.com.

Transducer Circuit

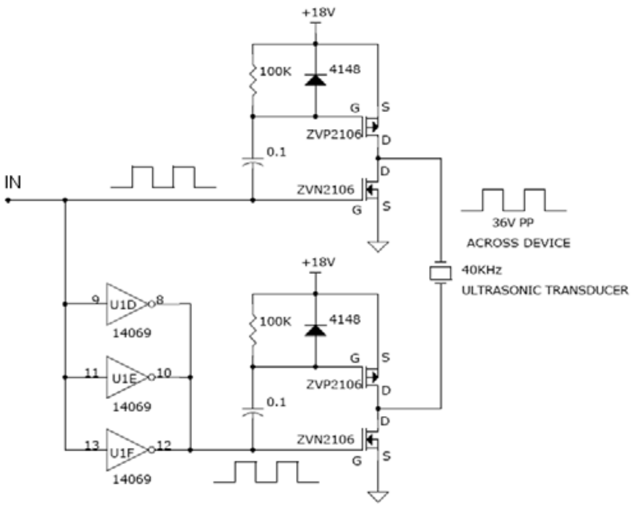


Figure 19: *Sample transducer circuit*

Description

This crystal controlled circuit drives a 40KHz piezoelectric transducer with a 30v peak to peak signal.

Source Location

http://www.discovercircuits.com/PDF-FILES/ULTRA40KHZXTR1.pdf

Receiver Circuit

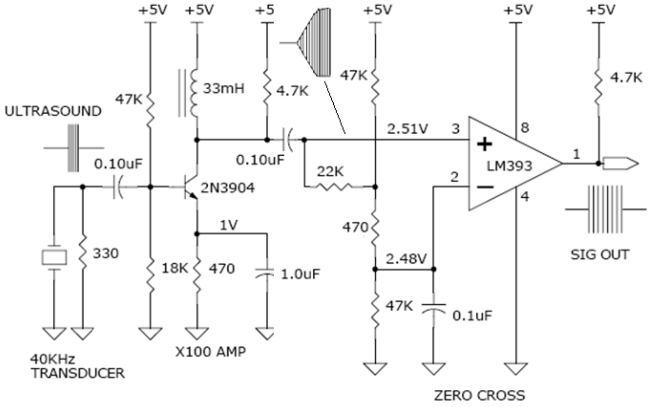


Figure 20: *Sample receiver circuit.*

Description

A X100 transistor amplifier is followed by a zero cross detector circuit, using a voltage comparator. The output is a TTL logic signal, corresponding to the received 40KHz signal.

Source Location

http://www.discovercircuits.com/PDF-FILES/40KULTRASOUNDRVR2.pdf

Disclaimer

This circuit has not been tested nor has it been confirmed to work. Rather it is an example of how one would begin to interface the EnviroMonitor with such a device. Use with discretion.

Components Listing / Source Code

As previously declared on page 15, component data sheets and a variety of other data sheets have been compiled and located online at http://justinreina.com/EnviroMonitor. Please visit the site for a complete listing of device data sheets, in addition to several informative pages of windmill design and theory of operation.

http://justinreina.com/enviromonitor.htm

SYSTEM ARCHITECTURE

The system architecture is summarized by the diagram in Figure 22. The system consist scheduler, data acquisition, computation, display, and control units. Description of each unit is as follow:

Scheduler

The scheduler of this system relies on the use of µCOS OS source code to implement real time operation. It will ensure proper sequencing of all tasks. It will statically schedule the Measure, Digital Flow, Thermal, and Status tasks. Other tasks such as Compute, Warn, and Display are dynamically schedule and terminate when the operation is complete.

Data Acquisition

Data acquisition includes Measure, Digital Flow, Glacial Depth, and Thermal Task. These tasks use the TD40 source code to implement the Analog-to-Digital Conversion in the Tern TinyDrive 40 to collect data. In addition, a differential 4-channel analog multiplexer was used in the data acquisition process due to limitation of ADC input ports.

Computation

The Compute task normalizes the collected data in specified range. Also a Fast Fourier Transformation, courtesy of Brent Plump, was implementing in to determine the signal frequency of the thermal data collected.

The display task is responsible for displaying the calibrated data and battery status. In addition, the display task will notify the user if the data collected have excess the limits.

Control

The User Handler task allow user to input the warning and alarm limits. Also this system can be control remotely via the serial port. The Serial and Parse task import and interpret the controls and notify the Command Task to request operations from the system.

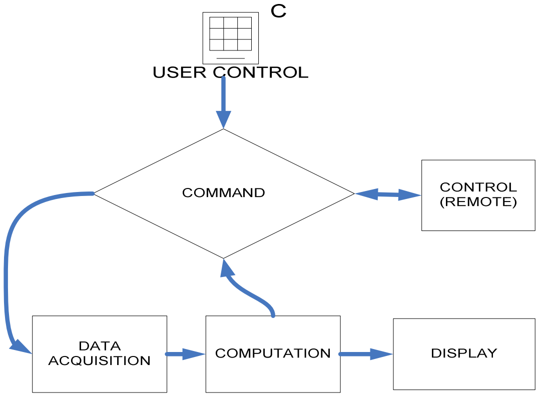


Figure 21: *High-Level System Diagram*

PRIMARY SEQUENCE DIAGRAMS

The following are a summary of the primary sequences in the operation of the Enviromonitor. While there are countless communication mechanisms employed; the following convey both the method and general philosophy of the entire system.

Warning Event

This event is generated on the event that a measurement transitions to an unsafe level. The data flow is such that it takes the following route, through the computational until into the serial com, and then out to the remote user. A sequence chart is as follows:

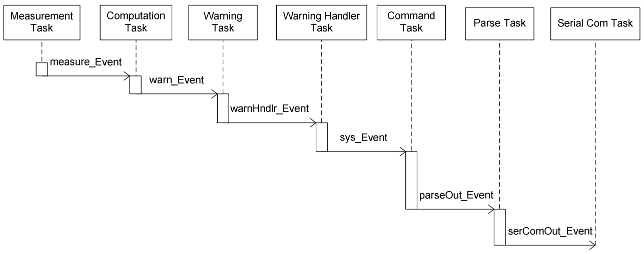


Figure 22: *Warning event sequence diagram*

Serial Measurement Request

The serial measurement request shows the sequence resulting from a remote measurement request, and highlights the capabilities of the EnviroMonitor's communication organization. It begins from the remote request, Is followed by an acknowledgement, and is concluded by the serial task 'digging' into the system and pulling out the current statistics. A sequence chart is as follows:

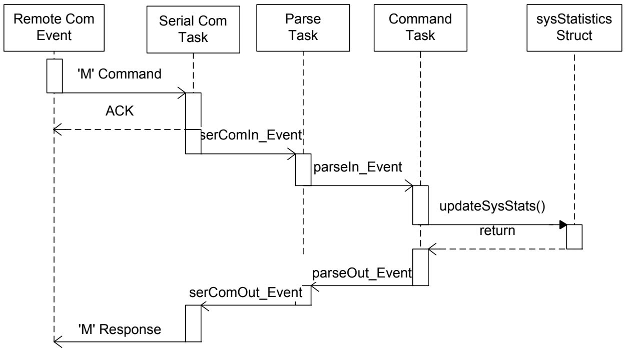


Figure 23: *Remote measurement request*

Glacial Measurement Request

The glacial request is initiated by the local user Interface, and initiates the User Handler Task. A message/request is then sent to the Glacial Task, which attempts a measurement. As with all of the timing critical tasks, the Glacial Task has a user-defined timeout period, and a maximum retry count of five. If successful, as the following scenario depicts, it then continues the data through the computation and display tasks, updating the system status. A sequence chart is as follows:

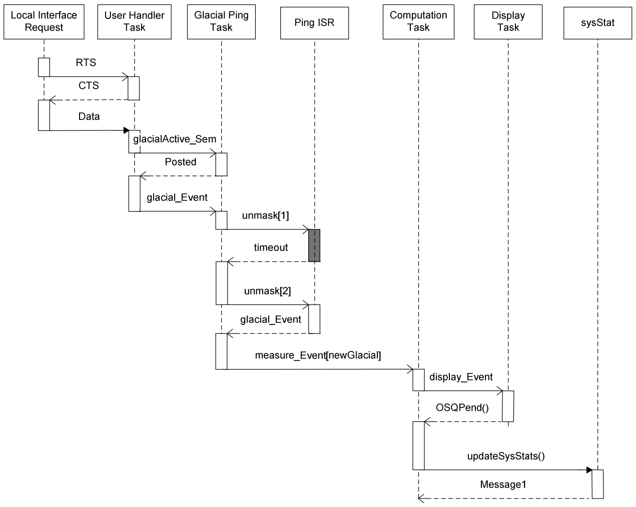


Figure 24: *Glacial measurement request sequence chart.*

INTERTASK COMMUNICATION

Task communication is accomplished via µCOS's rich set of event types. The utilized events are as follows:

- Binary Semaphores
- Counting Semaphores
- Mailboxes
- Message Queues
- Interrupt Routines

While the utilization of these mechanisms increases memory footprint, the increased flexibility of dynamic scheduling and task waiting; in addition to the real-time benefits of pre-emptive scheduling, allow for significant drops in CPU consumption, allowing all of the tasks to execute efficiently together.

DATA FLOW/CONTROL DIAGRAM

From the previously described communication mechanisms, the following dataflow/control diagram is realized.:

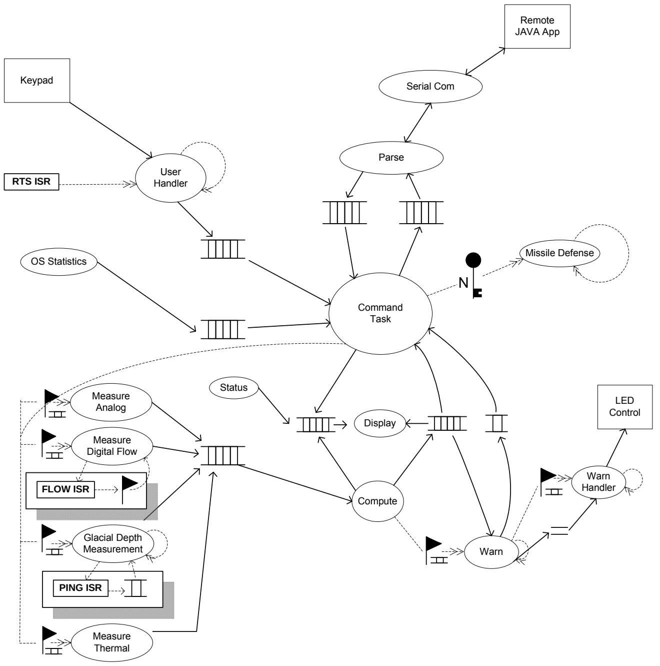


Figure 25: *Data flow diagram for the OS*

9. System Pseudo Code EnviroMonitor 188S

PSEUDO CODE
The following is a one page summary of the pseudo code for the EnviroMonitor Operating Software:

Measure Task

```
void measureTask(void* data){
  initialize ADC ports
  if(i = 0; i < 4; i++){
    if(i == 3)
        read port i;
    else
       read port i+1;
    store read into buffer
  }
}
```

Compute Task

```
void computeTask(void* data){
  //linearize data y = mx +b;
  CorrTemp = 5 + 0.8(Raw Temperature)
  CorrFlow = 9 + 2.0(Raw Flow Rate)
  CorrCarb = 7 +1.2(Raw Carbon Level)
  CorrSulf = 6 + 0.9(Raw Sulfur
Level)
}
```

Battery Status Task

```
void statusTask(void* data){
 Decrement battery state
}
```

Missile Defense Task

```
void defTask(void* data){
  if(user request)
    fire missile
}
```

Digital Flow Task

```
void digFlowTask(void* data){
  Initialize pio line
  read for 5 period
  average the read
  store read
}
```

Display Task

```
void displayTask(void* data)
{
   initialize lcd
 display corrected data
 clear screen
 display digital flow
 display glacial depth
 display signal frequency
}
```

Command Task

```
void comTask(void* data){
 if(warning)
   send data to serial
 if(alarm)
  signal DefenseTask;
 if(signal from parse)
   perform requested task
}
```

Glacial Depth Task

```
void glacialTask(void* data){
   pio_init()
   start timer2
   outportb(high)
   waith 1 ms
   outportb(low)
   waith for return ping
   //calculate depth
   Depth = v*time;
}
```

Warn Task

```
void warnTask(void* data){
 if(warning state)
   Flash LED
   Signal Serial
   if(Alarm state & !Ack)
     Turn on speaker
     Signal MissileDef
   else
      Speaker off
}
```

Parse Task

```
void parseTask(void* data){
  if(signal from serial)
     check body
  if(corrected)
     send request to command
  if(signal from command
     build message body
     send to serial
}
```

Thermal Task

```
void thermalTask(void* data){
 initialize ADC ports
 timer2 enable
 read ADC port //256 time
 timer2 disable

 calibrate read time
 calibrate Fs
  FFT the signal
   //calibrate signalFn
   Fn = Fs*m_index/256
}
```

User Handler Task

```
void userTask(void* data){
   read data from input
   process data
}
```

Java Interface

```
java{
  read user input
  build frame
  post to serial

  pend of serial
  length & checksum check
  if incorrect re-request
  if incorrect four time
     connection down
}
```

Serial Task

```
void serialTask(void* data){
  if(signal from Java)
     check for length and
checksum
     if (correct)
     send body to parse
  if(signal from parse)
     build frame
     send to java
}
```

SYSTEM SOURCE CODE

The following is a comprehensive presentation of the source code used in implementation of the EnviroMonitor system:

includes.h – Master Includes Header File

```
/***********************************************************************************************************
*                             MASTER INCLUDES FILE
*
* FILE:                  includes.h
* SOURCE:               accessed by all source files
* VERSION:              1.0
* PROJECT:              Glacial Monitoring System
* MODIFIED:             May 29 2008
* AUTHOR:               Justin Reina, Khoa Nguyen, Thuat Nguyen
***********************************************************************************************************/
#ifndef        STDLIBS
       #define        STDLIBS
   #include <stdio.h>
   #include <stdlib.h>
   #include <dos.h>
#endif

extern unsigned int myTempCt;

#ifndef               UCOS
       #define        UCOS
       #include "ucos.h"
#endif

#ifndef               TD40
       #define        TD40
   #include "td40.h"
#endif

#ifndef               MYTERN
       #define        MYTERN
   #include "myTern.h"
#endif

#ifndef               MAIN_DEF
       #define        MAIN_DEF
   #include "main.h"
#endif

 #ifndef              OS_FILES
       #define OS_FILES
       #include "OS_Files.h"
#endif

#ifndef               MAIN_TASKS
       #define MAIN_TASKS
   #include "mainTasks.h"
#endif

#ifndef               SEC_TASKS
       #define        SEC_TASKS
   #include "secondaryTasks.h"
#endif

#ifndef               USER_TASKS
       #define        USER_TASKS
   #include "userTasks.h"
#endif
```

Figure 26. Master Includes Header File

main.h – Main Source Header File

```
/***********************************************************************************************************
*                             MAIN SOURCE HEADER
*
* FILE:                 main.h
* SOURCE:              accessed by all source files
* VERSION:             1.0
* PROJECT:             Glacial Monitoring System
* MODIFIED:            May 29 2008
* AUTHOR:              Justin Reina, Khoa Nguyen, Thuat Nguyen
***********************************************************************************************************/
#ifndef UCOS
       #define        UCOS
   #include "ucos.h"
#endif

#ifndef TD40
       #define        TD40
   #include "td40.h"
#endif

#ifndef MAIN_TASKS
       #define MAIN_TASKS
   #include "mainTasks.h"
#endif

#ifndef SEC_TASKS
       #define SEC_TASKS
   #include "secondaryTasks.h"
#endif

#ifndef USER_TASKS
       #define USER_TASKS
   #include "userTasks.h"
#endif

//DEFINITIONS-----------------------------------------------------------------------------------------------
#define BASE_DEC      10             // radix
#define OUTPUT_MODE   2
#define INPUT_MODE    1
#define ON                    0
#define OFF                   1

//Alarm Definitions-----------------------------------------------------------------------------------------
#define PAUSE_DURATION         24
#define ONE_SEC_DURATION       85           //Equiv Count For One Sec In Interrupt
#define TWO_SEC_DURATION      170            //Equiv Count For Two Sec In Interrupt

#define TTL_OUTPUT_ADDRESS    0x0101
#define TTLCON                       0x0103

#define BUF_SIZE              8              //Data acquisition data buffer size

#define BAUD_RATE_9600        8
#define BUFFSIZE              2000
#define BRK_SIZE                    50

#define LED0_PIN              0x202          //LED pins
#define LED1_PIN              0x203
#define STATE0_PIN             0x204
#define STATE1_PIN             0x205
#define SPKR_PIN               0x201

#define RTS_PIN               13
#define CTS_PIN               0x206
#define DATA_PIN              3              //The pioRead uses the actual PIO Number(See user task)
#define DATA_PIN0             0
#define DATA_PIN1        5
#define DATA_PIN2        6
#define DATA_PIN3        14

#define PING_PIO_OUT     9
#define PING_PIO_IN      1
```

(main.h continue1/2)

(main.h continue 1/2)

```
#define DATA_IO_PIN     3    //This is only for personal use! Ignore it...

#define ADC_CH0         11   //ADC pins
#define ADC_CH1         11
#define ADC_CH2         11
#define ADC_CH3         11

#define DIG_PIO_IN    13   //Digital Flow Pin

#define PIO_ON  0            //PIO state
#define PIO_OFF 1

#define PIO_HIGH       0   //PIO state
#define PIO_LOW        1

//NEW OS PORTION-------------------------------------------------------------------------------------------
#define STACK_SIZE            1024                          // set the stack size
#define WAIT_FOREVER          0

#define STD_PERIOD 100

#define BAUD_RATE_19200       9
#define BUFFSIZE              2000

#define INBUF_DEPTH           20    //Buffer size for userHandler Task
#define INBUF_LENGTH          5

//FUNCTION PROTOTYPES--------------------------------------------------------------------------------------
void pioInit(void);
void timer2_Init(void);

void far interrupt Timer0_ISR(void);
extern void interrupt far timer2isr(void);
extern void interrupt far userISR(void);

void SetupTimers(void);
void far schedTask(void*);

void startOSSched(void);

extern int num_ms;

extern unsigned char c0;
extern unsigned char c11;
extern unsigned char c2;

//FCN PROTOTYPES-------------------------------------------------------------------------------------------
//Main Prototypes
void dataInit(void);

//Schedule Prototypes
void schedule(void *);

//HELPER FUNCTION PROTOTYPES
void indicatorInit(void);
void taskInit(void);
void interruptInit(void);
void interrupt far t2_isr (void);
void interrupt far Ext3_ISR(void);
void far interrupt pingRespond(void);
extern void serialOut(void*);

extern COM* c1;
extern COM ser1_com;

//VARIABLE PROTOTYPES--------------------------------------------------------------------------------------

//Annunciation Status Global Variables
extern unsigned int taskCounter;

void interrupt far t2_isr(void);

extern unsigned long pingCount, latchedCount;
```

(main.h continue 2/2)

```
extern unsigned int* tempBufPtr;
extern unsigned int* flowBufPtr;
extern unsigned int* carbonBufPtr;
extern unsigned int* sulfurBufPtr;
extern unsigned int  globalTicker;

extern unsigned int    digFlowConvRate;
extern unsigned long   digFlowCounts[7];
extern unsigned int    alarmAcknowledge;
```

Figure 27. Main Source Header File

main.c – Main Source File

```
/***********************************************************************************************************
*                      Main Source File
*
* FILE:               main.c
* HEADER:             main.h
* VERSION:            3.2
* PROJECT:            Glacial Monitoring System
* MODIFIED:           May 29 2008
* AUTHOR:             Justin Reina, Khoa Nguyen, Thuat Nguyen
***********************************************************************************************************/
#ifndef INCLUDES
    #define INCLUDES
    #include "includes.h"
#endif

MY_OS_Q*       debugQPtr;

/***********************************************************************************************************
*                     GLOBAL/STATIC VARIABLES
***********************************************************************************************************/

/***********************************************************************************************************
*                     RAW DATA BUFFERS
***********************************************************************************************************/

//3/X:RAW DATA BUFFERS-------------------------------------------------------------------------------------
unsigned int   tempRawBuf[BUF_SIZE];                // Temperature Buffer
unsigned int*  tempBufPtr = tempRawBuf;      // Temp Buffer Pointer

unsigned int   flowRawBuf[BUF_SIZE];         // Flow Rate Buffer
unsigned int*  flowBufPtr = flowRawBuf;      // Flow Buffer Pointer

unsigned int   carbonRawBuf[BUF_SIZE];       // Carbon level Buffer
unsigned int*  carbonBufPtr = carbonRawBuf;  // Carbon Level Buffer Pointer

unsigned int   sulfurRawBuf[BUF_SIZE];       // Sulfur Level Buffer
unsigned int*  sulfurBufPtr = sulfurRawBuf;  // Sulfur Level Buffer

unsigned int   glacialDepth = 205;
unsigned int   digitalFlow  = 300;

//4/X:CORRECTED DATA----------------------------------------------------------------------------------------
unsigned char  tempCorr[10];                        // Temperature Corrected
unsigned char  flowCorr[10];                        // Flow Rate Corrected
unsigned char  carbonCorr[10];                      // Carbon Level Corrected
unsigned char  sulfurCorr[10];                      // Sulfur Level Corrected

//5/X:STATUS DATA-------------------------------------------------------------------------------------------
unsigned char battState = BATT_CAPACITY;                                   // Battery State

//6/X:USER HANDLER DATA
unsigned short tempL0[2]      = {9,9}, tempL1[2]   = {9,9},
                  flowL0[2]   = {9,9}, flowL1[2]   = {9,9},
                  carbonL0[2] = {9,9}, carbonL1[2] = {9,9},
               sulfurL0[2]    = {9,9}, sulfurL1[2] = {9,9};

//USER INPUT BUFFER
unsigned short userInputBuf[INBUF_DEPTH][INBUF_LENGTH];
unsigned int   userInBufHead = 0, userInBufTail = 0;
```

(main.c continue 1/7)

```
//TIMER 2 DATA
unsigned long  pingCount          = 0,                  // 25ms Ticker in Timer2
                  latchedCount    = 0;                  // Latched For the Glacial Depth

//7/X:DIGITAL FLOW DATA
unsigned int   digFlowPeriod  = STD_DIG_FLOW_PERIOD;
unsigned int   digFlowConvRate = 999;
unsigned long   digFlowCounts[7] = {1,1,1,1,1,1,1};

//7/X:ADC DATA-----------------------------------------------------------------------------------------------
unsigned int   tempADCMin   = 0,      tempADCMax   = 100,         // Temp ADC Values
               flowADCMin   = 0,      flowADCMax   = 100,          // Flow ADC Values
               carbonADCMin = 0,      carbonADCMax = 100,          // Carbon ADC Values
               sulfurADCMin = 0,       sulfurADCMax = 100;          // Sulfur ADC Values

//8/10:COM PORT VARIABLES------------------------------------------------------------------------------------
unsigned char  ser1_in_buf[BUFFSIZE];                               // Serial Data In Buffer
unsigned char  ser1_out_buf[BUFFSIZE];                              // Serial Data Out Buffer

COM*           c1 = &ser1_com;                              // Pointer to the Tern COM Struct

//ALARM HANDLER DATA-----------------------------------------------------------------------------------------
unsigned int currState;
unsigned int currAlarmDuration;
unsigned int currInitDuration;

unsigned int alarmAcknowledge;        //He reads to see if alarm was acknowledged
unsigned int alarmCycleActive;        //He Writes to to indicate active alarm cycle

unsigned int newLEDType;
unsigned int newLEDState;
unsigned int newAlarmState;
unsigned int newAlarmDuration;
unsigned int newInitDuration;
unsigned int newInitCount;

unsigned int tempCount = 0;

//10/10:HELPER FUNCTION DATA---------------------------------------------------------------------------------
unsigned int   spkrState;
unsigned int   LEDFlashState;
unsigned int   scheduleData;
LED            LEDType;

//THERMAL DATA-----------------------------------------------------------------------------------------------
unsigned int thermalBuf[THERMAL_BUF_SIZE];
unsigned int thermalBufIX = 0 ;

OS_Q* myPQ;

//SERIAL COM DATA--------------------------------------------------------------------------------------------
unsigned char aRXBuf[BUFFSIZE];
unsigned char aTXBuf[BUFFSIZE];

/************************************************************************************************************
*                      Main Loop of Execution
*                      VOID MAIN(VOID)
*
* VERSION:             2.0
* PROJECT:             Glacial Monitoring System
* MODIFIED:            May 29 2008
* AUTHOR:              Justin Reina, Khoa Nguyen, Thuat Nguyen
************************************************************************************************************/

void main(void)
{
  /**********************************************************************************************************
                  TERN INITIALIZATIONS
  **********************************************************************************************************/
   ae_init();                                                  /* Tern Initialization     */
   lcd_init();                                                 /* LCD Initialization      */
   s1_init(BAUD_RATE_9600, ser1_in_buf, BUFFSIZE,              /* Serial COM port Initialization*/
                           ser1_out_buf,BUFFSIZE, c1);
```

(main.c continue 2/7)

```
/***********************************************************************************************************
                  UCOS INITIALIZATIONS
***********************************************************************************************************/
   OSInit();
                                                            /* Initialize uCOS                            */
   myOSQInit();                                             /* Initialize Mess Qs                          */

   setvect(uCOS, (void interrupt (*)(void))OSCtxSw);        /* Set The OS Interrupt                       */

/***********************************************************************************************************
                 SEMAPHORE INITIALIZATIONS
***********************************************************************************************************/
   timer2_Sem          = OSSemCreate(BINARY_SEM);           // Timer 2 Interrupt Semaphore
   msrMboxWrite_Sem           = OSSemCreate(BINARY_SEM);
   dispMboxWrite_Sem    = OSSemCreate(BINARY_SEM);
   parseOutWrite_Sem    = OSSemCreate(BINARY_SEM);

   missileLaunch_Sem    = OSSemCreate(MAX_MISSILE_RQSTS);
   missileLaunch_Sem->OSEventCnt = MAX_MISSILE_RQSTS;

/***********************************************************************************************************
                 MESSAGE BOX/QUEUE INITIALIZATIONS
***********************************************************************************************************/
   myOSEvenList_Init();

   msrAnlg_Sem        = OSSemCreate(BINARY_SEM);
   msrAnlg_Event      = myOSMboxCreate(&msrAnlg_Mbox);

   stat_Event         = myOSQCreate(&stat_MBox[0],OS_STAT_MESSQ_SIZE);/* OS Statistics
                                                                     Message Q     */
   measure_Event      = myOSQCreate(&measure_Mbox[0],MEASURE_MESSQ_SIZE);

   display_Event      = myOSQCreate(&display_Mbox[0],DISPLAY_MESSQ_SIZE);
   compute_Event      = myOSQCreate(&compute_Mbox[0],SYS_STATUS_MESSQ_SIZE);
   digFlowResp_Sem    = myOSMboxCreate(&digFlowResp_Mbox);
   glacialResponse_Event = myOSMboxCreate(&glacialResponse_Dbox);

   warnState_Event    = myOSMboxCreate(&warnState_Dbox);

   parseInbox_Event   = myOSMboxCreate(&parse_MInbox[0]);

   parseOutbox_Event  = myOSQCreate(&parse_MOutbox[0],PARSE_OUT_MESSQ_SIZE);

   serComIn_Event      = myOSQCreate(&serCom_MInbox[0],SER_IN_MESSQ_SIZE);
   serComOut_Event     = myOSQCreate(&serCom_MOutbox[0],SER_OUT_MESSQ_SIZE);

   debugQPtr           = display_Event->OSEventPtr;

   /********************************************************************************************************
                               POLL AND REQUEST INITIALIZATIONS
   *******************************************************************************************************/
   digFlow_Sem         = OSSemCreate(BINARY_SEM);
   digFlow_Event       = myOSMboxCreate(&digFlow_Mbox);

   glacial_Sem         = OSSemCreate(BINARY_SEM);
   glacial_Event       = myOSMboxCreate(&glacial_Mbox);

   acqFFT_Sem  = OSSemCreate(BINARY_SEM);
   acqFFT_Event        = myOSMboxCreate(&acqFFT_Mbox);

   warn_Sem    = OSSemCreate(BINARY_SEM);
   warn_Event  = myOSMboxCreate(&warn_Mbox);

   warnHndlr_Sem       = OSSemCreate(BINARY_SEM);
   warnHndlr_Event= myOSMboxCreate(&warnHndlr_Mbox);

   /********************************************************************************************************
                              OS START
    ******************************************************************************************************/
   startOSSched();                  /* Create The Schedule Task                                            */

   pioInit();                      /* Initializing the PIO Lines                                            */
   dataInit();                     /* Initialize Global Variables and Pointers                              */
```

(main.c continue 3/7)

```
   OSStart();                      /* Start the uC/OS                                      */
}
/************************************************************************************************************
*              PIO Initialization Routine
*
* VERSION:     2.0
* PROJECT:     Glacial Monitoring System
* MODIFIED:    May 29 2008
* AUTHOR:      Justin Reina, Khoa Nguyen, Thuat Nguyen
* COMMENTS:    This routine allows for consolidated initialization of the Tern PIO lines.This is to avoid
*              line assertion conflicts stemming from the Tern's resource sharing.
**************************************************************************************/
void pioInit()
{
   /*********************************************************************************************************
               ADC PORTS INITIALIZATION
   *********************************************************************************************************/
   pio_init(ADC_CH0_PIO,      INPUT_MODE);
   pio_init(ADC_CH1_PIO,      INPUT_MODE);
   pio_init(ADC_CH2_PIO,      INPUT_MODE);
   pio_init(ADC_CH3_PIO,      INPUT_MODE);

   ae_ad12(CH0);
   ae_ad12(CH1);
   ae_ad12(CH2);
   ae_ad12(CH3);
   /*********************************************************************************************************
                              USER COM PORTS INITIALIZATION
   *********************************************************************************************************/
   pio_init(RTS_PIN,  INPUT_MODE);
   pio_init(CTS_PIN,  OUTPUT_MODE);
   pio_init(DATA_PIN0,INPUT_MODE);
   pio_init(DATA_PIN1,INPUT_MODE);
   pio_init(DATA_PIN2,INPUT_MODE);
   pio_init(DATA_PIN3,INPUT_MODE);

   outportb(CTS_PIN, OFF);

   /*********************************************************************************************************
                               MEASUREMENT PORT
   *********************************************************************************************************/
   pio_init(PING_PIO_OUT,     OUTPUT_MODE);
   pio_init(PING_PIO_IN ,     INPUT_MODE);
   pio_init(DIG_PIO_IN,       INPUT_MODE);

   pio_wr(PING_PIO_OUT,OFF);                              /* Turn Off PING                                */

/************************************************************************************************************
                              LED STATE PORTS
************************************************************************************************************/
   outportb(LED0_PIN,PIO_OFF);                            /* LED to 'Green'                               */
   outportb(LED1_PIN,PIO_OFF);

   outportb(STATE0_PIN,PIO_OFF);                         /* LED State to 'Solid'                          */
   outportb(STATE1_PIN,PIO_OFF);

   outportb(SPKR_PIN,PIO_OFF);                          /* Speaker 'Off'                                  */
   pio_init(18,0);                                      /* Ported with magic numbers!                     */
}
/************************************************************************************************************
*                Global Variables Initialization Routine
*
* VERSION:       2.0
* PROJECT:       Glacial Monitoring System
* MODIFIED:      May 29 2008
* AUTHOR:        Justin Reina, Khoa Nguyen, Thuat Nguyen
* COMMENTS:      Consolidated location for all global variable initializations.
*                It can alsobe called by any routine/task in the event of a system error.
************************************************************************************************************/
/* All Values to Default Spec Unless Otherwise Specified */
void dataInit()
{
   MY_OS_Q* sysStatusPtr = compute_Event->OSEventPtr;
   /*********************************************************************************************************
```

(main.c continue 4/7)

```
                       (1/X)    MEASURE TASK DATA INITIALIZATION
    ********************************************************************************************************/
    measureData.ADC_Counts[0]         = 4;
    measureData.ADC_Counts[1]         = 9;
    measureData.ADC_Counts[2]         = 12;
    measureData.ADC_Counts[3]         = 13;

    measureData.sampleRate            = MSR_ANALOG_SAMPLE_RATE;
    measureData.msrAnlg_Event         = msrAnlg_Event;
    measureData.msrAnlg_Sem           = msrAnlg_Sem;
    measureData.msrMboxWrite_Sem      = msrMboxWrite_Sem;
    measureData.measure_Event         = measure_Event;
    /********************************************************************************************************
                        (2/X)    COMPUTE TASK DATA INITIALIZATION
    ********************************************************************************************************/
    computeData.tempBufPtr   = &tempBufPtr;                                                     // Raw data pointers
    computeData.flowBufPtr   = &flowBufPtr;
    computeData.carbonBufPtr = &carbonBufPtr;
    computeData.sulfurBufPtr = &sulfurBufPtr;

    computeData.battStatePtr = &battState;

    computeData.tempCorrPtr   = (unsigned char*)&tempCorr;                                      //corrected data pointers
    computeData.flowCorrPtr   = (unsigned char*)&flowCorr;
    computeData.carbonCorrPtr = (unsigned char*)&carbonCorr;
    computeData.sulfurCorrPtr = (unsigned char*)&sulfurCorr;

    computeData.tempADCMax    = &tempADCMax;
    computeData.tempADCMin    = &tempADCMin;
    computeData.flowADCMax    = &flowADCMax;
    computeData.flowADCMin    = &flowADCMin;
    computeData.sulfurADCMax  = &sulfurADCMax;
    computeData.sulfurADCMin  = &sulfurADCMin;
    computeData.carbonADCMax  = &carbonADCMax;
    computeData.carbonADCMin  = &carbonADCMin;

    computeData.measure_Event = measure_Event;
    computeData.dataLogging   = DATA_LOGGING_ON;
    /********************************************************************************************************
                        (3/X)   DISPLAY TASK DATA INITIALIZATION
    ********************************************************************************************************/
    displayData.tempCorrPtr   = tempCorr;                                  /* Pointer to corrected data         */
    displayData.flowCorrPtr   = flowCorr;
    displayData.carbonCorrPtr = carbonCorr;
    displayData.sulfurCorrPtr = sulfurCorr;
    displayData.battStatePtr  = &battState;

    displayData.display_Event         = display_Event;
    displayData.currSysStatusPtr      = &(sysStatusPtr->OSQCurr);

    /********************************************************************************************************
                        (4/X)   STATUS TASK DATA INITIALIZATION
    ********************************************************************************************************/
    statusData.battStatePtr    = &battState;
    statusData.battDrainPeriod = OS_TICKS_PER_SEC;
    statusData.minBattState    = 20;                                        //(2*BATT_CAPACITY)/10;

    /********************************************************************************************************
                        (5/X)   WARN TASK DATA INITIALIZATION
    ********************************************************************************************************/
    warnData.tempBufPtr   = &tempBufPtr;       /* All Data Initializations                                  */
    warnData.flowBufPtr   = &flowBufPtr;       /* Please Contact Engineering                               */
    warnData.carbonBufPtr = &carbonBufPtr;     /* Support Or Refer To Documentation                        */
    warnData.sulfurBufPtr = &sulfurBufPtr;     /* For Referenced Values.                                   */
    warnData.battStatePtr = &battState;

    warnData.tempL0   = tempL0;
    warnData.tempL1   = tempL1;
    warnData.flowL0   = flowL0;
    warnData.flowL1   = flowL1;
    warnData.carbonL0 = carbonL0;
    warnData.carbonL1 = carbonL1;
    warnData.sulfurL0 = sulfurL0;
    warnData.sulfurL1 = sulfurL1;
```

(main.c continue 5/7)

```
    warnData.tempOutRange   = 0;
    warnData.flowOutRange   = 0;
    warnData.carbonOutRange = 0;
    warnData.sulfurOutRange = 0;
    warnData.currSysStatusPtr = &(sysStatusPtr->OSQCurr);

    /********************************************************************************************************
                         (6/X)   USER HANDLER TASK DATA INITIALIZATION
    ********************************************************************************************************/
    userHandlerData.userInputBufPtr   = (unsigned short*)&userInputBuf;               /*Cast is from unShort*/
    userHandlerData.userInBufHeadPtr = &userInBufHead;                                /* UserHandler Buffer*/
    userHandlerData.userInBufTailPtr = &userInBufTail;

    userHandlerData.tempL0Ptr = (unsigned short*) &tempL0;                            /*Unser input limits*/
    userHandlerData.tempL1Ptr = (unsigned short*) &tempL1;
    userHandlerData.flowL0Ptr = (unsigned short*) &flowL0;
    userHandlerData.flowL1Ptr = (unsigned short*) &flowL1;
    userHandlerData.carbL0Ptr = (unsigned short*) &carbonL0;
    userHandlerData.carbL1Ptr = (unsigned short*) &carbonL1;
    userHandlerData.sulfL0Ptr = (unsigned short*) &sulfurL0;
    userHandlerData.sulfL1Ptr = (unsigned short*) &sulfurL1;

    userHandlerData.tempADCMinPtr = &tempADCMin;                                                 //ADC Limits
    userHandlerData.tempADCMaxPtr = &tempADCMax;
    userHandlerData.flowADCMinPtr = &flowADCMin;
    userHandlerData.flowADCMaxPtr = &flowADCMax;
    userHandlerData.carbADCMinPtr = &carbonADCMin;
    userHandlerData.carbADCMaxPtr = &carbonADCMax;
    userHandlerData.sulfADCMinPtr = &sulfurADCMin;
    userHandlerData.sulfADCMaxPtr = &sulfurADCMax;

    /********************************************************************************************************
                                (7/X)   ACQUIRE LIMITS TASK DATA INITIALIZATION
    ********************************************************************************************************/
    setLimitsData.userInputBufPtr  = (unsigned short*) &userInputBuf;
    setLimitsData.userInBufHeadPtr = &userInBufHead;
    setLimitsData.userInBufTailPtr = &userInBufTail;

    setLimitsData.tempL0Ptr  = (unsigned short*) &tempL0;
    setLimitsData.tempL1Ptr  = (unsigned short*) &tempL1;
    setLimitsData.flowL0Ptr  = (unsigned short*) &flowL0;
    setLimitsData.flowL1Ptr  = (unsigned short*) &flowL1;
    setLimitsData.carbL0Ptr  = (unsigned short*) &carbonL0;
    setLimitsData.carbL1Ptr  = (unsigned short*) &carbonL1;
    setLimitsData.sulfL0Ptr  = (unsigned short*) &sulfurL0;
    setLimitsData.sulfL1Ptr  = (unsigned short*) &sulfurL1;

    /********************************************************************************************************
                         (8/X)   SET ADC TASK DATA INITIALIZATION
    ********************************************************************************************************/
    setADCData.userInputBufPtr  = (unsigned short*) &userInputBuf;
    setADCData.userInBufHeadPtr = &userInBufHead;
    setADCData.userInBufTailPtr = &userInBufTail;

    setADCData.tempADCMinPtr = &tempADCMin;
    setADCData.tempADCMaxPtr = &tempADCMax;
    setADCData.flowADCMinPtr = &flowADCMin;
    setADCData.flowADCMaxPtr = &flowADCMax;
    setADCData.carbADCMinPtr = &carbonADCMin;
    setADCData.carbADCMaxPtr = &carbonADCMax;
    setADCData.sulfADCMinPtr = &sulfurADCMin;
    setADCData.sulfADCMaxPtr = &sulfurADCMax;

    /********************************************************************************************************
                         (9/X)   GLACIAL TASK DATA INITIALIZATION
    ********************************************************************************************************/
    glacialData.pingCount      = &pingCount;
    glacialData.latchedCount   = &latchedCount;
    glacialData.glacialDepth   = &glacialDepth;
    glacialData.glacialResponse_Event = glacialResponse_Event;

    /********************************************************************************************************
                        (10/X)   DIGITAL FLOW TASK DATA INITIALIZATION
```

(main.c continue 6/7)

```
   ********************************************************************************************************/
   digitalFlowData.digitalFlow               = &digitalFlow;
   digitalFlowData.digFlowPeriod           = &digFlowPeriod;
   digitalFlowData.digFlowConvRate         = &digFlowConvRate;
   digitalFlowData.latchedCount              = &latchedCount;

   digitalFlowData.digFlowErr                = OS_NO_ERR;
   digitalFlowData.sampleRate                = DIG_FLOW_SAMPLE_RATE;

   digitalFlowData.digFlow_Sem               = digFlow_Sem;
   digitalFlowData.digFlow_Event             = digFlow_Event;

   /********************************************************************************************************
                       (11/X)   ALARM HANDLER TASK DATA INITIALIZATION
   ********************************************************************************************************/
   alarmHandlerData.currState                = &currState;
   alarmHandlerData.currAlarmDuration        = &currAlarmDuration;
   alarmHandlerData.currInitDuration  = &currInitDuration;
   //FLAGS--------------------------------------------------------------------------------------------------
   alarmHandlerData.alarmAcknowledge  = &alarmAcknowledge; //He reads to see if alarm was ack              */
   alarmHandlerData.alarmCycleActive  = &alarmCycleActive; //He Writes to indactive alarm                  */
   //INPUTS-------------------------------------------------------------------------------------------------
   alarmHandlerData.newLEDType               = &newLEDType;
   alarmHandlerData.newLEDState              = &newLEDState;
   alarmHandlerData.newAlarmState            = &newAlarmState;
   alarmHandlerData.newAlarmDuration  = &newAlarmDuration;
   alarmHandlerData.newInitDuration          = &newInitDuration;
   alarmHandlerData.newInitCount             = &newInitCount;

   /********************************************************************************************************
                       (10/X)   MYOS STAT TASK DATA INITIALIZATION
   ********************************************************************************************************/
   myOSStatData.pevent                       = stat_Event;
   myOSStatData.stat_Dbox                    = (StatDbox*)&stat_Dbox;
   myOSStatData.MboxFull                     = OS_STAT_MBOX_NOT_FULL;

   /********************************************************************************************************
                       (11/X)   THERMAL TASK DATA INITIALIZATION
   ********************************************************************************************************/
   thermalData.thermalBufPtr         = &thermalBuf[0];
   thermalData.thermalBufIXPtr       = &thermalBufIX;
   thermalData.timerCtrPtr           = &pingCount;
   thermalData.acqFFT_Sem            = acqFFT_Sem;
   thermalData.timer2_Sem            = timer2_Sem;
   thermalData.acqFFT_Event          = acqFFT_Event;
   thermalData.measure_Event         = measure_Event;
   thermalData.msrMboxWrite_Sem  = msrMboxWrite_Sem;
   thermalData.sampleRate        = 2*OS_TICKS_PER_SEC;
   /********************************************************************************************************
                       (11/X)   COMMAND RESPONSE DATA INITIALIZATION
   ********************************************************************************************************/
   commandRespData.currSysStatus = &(sysStatusPtr->OSQCurr);
   commandRespData.dataLogging   = &computeData.dataLogging;

   /********************************************************************************************************
                       (11/X)   SERIAL COM DATA INITIALIZATION
   ********************************************************************************************************/
   serComData.tempCorrPtr   = tempCorr;                     /* Pointer to corrected data                  */
   serComData.flowCorrPtr   = flowCorr;
   serComData.carbonCorrPtr = carbonCorr;
   serComData.sulfurCorrPtr = sulfurCorr;

   serComData.rxBufPtr = aRXBuf;
   serComData.txBufPtr = aTXBuf;
}
```

Figure 28. Main Source File

(main.c continue 7/7)

os_file.h – OS Header File

```
/*****************************************************************************************************
*                      OS HEADER FILE
*
* FILE:                 os_files.h
* SOURCE:               os_files.c
* VERSION:              2.0
* PROJECT:              Glacial Monitoring System
* MODIFIED:             May 29 2008
* AUTHOR:               Justin Reina, Khoa Nguyen, Thuat Nguyen
* OUTLINE:             TBListed...
*****************************************************************************************************/

#define OS_TICKS_PER_SEC             200
#define OS_LOWEST_PRIO               63
#define MY_OS_MAX_QS                 20
#define MY_OS_STAT_Q                        0x04
#define MY_OS_STAT_RDY                      0x00
#define MY_OS_EVENT_TBL_SIZE  ((OS_LOWEST_PRIO)/8 + 1)
#define BINARY_SEM                          1
#define WINDMILL 0

#define MBOX_FULL       1
#define MBOX_NOT_FULL   0

#define Q_FULL          0                          /* These are backwards due to improper planning.     */
#define Q_NOT_FULL      1


#define                OS_FILES

#ifndef                INCLUDES
       #define         INCLUDES
   #include "includes.h"
#endif

extern UBYTE           nullErr;
extern unsigned int    myOS_ErrCount;

#define NO_TASK        0
#define NUM_TASKS 5
extern unsigned int   OSTaskCtrs[OS_LOWEST_PRIO+1];
extern unsigned char  myOSTaskPriorites[NUM_TASKS];

typedef struct my_os_q {
    struct my_os_q   *OSQPtr;        /* Link to next queue control block in list of free blocks           */
    void              **OSQStart;   /* Pointer to start of queue data                                      */
    void              **OSQEnd;     /* Pointer to end   of queue data                                      */
    void              **OSQIn;      /* Pointer to where next message will be inserted  in   the Q          */
    void              **OSQOut;     /* Pointer to where next message will be extracted from the Q          */
    UBYTE             OSQSize;      /* Size of queue (maximum number of entries)                           */
    UBYTE             OSQEntries;   /* Current number of entries in the queue                              */
    UWORD             maxMsgCt;
    UBYTE            queueIndex;
    UBYTE             queueOut;
    void              **OSQCurr;
} MY_OS_Q;

/*****************************************************************************************************
*                      Function Protoyopes - Program
*****************************************************************************************************/
void far schedTask(void*);
void startOSSched(void);

void timer0_Init(void);
void timer2_Init(void);

void far interrupt timer2isr(void);
void far interrupt Timer0_ISR(void);

void far interrupt    pingRespond(void);
void far interrupt    flowRespond(void);
void far interrupt    RTSInterrupt(void);
```

(os_file.h continue 1/10)

```
unsigned int myOSMboxPeek(OS_EVENT*);
unsigned int myOSQPeek(OS_EVENT*);

void myOSQInit(void);
void myOSEventWaitListInit(OS_EVENT*);
void myOSEventTaskRdy(OS_EVENT*, void*,UBYTE);
void myOSEventTaskWait(OS_EVENT*);
void myOSEventTO(OS_EVENT*);
void myOSUnMapTblInit(void);
void myOSEvenList_Init(void);
/***********************************************************************************************************
*                     Function Protoyopes - myUCOS
*
* VERSION:            5.0
* MODIFIED:           May 29 2008
***********************************************************************************************************/
OS_EVENT       *myOSMboxCreate(void* msg);                                                /* MBox Create  */
void           *myOSMboxPend(OS_EVENT *pevent, UWORD timeout, UBYTE *err);                 /* MBox Pend    */
UBYTE          *myOSMboxPost(OS_EVENT *pevent, void *msg);                                 /* MBox Post    */
void           *myOSMboxAccept(OS_EVENT*);                                                 /* MBox Accept  */
OS_EVENT       *myOSQCreate(void**, UWORD);
UBYTE          *myOSQPost(OS_EVENT*, void*);
void           *myOSQPend(OS_EVENT*, UWORD, UBYTE*);
UBYTE           myOSQPostOverFront(OS_EVENT*,void*);

/***********************************************************************************************************
*                     Task Priorities
*
* VERSION:            2.0
* MODIFIED:           May 29 2008
***********************************************************************************************************/
#define MISSILE_PRIORITY             4

#define SCHED_PRIORITY               5               /* Scheduler Priority                              */

#define ALARM_HANDLER_PRIORITY       6               /* User Task Priorities
        */
#define COMMAND_PRIORITY             7
#define SER_COM_PRIORITY             8
#define PARSE_PRIORITY               9

#define SERHANDLER_PRIORITY          10
#define ALARM_ACK_PRIORITY           11
#define GLACIAL_PRIORITY             12
#define SERBUF_PRIORITY              13
#define SETADC_PRIORITY              14
#define SETLIMITS_PRIORITY           15
#define SENDVALUES_PRIORITY          16

#define WARN_PRIORITY                17             /*Secondary Task Priorities                          */
#define DISPLAY_PRIORITY             18
#define COMPUTE_PRIORITY             19
#define USERHANDLER_PRIORITY         20

#define THERMAL_PRIORITY             21
#define MEASURE_PRIORITY             22             /* Main Task Priorities                             */
#define STATUS_PRIORITY              23
#define DIGITAL_FLOW_PRIORITY        24
#define JIMS_FACE_PRIORITY           25

/***********************************************************************************************************
*                             Error Handling
*
* TIMERS:                     Timer0,Timer2
* VERSION:                    1.0
* MODIFIED:                   May 29 2008
* COMMENTS:                   This was ported over from Lab 4. Residual CPU Timer Control
*                             Still Lingers in this section.
*
*                             Timer2 also has a binary semaphore (See 'Resource Semaphores')
***********************************************************************************************************/
#define MAX_ERROR_CODES       100

typedef enum {
```

(os_file.h continue 3/10)

```
               NO_ERR                                                       = 0,
               ERR_DAQ_SCHED_DECODE_USER_PROC = 1,
               ERR_DAQ_SCHED_DECODE_SER_PROC  = 2,
               ERR_OS_STAT_MBOX_TOO_FULL      = 3,
               ERR_MSR_ANLG_MESSAGE           = 4,
               ERR_MEASURE_AUTHOR_UNKNOWN     = 5,
               ERR_DISP_MBOX_DECODE_UNKNOWN   = 6,
               ERR_QUEUE_PEND                 = 7,
               ERR_DISP_MBOX_FULL             = 8,
               ERR_DIG_FLOW_TIMEOUT_ON_T2     = 9,
               ERR_DIG_MEAS_ZERO_CT           = 10,
               ERR_DIG_FLOW_MESSAGE           = 11,
               ERR_DIG_FLOW_TIMEOUT_ON_INT_RSP= 12,
               ERR_UNKNOWN_MISSILE_REQUEST    = 13,
               ERR_THERMAL_MEAS_NO_COUNT      = 14,
               ERR_COMPUTE_MBOX_FULL          = 15,
               ERR_THERMAL_T2_TIMEOUT         = 16,
               ERR_GLACIAL_PING_NO_RESPONSE   = 17,
               ERR_GLACIAL_T2_TIMEOUT         = 18,
               ERR_PEND_PARSE_OUT_TIMEOUT     = 19,
               ERR_COMMAND_MESSAGE_UNKNOWN    = 20,
               ERR_PARSE_UNKNOWN_COM_MESS     = 21,
               ERR_PARSE_ERROR_CODE_INBOX     = 22,
               ERR_PARSE_INIT_CODE_INBOX      = 23,
               WINDMILL_WINDMILL_WINDMILL     = 24,
               ERR_SER_COM_OUTBOX_FULL        = 25
               //...
               //...
               //...
               //...
               } MyOS_ErrCodes;

extern unsigned int   myOS_ErrCount;
extern MyOS_ErrCodes  myOS_ErrReport[MAX_ERROR_CODES];

/***********************************************************************************************************
*                             Timer Control
*
* TIMERS:                     Timer0,Timer2
* VERSION:                    1.0
* MODIFIED:                   May 29 2008
* COMMENTS:                   This was ported over from Lab 4. Residual CPU Timer Control
*                             Still Lingers in this section.
*
*                             Timer2 also has a binary semaphore (See 'Resource Semaphores')
**********************************************************************************************************/
#define ALL_TIMERS_UNMASK     0x0007                               /* All Timers                         */
#define TIMER_EOI             0x0008

#define TIMERS_UNMASK         0x0007
#define TIMERS_MASK           0x000F

#define TIMER2_VECT           0x004C                               /* Timer 2                            */
#define TCUCON                0xFF32
#define T2INTCON              0xFF3A
#define T2COMPA               0xFF62
#define TIMER2_COUNT          500
#define TIMER2_MASK           0x000F
#define TIMER2_UNMASK         0x0007

#define T2_WAIT_COUNT         (4*OS_TICKS_PER_SEC)

#define T0COMPA               0xFF52                               /* Timer 0 Page 126 (t8.1)             */
#define T0CON                 0xFF56                               /* Page 126 (t8.1)                     */
#define T0INTCON              0xFF32                               /* Page 116 (t7.5)                     */
#define TIMER0_MASK           0x4000
#define TIMER0_UNMASK         0xE001
#define TIMER0_COUNT          50000                                /* Count for timer0 to reset at       */

/***********************************************************************************************************
*                     Interrupts
*
* INTERRUPTS:         INT1, INT3, INT6
* VERSION:            1.0
```

(os_file.h continue 4/10)

```
* MODIFIED:            May 29 2008
***********************************************************************************************************/
#define NMI_VECTOR            0x0008                               /* All interrupts (Page 92, t7.1)      */
#define OS_APP_VECTOR         0x0081                               /* uC/OS Vector                        */
#define RISING_TRIGGER        1
#define ISR_EOI_REG           0xFF22
#define INT_RQST              0xFF2E

#define INT1CON               0xFF3A                               /* INT 1:                              */
#define INT1_EOI              0x000D
#define INT1_MASK             0x000F
#define INT1_UNMASK           0x0006

#define INT2CON               0xFF3C                               /* INT 2:                              */
#define INT2_EOI              0x000E
#define INT2_MASK             0x000F
#define INT2_UNMASK           0x0006

#define INT3CON               0xFF3E                               /* INT 3:                              */
#define INT3_EOI              0x000F
#define INT3_MASK             0x000F
#define INT3_UNMASK           0x0006

#define INT6CON               0xFF36                                /* INT 6:Page 116 (t7.5)              */
#define INT6_EOI              0x000B
#define INT6_MASK             0x000F
#define INT6_UNMASK           0x0006
#define INT6_VECTOR           0x002C                                /* Page 92 (t7.1)                    */

/***********************************************************************************************************
*                             Task Data Structures And Stacks
*
* TASKS:                      All
* VERSION:                    2.0
* MODIFIED:                   May 29 2008
***********************************************************************************************************/
#define STD_STACK_SIZE        800

//DATA STRUCTURES-------------------------------------------------------------------------------------------
/*extern MyOSStatData         myOSStatData;          !! See Bottom of File For Actual Line...             */

extern MeasureData            measureData;           /* Main Tasks                                        */
extern StatusData             statusData;
extern DigitalFlowData        digitalFlowData;
extern ThermalData            thermalData;

extern WarnData               warnData;              /* Secondary Tasks                                   */
extern ComputeData            computeData;
extern DisplayData            displayData;
extern AlarmAckData           alarmAckData;
extern AlarmHandlerData       alarmHandlerData;

extern UserHandlerData        userHandlerData;       /* User Tasks                                        */
extern GlacialData            glacialData;
extern SerBufData             serBufData;
extern SetADCData       setADCData;
extern SetLimitsData    setLimitsData;
extern SendValuesData    sendValuesData;
extern CommandRespData   commandRespData;
extern ParseData        parseData;


//STACKS----------------------------------------------------------------------------------------------------
extern UWORD missile_Stk             [STD_STACK_SIZE];
extern UWORD sched_Stk               [STD_STACK_SIZE];             /* Scheduler Task                     */
extern UWORD myOSStat_Stk            [STD_STACK_SIZE];

extern UWORD measure_Stk             [STD_STACK_SIZE];             /* Main Tasks                         */
extern UWORD status_Stk              [STD_STACK_SIZE];
extern UWORD digitalFlow_Stk          [STD_STACK_SIZE];
extern UWORD thermal_Stk             [STD_STACK_SIZE];

extern UWORD serHandler_Stk         [STD_STACK_SIZE];              /* Secondary Tasks                     */
extern UWORD compute_Stk             [STD_STACK_SIZE];
```

(os_file.h continue 5/10)

```
extern UWORD display_Stk             [STD_STACK_SIZE];
extern UWORD warn_Stk                [STD_STACK_SIZE];
extern UWORD alarmAck_Stk            [STD_STACK_SIZE];
extern UWORD alarmHandler_Stk         [STD_STACK_SIZE];

extern UWORD userHndlr_Stk           [STD_STACK_SIZE];              /* User Tasks                       */
extern UWORD glacial_Stk             [STD_STACK_SIZE];
extern UWORD serBuf_Stk              [STD_STACK_SIZE];
extern UWORD setADC_Stk              [STD_STACK_SIZE];
extern UWORD setLimits_Stk           [STD_STACK_SIZE];
extern UWORD sendValues_Stk           [STD_STACK_SIZE];
extern UWORD dispGlacial_Stk          [STD_STACK_SIZE];

extern UWORD commResp_Stk            [STD_STACK_SIZE];
extern UWORD serCom_Stk                     [STD_STACK_SIZE];
extern UWORD parse_Stk               [STD_STACK_SIZE];
extern UWORD jimsFace_Stk            [150];
/***********************************************************************************************************
*                     InterTask Communication
*
* OS_EVENTS:          Mailboxes, Message Queues, Binary Semaphores, Cntg Semaphores
* COMM REGIONS:       To Command, With Parse,       From Measure,  Missile,
*                     From Compute,   From Warn,     To Display
* VERSION:            1.0
* MODIFIED:           May 29 2008
***********************************************************************************************************/

//TO COMMAND:----------------------------------------------------------------------------------------------
#define USER_HNDLR_MESSQ_SIZE        20                                    /* User Handler Message Q    */
#define USER_HNDLR_FIELDS            3

typedef struct
{
   unsigned int               request;
   unsigned char              identifiers[USER_HNDLR_FIELDS];
   unsigned char              dataFields[USER_HNDLR_FIELDS];
} UserHndlrDBox;

extern OS_EVENT               *userHndlr_Event;
extern void*                  userHndlr_MBox[USER_HNDLR_MESSQ_SIZE];
extern UserHndlrDBox          userHndlr_DBox[USER_HNDLR_MESSQ_SIZE];

#define OS_STAT_MESSQ_SIZE    20                                           /* Statistics Task Message Q  */
#define NUM_STAT_TASKS        4
#define NUM_STAT_MESSAGE_Q    10
typedef struct
{
   unsigned long              statTimeStamp;
   unsigned long              cpuConsumption [NUM_STAT_TASKS];

   unsigned int               queueLengths   [NUM_STAT_MESSAGE_Q];
   unsigned int               maxQueueLengths[NUM_STAT_MESSAGE_Q];
   unsigned int               avgQueueWait   [NUM_STAT_MESSAGE_Q];
} StatDbox;

extern OS_EVENT               *stat_Event;
extern void*                  stat_MBox[OS_STAT_MESSQ_SIZE];
extern StatDbox               stat_Dbox[OS_STAT_MESSQ_SIZE];

//WITH PARSE:----------------------------------------------------------------------------------------------
#define PARSE_IN_MESSQ_SIZE   20
#define PARSE_OUT_MESSQ_SIZE  20

#define SYS_STAT_MESSQ_SIZE   10
#define WARN_OUT_MESSQ_SIZE   10


typedef struct
{
   unsigned char author;
   unsigned long timeStamp;

   unsigned char warnMsg;
```

(os_file.h continue 6/10)

```
   unsigned char tempCorr[4];
   unsigned char flowCorr[4];
   unsigned char carbonCorr[4];
   unsigned char sulfurCorr[4];

   unsigned char thermalCorr[4];
   unsigned char digFlowCorr[4];
   unsigned char glacialCorr[4];

   unsigned char battState[4];
} ParseOut_Msg;

typedef struct
{
   unsigned char sysState;
   unsigned long timeStamp;

   unsigned int  tempVal;
   unsigned int  flowVal;
   unsigned int  carbonVal;
   unsigned int  sulfurVal;
   unsigned int  digFlowVal;
   unsigned int  glacialVal;

   unsigned long statTimeStamp;
   unsigned long cpuConsumption[NUM_STAT_TASKS];

   unsigned int  queueLengths [NUM_STAT_MESSAGE_Q];
   unsigned int  maxQueueLengths[NUM_STAT_MESSAGE_Q];
   unsigned int  avgQueueWait [NUM_STAT_MESSAGE_Q];
} SysStat_Msg;

typedef struct
{
   unsigned int   sysState;
   unsigned long  timeStamp;

   unsigned int   tempVal;
   unsigned int   flowVal;
    unsigned int    carbonVal;
    unsigned int    sulfurVal;
} WarnOut_Msg;

extern OS_EVENT      *parseInbox_Event;
extern void*         parse_MInbox[PARSE_IN_MESSQ_SIZE];
extern unsigned char parse_DInbox[PARSE_IN_MESSQ_SIZE];

extern OS_EVENT      *parseOutbox_Event;
extern void*         parse_MOutbox[PARSE_OUT_MESSQ_SIZE];
extern ParseOut_Msg  parse_DOutbox[PARSE_OUT_MESSQ_SIZE];

extern SysStat_Msg   sysStat_Msg[SYS_STAT_MESSQ_SIZE];
extern unsigned int  sysStat_MsgHead, sysStat_MsgTail;

extern WarnOut_Msg   warnOut_Msg[WARN_OUT_MESSQ_SIZE];
extern unsigned int  warnOut_MsgHead, warnOut_MsgTail;

//FROM MEASURE:--------------------------------------------------------------------------------------------
#define MEASURE_MESSQ_SIZE    20
typedef struct
{
   unsigned short     author;
   unsigned long      timeStamp;

   unsigned int       tempRaw;
   unsigned int       flowRaw;
   unsigned int       carbonRaw;
   unsigned int       sulfurRaw;

   unsigned int       digFlowRaw;
   unsigned int       glacialRaw;
   unsigned int       thermalRaw;
} Measure_Msg;
```

(os_file.h continue 7/10)

```
extern OS_EVENT       *measure_Event;
extern void*          measure_Mbox[MEASURE_MESSQ_SIZE];
extern Measure_Msg     measure_Dbox[MEASURE_MESSQ_SIZE];

//FROM COMPUTE:---------------------------------------------------------------------------------------------
#define SYS_STATUS_MESSQ_SIZE 20

typedef struct compMsg
{
   unsigned long      timeStamp;

   unsigned int       tempVal;
   unsigned int       flowVal;
   unsigned int       carbonVal;
   unsigned int       sulfurVal;

   unsigned int       digFlowVal;
   unsigned int       glacialVal;
   unsigned int       thermalVal;

   unsigned char      tempCorr[5];
   unsigned char      flowCorr[5];
   unsigned char      carbonCorr[5];
   unsigned char      sulfurCorr[5];
   unsigned char      digFlowCorr[6];
   unsigned char      thermalCorr[6];
   unsigned char      glacialCorr[6];
   unsigned char      battState;
} Compute_Msg;

extern OS_EVENT       *compute_Event;
extern void*          compute_Mbox[SYS_STATUS_MESSQ_SIZE];
extern Compute_Msg     compute_Dbox[SYS_STATUS_MESSQ_SIZE];

//FROM WARN:------------------------------------------------------------------------------------------------
typedef struct
{
   unsigned int       sysState;
   unsigned long      timeStamp;

   unsigned int       tempVal;
   unsigned int       flowVal;
   unsigned int       carbonVal;
   unsigned int       sulfurVal;
} WarnState_Msg;

extern OS_EVENT       *warnState_Event;
extern void*          warnState_Mbox;
extern WarnState_Msg   warnState_Dbox;

//TO DISPLAY------------------------------------------------------------------------------------------------
#define DISPLAY_MESSQ_SIZE      5

extern OS_EVENT       *display_Event;
extern void*          display_Mbox[DISPLAY_MESSQ_SIZE];
extern unsigned int   display_Dbox[DISPLAY_MESSQ_SIZE];

//TO MISSILE------------------------------------------------------------------------------------------------
#define MISSILE_SUPPLY
extern OS_EVENT       *missileLaunch_Sem;

// FROM USERPROC--------------------------------------------------------------------------------------------
#define USERPROC_MESSQ_SIZE     20

extern OS_EVENT       *userProc_Event;
extern void*          userProc_MBox[USERPROC_MESSQ_SIZE];
extern unsigned int   userProc_DBox[USERPROC_MESSQ_SIZE];

// FROM SERIALPROC------------------------------------------------------------------------------------------
#define SERPROC_MESSQ_SIZE      20

extern OS_EVENT       *serialProc_Event;
extern void*          serialProc_MBox[SERPROC_MESSQ_SIZE];
extern unsigned int   serialProc_DBox[SERPROC_MESSQ_SIZE];
```

(os_file.h continue 8/10)

```
//TO FFT PROCESSOR------------------------------------------------------------------------------------------
#define RAW_THERMAL_SIZE      256
#define THERMAL_MESSQ_SIZE     3

typedef struct
{
   unsigned char rawThermal[RAW_THERMAL_SIZE];
   unsigned int timeStamp;
} Thermal_Msg;

extern OS_EVENT       *thermal_Event;
extern void*          thermal_MBox[THERMAL_MESSQ_SIZE];
extern Thermal_Msg    thermal_DBox[THERMAL_MESSQ_SIZE];

//TO GLACIAL TASK-------------------------------------------------------------------------------------------
extern OS_EVENT*      glacialResponse_Event;
extern void*          glacialResponse_Mbox;
extern unsigned long  glacialResponse_Dbox;

//FROM SERIAL COM-------------------------------------------------------------------------------------------
#define SER_IN_MESSQ_SIZE     20
extern OS_EVENT*              serComIn_Event;
extern void*                  serCom_MInbox[SER_IN_MESSQ_SIZE];
extern unsigned char         serCom_DInbox[SER_IN_MESSQ_SIZE];

//TO SERIAL COM---------------------------------------------------------------------------------------------
#define SER_OUT_MESSQ_SIZE 20
#define MAX_SER_MESSQ_SIZE 26
extern OS_EVENT*              serComOut_Event;
extern void*                  serCom_MOutbox[SER_OUT_MESSQ_SIZE];
extern unsigned char          serCom_DOutbox[SER_OUT_MESSQ_SIZE][MAX_SER_MESSQ_SIZE];

/***********************************************************************************************************
*                     Activity Semaphores
*                      (Poll & Request, 'P&R')
*
* OS_EVENTS:          Mailboxes, Binary Semaphores
* COMM REGIONS:       Measure/Digital/Glacial/Thermal/Warn/Warn Handler
* VERSION:            1.0
* MODIFIED:           May 29 2008
***********************************************************************************************************/
extern OS_EVENT       *msrAnlg_Sem;
extern OS_EVENT       *msrAnlg_Event;
extern unsigned int   msrAnlg_Mbox;

extern OS_EVENT       *digFlow_Sem;                                        /* Digital Flow P&R           */
extern OS_EVENT       *digFlow_Event;
extern unsigned int   digFlow_Mbox;
                                                                           /* Glacial Depth P&R       */
extern UBYTE          glacialErr;
extern OS_EVENT       *glacial_Sem;
extern OS_EVENT       *glacial_Event;
extern unsigned int   glacial_Mbox;

extern OS_EVENT       *acqFFT_Sem;                                          /* Digital Flow P&R          */
extern OS_EVENT       *acqFFT_Event;
extern unsigned int   acqFFT_Mbox;

extern OS_EVENT       *warn_Sem;                                          /* Warn Task P&R               */
extern OS_EVENT       *warn_Event;
extern unsigned int   warn_Mbox;

extern OS_EVENT       *warnHndlr_Sem;
extern OS_EVENT*      warnHndlr_Event;                                     /* Warn Handler P&R           */
extern unsigned int   warnHndlr_Mbox;

/***********************************************************************************************************
*                     Resource Semaphores
*
* OS_EVENTS:          Binary Semaphores
* COMM REGIONS:       Glacial/Digital Flow
* VERSION:                    1.0
* MODIFIED:                   May 29 2008
```

(os_file.h continue 9/10)

```
***********************************************************************************************************/
extern OS_EVENT       *timer2_Sem;                  /* Holds Timer 2                             */
extern OS_EVENT*      msrMboxWrite_Sem;
extern OS_EVENT*      dispMboxWrite_Sem;
extern OS_EVENT*      parseOutWrite_Sem;

extern OS_EVENT       *digFlowResp_Sem;
extern void*          digFlowResp_Mbox;

/***********************************************************************************************************
*                     myOSStatTask
*
* VERSION:            5.0
* MODIFIED:           May 29 2008
***********************************************************************************************************/
#define OS_STAT_MBOX_FULL     1
#define OS_STAT_MBOX_NOT_FULL 0
#define OS_STAT_PERIOD 2*OS_TICKS_PER_SEC
typedef struct
{
   OS_EVENT*          pevent;
   StatDbox*          stat_Dbox;
   unsigned int       MboxFull;
} MyOSStatData;

extern MyOSStatData   myOSStatData;
void far myOSStatTask(void*);
```

Figure 29. OS Header File

os_files.c – OS Source File

```
/***********************************************************************************************************
*                     OS Source File
*
* FILE:               os_files.c
* HEADER:             os_files.h
* VERSION:            2.0
* PROJECT:            Glacial Monitoring System
* MODIFIED:           May 29 2008
* AUTHOR:             Justin Reina, Khoa Nguyen, Thuat Nguyen
***********************************************************************************************************/

#ifndef        INCLUDES
   #define     INCLUDES
   #include    "includes.h"
#endif

unsigned int   OSTaskCtrs[OS_LOWEST_PRIO+1];

unsigned char  myOSTaskPriorites[NUM_TASKS] = {NO_TASK,
                                 MEASURE_PRIORITY,
                                 STATUS_PRIORITY,
                                 DIGITAL_FLOW_PRIORITY,
                                 OS_LOWEST_PRIO
                                 };

MY_OS_Q* justinPtr;

/***********************************************************************************************************
*                     Global/Static Variables
*
* VERSION:            2.0
* PROJECT:            Glacial Monitoring System
* MODIFIED:           May 29 2008
* AUTHOR:             Justin Reina, Khoa Nguyen, Thuat Nguyen
***********************************************************************************************************/

/***********************************************************************************************************
*                                    TASK OS DATA STRUCTURES
***********************************************************************************************************/
int                   silly_Khoa[4];
MyOSStatData          myOSStatData;                            /* OS/myOS Tasks                       */
```

(os_file.h continue 10/10)

```
MeasureData            measureData;                              /* Main Tasks                        */
StatusData             statusData;
DigitalFlowData        digitalFlowData;
ThermalData            thermalData;

WarnData               warnData;
AlarmAckData           alarmAckData;
AlarmHandlerData        alarmHandlerData;

ComputeData            computeData;                               /* Secondary Tasks                  */
DisplayData            displayData;

UserHandlerData        userHandlerData;                           /* User Tasks                       */
GlacialData            glacialData;
SerBufData             serBufData;
SetADCData             setADCData;
SetLimitsData          setLimitsData;
SendValuesData         sendValuesData;

CommandRespData   commandRespData;                                /* Ser Com Tasks                    */
SerComData        serComData;
ParseData         parseData;


/*******************************************************************************************************************
*                                           TASK OS STACKS
*******************************************************************************************************************/
UWORD missile_Stk             [STD_STACK_SIZE];
UWORD sched_Stk               [STD_STACK_SIZE];      /* Schedule Stack                                  */
UWORD   myOSStat_Stk          [STD_STACK_SIZE];

UWORD measure_Stk             [STD_STACK_SIZE];       /* Main Tasks                                     */
UWORD status_Stk              [STD_STACK_SIZE];
UWORD digitalFlow_Stk          [STD_STACK_SIZE];
UWORD thermal_Stk             [STD_STACK_SIZE];

UWORD serHandler_Stk          [STD_STACK_SIZE];        /* Secondary Tasks                              */
UWORD compute_Stk             [STD_STACK_SIZE];
UWORD display_Stk             [STD_STACK_SIZE];
UWORD warn_Stk                [STD_STACK_SIZE];
UWORD alarmAck_Stk            [STD_STACK_SIZE];
UWORD alarmHandler_Stk         [STD_STACK_SIZE];

UWORD userHndlr_Stk           [STD_STACK_SIZE];        /* User Tasks                                   */
UWORD glacial_Stk             [STD_STACK_SIZE];
UWORD serBuf_Stk              [STD_STACK_SIZE];
UWORD setADC_Stk              [STD_STACK_SIZE];
UWORD setLimits_Stk           [STD_STACK_SIZE];
UWORD sendValues_Stk          [STD_STACK_SIZE];
UWORD dispGlacial_Stk          [STD_STACK_SIZE];

UWORD commResp_Stk            [STD_STACK_SIZE];       /* Serial Com Tasks                              */
UWORD serCom_Stk              [STD_STACK_SIZE];
UWORD parse_Stk               [STD_STACK_SIZE];
UWORD jimsFace_Stk            [150];
/*******************************************************************************************************************
*                               INTERTASK COMMUNICATION MECHANISMS
*******************************************************************************************************************/
OS_EVENT       *timer2_Sem;
UBYTE          nullErr;
unsigned int   myOS_ErrCount = 0;
MyOS_ErrCodes  myOS_ErrReport [MAX_ERROR_CODES];

// TO COMMAND------------------------------------------------------------------------------------------
//1. User Handler Message Queue
OS_EVENT       *userHndlr_Event;
void*          userHndlr_MBox [USER_HNDLR_MESSQ_SIZE];
UserHndlrDBox  userHndlr_DBox [USER_HNDLR_MESSQ_SIZE];

//2. Statistics Task Message Queue
OS_EVENT       *stat_Event;
void*          stat_MBox      [OS_STAT_MESSQ_SIZE];
StatDbox       stat_Dbox      [OS_STAT_MESSQ_SIZE];
```

(os_file.c continue 1/13)

```
// WITH PARSE-----------------------------------------------------------------------------------------------
OS_EVENT       *parseInbox_Event;
void*          parse_MInbox   [PARSE_IN_MESSQ_SIZE];
unsigned char  parse_DInbox   [PARSE_IN_MESSQ_SIZE];

OS_EVENT       *parseOutbox_Event;
void*          parse_MOutbox  [PARSE_OUT_MESSQ_SIZE];
ParseOut_Msg   parse_DOutbox  [PARSE_OUT_MESSQ_SIZE];

SysStat_Msg    sysStat_Msg    [SYS_STAT_MESSQ_SIZE];        /* Maybe Deprecated                           */
unsigned int   sysStat_MsgHead, sysStat_MsgTail;

WarnOut_Msg    warnOut_Msg    [WARN_OUT_MESSQ_SIZE];
unsigned int   warnOut_MsgHead, warnOut_MsgTail;

// TO DISPLAY-----------------------------------------------------------------------------------------------
OS_EVENT       *display_Event;
void*          display_Mbox[DISPLAY_MESSQ_SIZE];
unsigned int   display_Dbox[DISPLAY_MESSQ_SIZE];

// FROM MEASURE---------------------------------------------------------------------------------------------
OS_EVENT       *measure_Event;
void*          measure_Mbox   [MEASURE_MESSQ_SIZE];
Measure_Msg    measure_Dbox   [MEASURE_MESSQ_SIZE];

// FROM COMPUTE---------------------------------------------------------------------------------------------
OS_EVENT       *compute_Event;
void*          compute_Mbox   [SYS_STATUS_MESSQ_SIZE];
Compute_Msg    compute_Dbox   [SYS_STATUS_MESSQ_SIZE];

// FROM WARN------------------------------------------------------------------------------------------------
OS_EVENT       *warnState_Event;
void*          warnState_Mbox;
WarnState_Msg  warnState_Dbox;

//TO MISSILE------------------------------------------------------------------------------------------------
OS_EVENT       *missileLaunch_Sem;

//FROM USER PROC--------------------------------------------------------------------------------------------
OS_EVENT       *userProc_Event;
void*          userProc_MBox  [USERPROC_MESSQ_SIZE];
unsigned int   userProc_DBox  [USERPROC_MESSQ_SIZE];

//FROM SERIAL PROCESSOR-------------------------------------------------------------------------------------
OS_EVENT       *serialProc_Event;
void*          serialProc_MBox[SERPROC_MESSQ_SIZE];
unsigned int   serialProc_DBox[SERPROC_MESSQ_SIZE];

//TO FFT PROCESSOR------------------------------------------------------------------------------------------
OS_EVENT       *thermal_Event;
void*          thermal_MBox   [THERMAL_MESSQ_SIZE];
Thermal_Msg    thermal_DBox   [THERMAL_MESSQ_SIZE];

//TO GLACIAL TASK-------------------------------------------------------------------------------------------
OS_EVENT*      glacialResponse_Event;
void*          glacialResponse_Mbox;
unsigned long  glacialResponse_Dbox;

//FROM SERIAL COM-------------------------------------------------------------------------------------------

OS_EVENT*      serComIn_Event;
void*          serCom_MInbox[SER_IN_MESSQ_SIZE];
unsigned char  serCom_DInbox[SER_IN_MESSQ_SIZE];

//TO SERIAL COM---------------------------------------------------------------------------------------------
OS_EVENT*      serComOut_Event;
void*          serCom_MOutbox[SER_OUT_MESSQ_SIZE];
unsigned char  serCom_DOutbox[SER_OUT_MESSQ_SIZE][MAX_SER_MESSQ_SIZE];

/***********************************************************************************************************
*                              POLL AND REQUEST COMMUNICATION MECHANISMS
***********************************************************************************************************/
OS_EVENT       *msrAnlg_Sem;
OS_EVENT       *msrAnlg_Event;
```

(os_file.c continue 2/13)

```
unsigned int   msrAnlg_Mbox;

OS_EVENT       *digFlow_Sem;                     /* Digital Flow P&R                                      */
OS_EVENT       *digFlow_Event;
unsigned int   digFlow_Mbox;
                                                 /* Glacial Depth P&R                                     */
UBYTE          glacialErr;
OS_EVENT       *glacial_Sem;
OS_EVENT       *glacial_Event;
unsigned int   glacial_Mbox;

OS_EVENT       *acqFFT_Sem;                       /* Digital Flow P&R                                      */
OS_EVENT                      *acqFFT_Event;
unsigned int   acqFFT_Mbox;

OS_EVENT       *warn_Sem;                         /* Warn Task P&R                                         */
OS_EVENT       *warn_Event;
unsigned int   warn_Mbox;

OS_EVENT       *warnHndlr_Sem;
OS_EVENT*      warnHndlr_Event;                   /* Warn Handler P&R                                     */
unsigned int   warnHndlr_Mbox;

/***********************************************************************************************************
*                                            BINARY SEMAPHORES
***********************************************************************************************************/
OS_EVENT*              msrMboxWrite_Sem;
OS_EVENT*              dispMboxWrite_Sem;
OS_EVENT*              parseOutWrite_Sem;

OS_EVENT               *digFlowResp_Sem;
void*                  digFlowResp_Mbox;

/***********************************************************************************************************
*                                    ADDITIONAL QUEUE STRUCTURES
***********************************************************************************************************/
MY_OS_Q                       myOSQTbl[MY_OS_MAX_QS];
MY_OS_Q*                      myOSQFreeList = &myOSQTbl[0];

/***********************************************************************************************************
*                      Schedule Task
*
* VERSION:             2.0
* PROJECT:             Glacial Monitoring System
* MODIFIED:            May 29 2008
* AUTHOR:              Justin Reina, Khoa Nguyen, Thuat Nguyen
***********************************************************************************************************/
void far schedTask(void* data)
{
   data = data;
   OS_ENTER_CRITICAL();
   /********************************************************************************************************
   *                            [1/3]   TIMER INITIALIZATION
   ********************************************************************************************************/
   //   Install the uC/OS timer ISR with the NMI Priority
   setvect(NMI_VECTOR, (void interrupt (*)(void))OSTickISR);

   //   Install OS Application Timer ISR
   setvect(OS_APP_VECTOR, Timer0_ISR);

   timer0_Init();                                          /* Timer0 is the OS Timer. (Unmasked)        */
   timer2_Init();                                          /* Timer2 is set for a 25ms ISR. (Masked)    */

   /********************************************************************************************************
   *                            [2/3]   INTERRUPT INITIALIZATION
   ********************************************************************************************************/
   int3_init(RISING_TRIGGER,pingRespond);                  // Ext Int3 is for Glacial Depth
   outportb(INT3CON,INT3_MASK);                            // pingRespond  - Masked
   //   int6_init(1,RTSInterrupt);                          // Ext Int 6 is the RTS Line*/
   /********************************************************************************************************
   *                             [3/3]     MAIN TASK CREATION
   ********************************************************************************************************/
   OSTaskCreate(myOSStatTask, (void*) &myOSStatData, (void*)&myOSStat_Stk[STACK_SIZE], OS_LOWEST_PRIO-1);
   OSTaskCreate(missileDefenseTask,(void*)&silly_Khoa[WINDMILL],
```

(os_file.c continue 4/13)

```
               (void*)&missile_Stk[STACK_SIZE], MISSILE_PRIORITY);

   OSTaskCreate(jimsFaceTask,(void*) 0,(void*)&jimsFace_Stk [150], JIMS_FACE_PRIORITY);

   //Main Tasks (3)
   OSTaskCreate(measureTask,(void*) &measureData,(void*)&measure_Stk[STACK_SIZE], MEASURE_PRIORITY);
   OSTaskCreate(statusTask, (void*) &statusData, (void*)&status_Stk [STACK_SIZE], STATUS_PRIORITY);
   OSTaskCreate(digitalFlowTask,(void*) &digitalFlowData,
               (void*)&digitalFlow_Stk[STACK_SIZE], DIGITAL_FLOW_PRIORITY);

   OSTaskCreate(thermalTask,(void*) &thermalData,(void*)&thermal_Stk[STACK_SIZE], THERMAL_PRIORITY);

   /********************************************************************************************************
   *                             [4/4] COM COMMUNICATION CREATION
   ********************************************************************************************************/
   OSTaskCreate(commandRespTask,(void*)&commandRespData,
                (void*)&commResp_Stk[STD_STACK_SIZE],COMMAND_PRIORITY);

   OSTaskCreate(serComTask,(void*)&serComData,(void*)&serCom_Stk[STD_STACK_SIZE], SER_COM_PRIORITY);
   OSTaskCreate(parseTask,(void*)&parseData, (void*)&parse_Stk[STD_STACK_SIZE], PARSE_PRIORITY);

   /********************************************************************************************************
   *                             [3/3]    SECONDARY TASK CREATION
   ********************************************************************************************************/
   OSTaskCreate(warnTask,(void*) &warnData,(void*)&warn_Stk [STACK_SIZE], WARN_PRIORITY);
   OSTaskCreate(alarmAckTask,(void*) &alarmAckData,(void*)&alarmAck_Stk[STACK_SIZE], ALARM_ACK_PRIORITY);
      //OSTaskSuspend(ALARM_ACK_PRIORITY);
   OSTaskCreate(alarmHandlerTask,(void*) &alarmHandlerData,
                (void*)&alarmHandler_Stk[STACK_SIZE], ALARM_HANDLER_PRIORITY);
      //OSTaskSuspend(ALARM_HANDLER_PRIORITY);
   OSTaskCreate(computeTask,(void*) &computeData,(void*)&compute_Stk[STACK_SIZE], COMPUTE_PRIORITY);
   OSTaskCreate(displayTask,(void*) &displayData, (void*)&display_Stk[STACK_SIZE], DISPLAY_PRIORITY);


   /********************************************************************************************************
   *                             [3/3]    USER TASK CREATION
   ********************************************************************************************************/
   OSTaskCreate(userHandlerTask, (void*) &userHandlerData,
               (void*)&userHndlr_Stk[STACK_SIZE], USERHANDLER_PRIORITY);
       OSTaskSuspend(USERHANDLER_PRIORITY);

   OSTaskCreate(glacialTask,(void*) &glacialData,(void*)&glacial_Stk[STACK_SIZE], GLACIAL_PRIORITY);
       OSTaskSuspend(GLACIAL_PRIORITY);

   OSTaskCreate(serBufTask,(void*) &serBufData,(void*)&serBuf_Stk[STACK_SIZE], SERBUF_PRIORITY);
       OSTaskSuspend(SERBUF_PRIORITY);

   OSTaskCreate(setADCTask,(void*) &setADCData,(void*)&setADC_Stk[STACK_SIZE], SETADC_PRIORITY);
       OSTaskSuspend(SETADC_PRIORITY);

   OSTaskCreate(setLimitsTask,(void*) &setLimitsData,(void*)&setLimits_Stk[STACK_SIZE],SETLIMITS_PRIORITY);
       OSTaskSuspend(SETLIMITS_PRIORITY);

   OSTaskCreate(sendValuesTask,(void*) &sendValuesData,
                (void*)&sendValues_Stk[STACK_SIZE],SENDVALUES_PRIORITY);
       OSTaskSuspend(SENDVALUES_PRIORITY);

   OS_EXIT_CRITICAL();
   OSTaskDel(OS_PRIO_SELF);                          /* Delete Itself                                    */
}

/***********************************************************************************************************
*                      OS Statistics Task
*
* VERSION:             1.0
* PROJECT:             Glacial Monitoring System
* MODIFIED:            May 29 2008
* AUTHOR:              Justin Reina
***********************************************************************************************************/
#define OS_TICKS_PER_SEC 200
void far myOSStatTask(void* data)
{
 unsigned int  i = 0,  cpuTotal  = 0, cpuTotal2 = 0;
 unsigned long lastStatPost     = 0;
```

(os_file.c continue 5/13)

```
 MyOSStatData* myVars           = data;
 OS_EVENT*     pevent            = myVars->pevent;
 MY_OS_Q*      pq               = pevent->OSEventPtr;

 StatDbox*     myStatDbox;

 unsigned long cpuConsumption[NUM_TASKS+1];
 for(;;)
 {
   OS_ENTER_CRITICAL();
   /*********************************************************************************************************
   *                             [1/3]   TASK CPU CONSUMPTION
   ********************************************************************************************************/
   myStatDbox  = (myVars->stat_Dbox+ (int)pq->queueIndex);
   myStatDbox->statTimeStamp = OSTimeGet();

   cpuTotal  = 0;
   cpuTotal2 = 0;
   for(i = 0; i <= NUM_TASKS;i++)
   {
      cpuConsumption[i] = OSTaskCtrs[myOSTaskPriorites[i+1]];
      cpuTotal+= (unsigned int) cpuConsumption[i];
      OSTaskCtrs[myOSTaskPriorites[i+1]] = 0;
   }

   for(i = 0; i <= NUM_STAT_TASKS;i++)
   {
       cpuConsumption[i] *= 200;
      if(cpuTotal)
        cpuConsumption[i] /= cpuTotal;
      *(myStatDbox->cpuConsumption+i) =   cpuConsumption[i];
      cpuTotal2+= (unsigned int) cpuConsumption[i];
   }

   if(cpuTotal2!=200)
       cpuConsumption[3] += 200-cpuTotal2;

   /*********************************************************************************************************
   *                             [2/3]   MESSAGE QUEUE STATISTICS
   ********************************************************************************************************/
   for(i=0;i<NUM_STAT_MESSAGE_Q;i++)
   {
         *(myStatDbox->queueLengths + i) = 10;
       *(myStatDbox->maxQueueLengths + i) = 12;
       *(myStatDbox->avgQueueWait + i) = 13;
   }

   /*********************************************************************************************************
   *                             [3/3]   POST TO MESSAGE QUEUE AND LEAVE
   ********************************************************************************************************/
   if(!myOSQPeek(stat_Event))                /* Peek                                                     */
      //myOS_ErrReport[myOS_ErrCount++] = ERR_DISP_MBOX_FULL;
      delay_ms(50);
   else if((OSTimeGet() - lastStatPost) > (OS_TICKS_PER_SEC>>1))
   {
      lastStatPost = OSTimeGet();
      myOSQPost(stat_Event, &stat_Dbox[pq->queueIndex]);    /* Post Message                              */
   }
   OS_EXIT_CRITICAL();
   OSTimeDly(OS_STAT_PERIOD);
 }
}
/************************************************************************************************************
                         OS Task Create Function to Clean Up Main Function
***********************************************************************************************************/
void startOSSched()

{
   OSTaskCreate(schedTask, (void*)0, (void*)&sched_Stk[STACK_SIZE-1], SCHED_PRIORITY);
}

/************************************************************************************************************
*                      Timer Functions
*
```

(os_file.c continue 6/13)

```
* VERSION:              1.0
* PROJECT:              Glacial Monitoring System
* MODIFIED:             May 29 2008
* AUTHOR:               Justin Reina
***********************************************************************************************************/

/***********************************************************************************************************
*                              TIMER 0
************************************************************************************************************
/
void timer0_Init(void)
{
   outport(T0INTCON, ALL_TIMERS_UNMASK);             // Unmask Timer0 interrupt and set Timer0
   outport(T0CON,TIMER0_MASK);                      // Disable TIMER0
   outport(T0COMPA,(int)50000L);                     // Initialize TIMER0's count (COMPA is the
   outport(T0CON,TIMER0_UNMASK);                     // Turn Timer0 Back On
}

void far interrupt Timer0_ISR()
{
   OSTaskCtrs[OSTCBCur->OSTCBPrio]++;
   outport(ISR_EOI_REG,TIMER_EOI);                   // Timer EOI
}

/***********************************************************************************************************
*                              TIMER 2
***********************************************************************************************************/
void timer2_Init(void)
{
   *(void far* far*)(TIMER2_VECT) = timer2isr;
   outport(TCUCON,     ALL_TIMERS_UNMASK);          //Unmask all Timer Interrupts
   outport(0xFF66, 0x4000);                          //Mask The Timer 2 Interrupt
   outport(T2COMPA,    TIMER2_COUNT);               //Setup T2CMPA to count 10000
   outport(0xFF66, 0xE001);                         //Unmask The Timer 2 Interrupt
}

void far interrupt timer2isr()
{
   pingCount++;
   outportb(ISR_EOI_REG,TIMER_EOI);
}

/***********************************************************************************************************
*                       External Interrupt Conigurations
*
* VERSION:              1.0
* PROJECT:              Glacial Monitoring System
* MODIFIED:             May 29 2008
* AUTHOR:               Justin Reina
***********************************************************************************************************/


/***********************************************************************************************************
*                          INT 1
***********************************************************************************************************/
void far interrupt flowRespond()
{
        static int ISRCount = 0;

   OSIntEnter();                                     //Resume digitalFlowTask
   digFlowCounts[ISRCount] = pingCount;             //Latch The Counter

   if(ISRCount == 5)
   {
        OSTaskResume(DIGITAL_FLOW_PRIORITY);
        outportb(INT1CON,INT1_MASK);
   }

   ISRCount = (ISRCount+1)%6;
   pingCount = 0;                                   //Reset The Counter

   outportb(ISR_EOI_REG,INT1_EOI);      //
   OSIntExit();                         //
}
```

(os_file.c continue 7/13)

```
/***********************************************************************************************************
*                                            INT 2
***********************************************************************************************************/
//NOT USED

/**********************************************************************************************************
*                                            INT 3
***********************************************************************************************************/
void far interrupt pingRespond()
{
   OSIntEnter();
   latchedCount= pingCount;                            //Acquire The Value
   myOSMboxPost(glacialResponse_Event,&latchedCount);  //Resume Task
   outportb(ISR_EOI_REG,INT3_EOI);                     //Signal The End of the Interrupt
   OSIntExit();                                        //Exit The Interrupt
}

/***********************************************************************************************************
*                                            INT 4
************************************************************************************************************
/
//NOT USED

/***********************************************************************************************************
*
*                                            INT 6
***********************************************************************************************************/
void far interrupt RTSInterrupt()
{
   outportb(INT6CON,INT6_MASK);
   OSIntEnter();
   OSTaskResume(USERHANDLER_PRIORITY);
   outportb(0xff22,INT6_EOI);
   OSIntExit();
}

/***********************************************************************************************************
*                     MAILBOX FEATURES
*                (myOSMboxCreate, myOSMboxPost, myOSMboxPend, myOSMboxAccept, myOSMboxPeek)
*
* VERSION:             1.0
* PROJECT:             Glacial Monitoring System
* MODIFIED:            May 29 2008
* AUTHOR:              Justin Reina
***********************************************************************************************************/
OS_EVENT *myOSMboxCreate(void* msg)
                                             /* MBox Create */
{
   OS_EVENT* pevent = OSMboxCreate(msg);
   OSMboxAccept(pevent);
   /* myOSEventTbl[(int)peventpevent->avgWait   = 0;
   /*  pevent->statCount = 0; */
   return pevent;
}

UBYTE *myOSMboxPost(OS_EVENT *pevent, void *msg)                                   /* MBox Post   */
{
   /*if(pevent->OSEventGrp || pevent->OSEventPtr == (void*) 0)
        pevent->timerIn = OSTimeGet(); */
   return (UBYTE*) OSMboxPost(pevent,msg);
}

void *myOSMboxPend(OS_EVENT *pevent, UWORD timeout, UBYTE *err)
               /*MBox Pend    */
{
   unsigned long newTime      = 0;
   void*       newResult      = OSMboxPend(pevent,timeout,err);

   newTime = newTime;
   return newResult;
}


void* myOSMboxAccept(OS_EVENT *pevent)                                             /* MBox Accept */
```

(os_file.c continue 8/13)

```
{
   unsigned long newTime        = 0;
        void*   newResult       = OSMboxAccept(pevent);

   newTime = newTime;
   return newResult;
}


unsigned int myOSMboxPeek(OS_EVENT *pevent)
{
   int* boxFull = pevent->OSEventPtr;

        if(boxFull != (void*)0)
        return MBOX_FULL;                    //Returns a 1
   else
        return MBOX_NOT_FULL;
}
/***********************************************************************************************************
*                     uC/OS-II Customized Functions
*                      Message Queue Features
*
* VERSION:            1.0
* PROJECT:            Glacial Monitoring System
* MODIFIED:           May 29 2008
* AUTHOR:             Justin Reina
***********************************************************************************************************/

/***********************************************************************************************************
*                     CREATE QUEUE
* PROJECT:            Glacial Monitoring System
* AUTHOR:             Justin Reina
***********************************************************************************************************/
OS_EVENT *myOSQCreate(void **start, UWORD size)
{
   /********************************************************************************************************
   *                                  UCOS-II Section
   ********************************************************************************************************/
   OS_EVENT* pevent = OSSemCreate(1);
   MY_OS_Q*  pq;

   OS_ENTER_CRITICAL();

   pq                  = myOSQFreeList;
   pevent->OSEventPtr  = pq;
   if(myOSQFreeList != (MY_OS_Q*)0)
   {
      myOSQFreeList    = myOSQFreeList->OSQPtr;

      pq->OSQStart     = start;
      pq->OSQEnd       = &start[size];
      pq->OSQCurr       = (start-1);
      pq->OSQIn        = start;
      pq->OSQOut        = start;
      pq->OSQSize       = size;
      pq->OSQEntries     = 0;
      pq->queueIndex    = 0;
      pq->queueOut      = 0;
      OSSemAccept(pevent);
   }

   /********************************************************************************************************
   *                  ADDITIONAL STATISTICS FEATURES
   ********************************************************************************************************/
   /*pevent->avgWait    = 0;
   pevent->statCount = 0;    */
   pq->maxMsgCt        = 0;
   pq->queueIndex      = 0;
   return pevent;
}

/***********************************************************************************************************
*                     POST TO QUEUE
* PROJECT:            Glacial Monitoring System
```

(os_file.c continue 9/13)

```
* AUTHOR:              Justin Reina
***********************************************************************************************************/
UBYTE *myOSQPost(OS_EVENT *pevent, void *msg)
{
   MY_OS_Q* pq = pevent->OSEventPtr;
   /********************************************************************************************************
   *                   ADDITIONAL STATISTICS FEATURES
   ********************************************************************************************************/
   //TBD
   justinPtr=display_Event->OSEventPtr;
   OS_ENTER_CRITICAL();

   /********************************************************************************************************
   *                   UCOS-II Section
   ********************************************************************************************************/
   if(pevent->OSEventGrp && (pq->OSQEntries < pq->OSQSize))
   {
      pq->OSQCurr = pq->OSQIn;

      *pq->OSQIn++ = msg;
       if(justinPtr == pq)
        delay_ms(1);
      pq->OSQEntries++;
      if(pq->OSQIn == pq->OSQEnd)
         pq->OSQIn = pq->OSQStart;
      pq->queueIndex = (pq->queueIndex+1)%(pq->OSQSize);

      OSSemPost(pevent);                             /* Assert Flag                                      */

      return (UBYTE*)1;
   }
        else
   {
        if(pq->OSQEntries >= pq->OSQSize)
      {
        OS_EXIT_CRITICAL();
        return (UBYTE*)0;
      }
      else
      {
         pq->OSQCurr = pq->OSQIn;

        *pq->OSQIn++ = msg;
        if(justinPtr == pq)
        delay_ms(1);
        pq->OSQEntries++;
        if(pq->OSQIn == pq->OSQEnd)
        pq->OSQIn = pq->OSQStart;
        pq->queueIndex = (pq->queueIndex+1)%(pq->OSQSize);
        OSSemPost(pevent);
        OS_EXIT_CRITICAL();
      }
   }
   return (UBYTE*)1;
}

/***********************************************************************************************************
*                      POST TO FRONT OF QUEUE
* PROJECT:             Glacial Monitoring System
* AUTHOR:              Justin Reina
***********************************************************************************************************/
UBYTE *myOSQPostFront(OS_EVENT *pevent, void *msg)
{
   MY_OS_Q* pq = pevent->OSEventPtr;
   /********************************************************************************************************
   *                   ADDITIONAL STATISTICS FEATURES
   ********************************************************************************************************/

   OS_ENTER_CRITICAL();
   if(pq->OSQEntries < pq->OSQSize)
   {
        //pevent->timerIn = OSTimeGet();
        delay_ms(1);
   }
```

(os_file.c continue 10/13)

```
   /********************************************************************************************************
   *                   UCOS-II Section
   ********************************************************************************************************/
   if(pevent->OSEventGrp && (pq->OSQEntries < pq->OSQSize))
   {
      if(pq->OSQOut == pq->OSQStart)
      {
         pq->OSQOut = pq->OSQEnd;
        pq->queueOut = (pq->OSQSize);
      }
      pq->OSQOut--;
      pq->queueOut--;

      *pq->OSQOut = msg;
      pq->OSQEntries++;


      OSSemPost(pevent);  //Assert Flag
      OS_EXIT_CRITICAL();
      return (UBYTE*)1;
   }
       else
   {
       if(pq->OSQEntries >= pq->OSQSize)
      {
       OS_EXIT_CRITICAL();
        return (UBYTE*)0;
      }
      else
      {
         if(pq->OSQOut == pq->OSQStart)
         {
            pq->OSQOut = pq->OSQEnd;
            pq->queueOut = (pq->OSQSize);
         }
         delay_ms(1);
         pq->OSQOut--;
         pq->queueOut--;
         *pq->OSQOut = msg;
                if(justinPtr == pq)
               delay_ms(1);
          pq->OSQEntries++;

      OSSemPost(pevent);
      OS_EXIT_CRITICAL();
      }
   }
   return (UBYTE*)1;
  /* if(pevent->OSEventGrp || pq->OSQEntries >= pq->OSQSize)
   pevent->timerIn = OSTimeGet();   */
}


/***********************************************************************************************************
*                      POST OVER THE FRONT OF QUEUE
* PROJECT:             Glacial Monitoring System
* AUTHOR:              Justin Reina
***********************************************************************************************************/
UBYTE myOSQPostOverFront(OS_EVENT* pevent,void* msg)
{
   MY_OS_Q* pq = pevent->OSEventPtr;
   msg = msg;
   /*******************************************************************************************************
   *                                                 ADDITIONAL STATISTICS FEATURES
   ********************************************************************************************************/

   OS_ENTER_CRITICAL();
   if(pq->OSQEntries < pq->OSQSize)
   {
       //pevent->timerIn = OSTimeGet();
               delay_ms(1);
   }
```

(os_file.c continue 11/13)

```
   /*******************************************************************************************************
   *                                                                        UCOS-II Section
   *******************************************************************************************************/
   OSSemPost(pevent);

   return (UBYTE)1;
  /* if(pevent->OSEventGrp || pq->OSQEntries >= pq->OSQSize)
        pevent->timerIn = OSTimeGet();   */
}


/*******************************************************************************************************
*                                                                           ACCEPT QUEUE MESSAGE
* PROJECT:              Glacial Monitoring System
* AUTHOR:               Justin Reina
*******************************************************************************************************/
void* myOSQAccept (OS_EVENT *pevent)
{
   unsigned long newTime      = 0;
   void*       newResult      = OSQAccept(pevent);
   newTime = newTime;
   return newResult;
}

/********************************************************************************************************
*                      PEND ON QUEUE
* PROJECT:              Glacial Monitoring System
* AUTHOR:               Justin Reina
********************************************************************************************************/
void *myOSQPend(OS_EVENT *pevent, UWORD timeout, UBYTE *err)
{
   MY_OS_Q* pq = pevent->OSEventPtr;
   MY_OS_Q* DPq = display_Event->OSEventPtr;
   MY_OS_Q* MPq = measure_Event->OSEventPtr;

   /*****************************************************************************************************
   *                                  UCOS-II Section
   *****************************************************************************************************/
   void* msg = (void*) 0;

   OSSemPend(pevent,timeout,err);

   if(DPq == pq)
       OSSemPost(dispMboxWrite_Sem);
   else if (MPq == pq)
      OSSemPost(msrMboxWrite_Sem);

   OS_ENTER_CRITICAL();
   if(pq->OSQEntries != 0 && (*err != OS_TIMEOUT))
   {
       msg = *(pq->OSQOut++);
      pq->queueOut = (pq->queueOut+1)%(pq->OSQSize);

      if(justinPtr == pq)
       delay_ms(1);
      pq->OSQEntries--;

      if(pq->OSQOut == pq->OSQEnd)
       pq->OSQOut = pq->OSQStart;
   }
   else if(pq->OSQEntries != 0 && (*err == OS_TIMEOUT))
               myOS_ErrReport[myOS_ErrCount++] = ERR_QUEUE_PEND;
   /*****************************************************************************************************
   *                                  ADDITIONAL STATISTICS FEATURES
   *****************************************************************************************************/
   else
       delay_ms(5);
   if(*err != OS_TIMEOUT && pq->OSQEntries)
       OSSemPost(pevent);
   OS_EXIT_CRITICAL();
   if(msg == (void*) 0)
       delay_ms(2);
   return msg;
}
```

(os_file.c continue 12/13)

```
/************************************************************************************************************
*                      Customized Event Functions
* PROJECT:             Glacial Monitoring System
* AUTHOR:              Justin Reina
************************************************************************************************************/
void myOSEvenList_Init()
{
   unsigned int i =0;
   for(i=0;i<(MY_OS_MAX_QS-1);i++)
      myOSQTbl[i].OSQPtr = &myOSQTbl[i+1];
    myOSQTbl[MY_OS_MAX_QS-1].OSQPtr   = (void*)0;
}

unsigned int myOSQPeek(OS_EVENT* pevent)
{
   MY_OS_Q* pq = pevent->OSEventPtr;     //Returns a '1' If There is space to post
        if(pq->OSQEntries < pq->OSQSize)
        return Q_NOT_FULL;
   else
        return Q_FULL;
}

void myOSQInit()
{
   unsigned int i;
   for(i=0;i<(MY_OS_MAX_QS-1);i++)
        myOSQTbl[i].OSQPtr = &myOSQTbl[i+1];
   myOSQTbl[MY_OS_MAX_QS-1].OSQPtr = (OS_Q*) 0;
   myOSQFreeList          = &myOSQTbl[0];
}
```

Figure 30. OS Source File

Maintask.h – Main Task Header File

```
//PREPROC COMMANDS----------------------------------------------------------------------------------------
#ifndef UCOS
   #define UCOS
   #include "ucos.h"
#endif

/************************************************************************************************************
*                                                            COMMAND RESPONSE TASK
************************************************************************************************************/
#define LOGGING_OFF           0x11
#define LOGGING_ON            0x22
#define MEASUREMENTS_ON       0x33
#define MEASUREMENTS_OFF      0x44
#define CURRENT_MEASUREMENTS  0x55

#define SAMPLE_RATE_MESS      0xFF
#define SUSPEND_REQ_MESS      0xCC
#define AUTHOR_COMMAND        0x98
#define AUTHOR_WARN_TRANSMIT  0x65
typedef struct
{
   UBYTE                      commErr;
   void              ***      currSysStatus;
   unsigned short* dataLogging;
} CommandRespData;


extern void msgToParse(void*, void*);
extern void far commandRespTask(void*);

/************************************************************************************************************
*                                                    PARSE TASK
************************************************************************************************************/
#define TEMP_IX               2
#define FLOW_IX      6
#define CARB_IX      10
```

(mainTask.h continue 1/3)

```
#define SULF_IX      14
#define THERMAL_IX   18
#define DIGIFLOW_IX  22
#define SPACE        ' '
#define M_RESPONSE   'M'
#define W_RESPONSE   'W'
typedef struct
{
   int deleteMe;
   unsigned char parseErr;
} ParseData;
void far parseTask(void*);

/***********************************************************************************************************
*                                                                  SERIAL COM TASK
***********************************************************************************************************/
#define  GLOBAL_RXBUF_SIZE 2000
#define  GLOBAL_TXBUF_SIZE 2000

#define  LOCAL_RXBUF_SIZE     75
#define  LOCAL_TXBUF_SIZE     75

#define  RX_ALMOST_FULL       70
#define  BYTE_THRESHOLD       50

#define  ENTER_KEY            13        // ascii dec
#define  LINE_FEED            10       // ascii dec
#define  XOFF                 0x13
#define XON                   0x11

#define BAUD_9600             8
#define SERIAL_BUF            2000
#define  LENGTH_IX 1
#define  BODY_IX 5

typedef struct
{
   unsigned char* tempCorrPtr;
   unsigned char* flowCorrPtr;
   unsigned char* carbonCorrPtr;
   unsigned char* sulfurCorrPtr;

   unsigned char* rxBufPtr;              /* receiver and transmit pointers                                */
       unsigned char* txBufPtr;

   unsigned int *serialWarnEvent;
   UBYTE                             serComErr;
} SerComData;

void far serComTask(void*);
int getInt(char);

//SERIAL HANDLER DEFINITIONS--------------------------------------------------------------------------------


extern SerComData serComData;
extern unsigned short  txEventFlag;
extern unsigned short  warnEventFlag;



/***********************************************************************************************************
*                                                   MEASURE TASK
***********************************************************************************************************/
#define ADC_CH0_PIO      11           // All PIO Definitions are copied from the main.h
#define ADC_CH1_PIO      11    // location. This allows for easy location of the
#define ADC_CH2_PIO      11    // PIO line of interest, and also to determine
#define ADC_CH3_PIO      11    // avail PIO lines when necessary.

#define CH0              0
#define CH1              1
#define CH2              2
#define CH3              3
```

(mainTask.h continue 2/3)

```
#define READ_MODE     1
#define WRITE_MODE    2

#define TEMP_ADC_MAX_COUNT    165                          // Emperically Established
#define FLOW_ADC_MAX_COUNT    24             //
#define CARBON_ADC_MAX_COUNT  92             //
#define SULFUR_ADC_MAX_COUNT  91             //

#define BUFF_8                8                             // Measurement Buffer Size

//STATUS DEFINITIONS----------------------------------------------------------------------------------------
#define BATT_DRAIN_PERIOD     100
#define BATT_CAPACITY         200

//DIGITAL FLOW DEFINITIONS----------------------------------------------------------------------------------
#define STD_DIG_FLOW_PERIOD   100
#define DIG_FLOW_MESSAGE      "Digital Flow Rate"

//MEASURE STRUCTS/PROTOTYPES--------------------------------------------------------------------------------
#define MSR_ANALOG_SAMPLE_RATE(OS_TICKS_PER_SEC>>1)


#define MUX_CARB              0
#define MUX_SULF              1
#define MUX_CNTRL_PORT        0x200
#define CH0_CH2               (i+1)
#define LAST_ANALOG           3
typedef struct
{
   unsigned int               ADC_Counts[4];
   unsigned int               sampleRate;
   OS_EVENT*                  msrAnlg_Event;
   OS_EVENT*                  msrAnlg_Sem;
   OS_EVENT*                  msrMboxWrite_Sem;
   OS_EVENT*                  measure_Event;
} MeasureData;

void far measureTask(void*);

//STATUS STRUCTS/PROTOTYPES---------------------------------------------------------------------------------
typedef struct
{
       unsigned char*         battStatePtr;
   unsigned int               battDrainPeriod;
   unsigned char              minBattState;
}StatusData;

void far statusTask(void*);

//DIGITAL FLOW STRUCTS/PROTOTYPES---------------------------------------------------------------------------
#define DIG_FLOW_WAIT (3*OS_TICKS_PER_SEC)
#define DIG_FLOW_SAMPLE_RATE  (4*OS_TICKS_PER_SEC)
typedef struct
{
   unsigned int       *digitalFlow;
   unsigned int       *digFlowPeriod;
   unsigned int       *digFlowConvRate;
   unsigned long      *latchedCount;
   unsigned int    sampleRate;

   unsigned char   digFlowErr;

   OS_EVENT*       digFlow_Sem;
   OS_EVENT*       digFlow_Event;
}DigitalFlowData;

void far digitalFlowTask(void*);

/***********************************************************************************************************
*              DAQ Scheduler Task
*
* MAILBOXES:   TBD
* PARTNERS:    userProc, serialProc, msrAnlg, digFlow, glacialDepth, acqFFT
* PROJECT:            Glacial Monitoring System
```

(mainTask.h continue 3/3)

```
* MODIFIED:             May 29 2008
* AUTHOR:               Justin Reina, Khoa Nguyen, Thuat Nguyen
***********************************************************************************************************/
#define USER_GLACIAL_RQST              1234
#define USER_SAMPLE_RATE               1234

#define  SER_GLACIAL_RQST              1234
#define  SER_ON_RQST                   1234
#define   SER_OFF_RQST                 1234

/***********************************************************************************************************
*                       Thermal Measurement Task
*
* PROJECT:              Glacial Monitoring System
* MODIFIED:             May 29 2008
* AUTHOR:               Justin Reina, Khoa Nguyen, Thuat Nguyen
***********************************************************************************************************/
#define THERMAL_SIZE          256
#define THERMAL_BUF_SIZE      16
#define DIV_64                6
#define OFFSET                32
#define TICK_TIME             25
#define DIV_256               8
#define DIV_128               7
#define FREQ_CONV             1000000L
#define RESET                 0
#define INC_VAL               1
#define FFT_DAC_PORT          CH0
typedef struct
{
   unsigned int*        sigFreqPtr;
   unsigned long*       timerCtrPtr;
   unsigned int*        thermalBufPtr;
   unsigned int*        thermalBufIXPtr;

   OS_EVENT*            acqFFT_Sem;
   OS_EVENT*            acqFFT_Event;
   OS_EVENT*            timer2_Sem;
   OS_EVENT*            measure_Event;
   OS_EVENT*            msrMboxWrite_Sem;
   UBYTE                thermalErr;
   unsigned int          sampleRate;
}ThermalData;

void far thermalTask(void*);
signed int optfft(signed int x[THERMAL_SIZE], signed int y[THERMAL_SIZE]);
```

Figure 31. Main Task Header File

mainTask.c – Main Tasks Source File

```
/***********************************************************************************************************
*                      Main Tasks Source File
*                       -Measure Analog Task
*                       -Digital Flow Task
*                       -Status Task
*
* FILE:                 maintasks.c
* HEADER:               maintasks.h
* VERSION:              2.0
* PROJECT:              Glacial Monitoring System
* MODIFIED:             May 29 2008
* AUTHOR:               Justin Reina, Khoa Nguyen, Thuat Nguyen
***********************************************************************************************************/
#ifndef INCLUDES
    #define INCLUDES
    #include "includes.h"
#endif


unsigned static char   localRXBuf[LOCAL_RXBUF_SIZE];
unsigned static char   localTXBuf[LOCAL_RXBUF_SIZE];
```

(mainTask.c continue 1/15)

```
unsigned char  logOff         = LOGGING_OFF;
unsigned char  logOn          = LOGGING_ON;
unsigned char  measOn         = MEASUREMENTS_ON;
unsigned char  measOff        = MEASUREMENTS_OFF;
unsigned char  currMeas       = CURRENT_MEASUREMENTS;

   unsigned static short  overLimit = 0;
   unsigned static short  countChar = 0;
   unsigned static short  index     = 0;
   unsigned short        rxFull     = 0;
   unsigned short        stopFlow   = 0;
   unsigned long         tempTime   = 0;
   unsigned short        serHit     = 0;

   unsigned short localTXEvent   = 0;
   unsigned short sendAckOrNak   = 0;
   unsigned short error          = 0;
   unsigned short ackFlag        = 0;
   unsigned short nakCount       = 0;

/*************************************************************************************************
*                      Command Response Task
*
* VERSION:             1.0
* PROJECT:             Glacial Monitoring System
* MODIFIED:            May 29 2008
* AUTHOR:              Justin Reina
*************************************************************************************************/
void far commandRespTask(void* data)
{
 unsigned char* newMsgPtr = (void*) 0;
 const unsigned char   suspendMsg = SUSPEND_REQ_MESS;
 const unsigned char   sampleMsg  = SAMPLE_RATE_MESS;
 MY_OS_Q*              PRSEq      = parseInbox_Event->OSEventPtr;
 unsigned char         newMsg     = '\0';
 CommandRespData*      myVars     = data;

 for(;;)
 {
   /**********************************************************************************************
   *                          CHECK THE PARSE INBOX
   **********************************************************************************************/
   newMsgPtr = myOSMboxPend(parseInbox_Event,WAIT_FOREVER,&nullErr);
   newMsg    = *newMsgPtr;
   /*********************************************************************************************
   *                                  DECODE THE COMMAND
   *********************************************************************************************/
   switch(newMsg)
   {
       case LOGGING_OFF:
       *(myVars->dataLogging) = 0;
         break;
      case LOGGING_ON:
       *(myVars->dataLogging) = 1;
         break;
       case MEASUREMENTS_ON:
         if(msrAnlg_Sem->OSEventCnt)
            OSTaskResume(MEASURE_PRIORITY);
         if(acqFFT_Sem->OSEventCnt)
            OSTaskResume(THERMAL_PRIORITY);
         if(digFlow_Sem->OSEventCnt)
            OSTaskResume(DIGITAL_FLOW_PRIORITY);
      break;
    case MEASUREMENTS_OFF:
         if(!msrAnlg_Sem->OSEventCnt)
            myOSMboxPost(msrAnlg_Event,&suspendMsg);
         if(!acqFFT_Sem->OSEventCnt)
            myOSMboxPost(acqFFT_Event,&suspendMsg);
         if(!digFlow_Sem->OSEventCnt)
            myOSMboxPost(digFlow_Event,&suspendMsg);
         break;
      case CURRENT_MEASUREMENTS:
       OSSemPend(parseOutWrite_Sem,2*OS_TICKS_PER_SEC,&(myVars->commErr));
```

(mainTask.c continue 2/15)

```
         if(myVars->commErr)
            myOS_ErrReport[myOS_ErrCount++] = ERR_PEND_PARSE_OUT_TIMEOUT;
         else
         {
            msgToParse(myVars->currSysStatus,&parse_DOutbox[PRSEq->queueIndex]);
            myOSQPost(parseOutbox_Event,&parse_DOutbox[PRSEq->queueIndex]);
         }
         break;
      default:
        myOS_ErrReport[myOS_ErrCount++] = ERR_COMMAND_MESSAGE_UNKNOWN;
   }

 }
}
/****************************************************************************************************
*                              msgToParse Helper Function
****************************************************************************************************/
void msgToParse(void* dataFrom, void* dataTo)
{
   unsigned char tempGlacial[6];
   unsigned char tempBattState[6];

Compute_Msg*   from = dataFrom;
ParseOut_Msg*  to   = dataTo;

   to->author = AUTHOR_COMMAND;
   to->timeStamp = from->timeStamp;

   to->tempCorr[0] = from->tempCorr[0];
   to->tempCorr[1] = from->tempCorr[1];
   to->tempCorr[2] = from->tempCorr[2];
   to->tempCorr[3] = from->tempCorr[3];

   to->flowCorr[0] = from->flowCorr[0];
   to->flowCorr[1] = from->flowCorr[1];
   to->flowCorr[2] = from->flowCorr[2];
   to->flowCorr[3] = from->flowCorr[3];

   to->carbonCorr[0] = from->carbonCorr[0];
   to->carbonCorr[1] = from->carbonCorr[1];
   to->carbonCorr[2] = from->carbonCorr[2];
   to->carbonCorr[3] = from->carbonCorr[3];

   to->sulfurCorr[0] = from->sulfurCorr[0];
   to->sulfurCorr[1] = from->sulfurCorr[1];
   to->sulfurCorr[2] = from->sulfurCorr[2];
   to->sulfurCorr[3] = from->sulfurCorr[3];

   to->thermalCorr[0] = from->thermalCorr[0];
   to->thermalCorr[1] = from->thermalCorr[1];
   to->thermalCorr[2] = from->thermalCorr[2];
   to->thermalCorr[3] = from->thermalCorr[3];

   to->digFlowCorr[0] = from->digFlowCorr[0];
   to->digFlowCorr[1] = from->digFlowCorr[1];
   to->digFlowCorr[2] = from->digFlowCorr[2];
   to->digFlowCorr[3] = from->digFlowCorr[3];

   itoa(from->glacialVal,tempGlacial,BASE_DEC);

   to->glacialCorr[0] = tempGlacial[0];
   to->glacialCorr[1] = tempGlacial[1];
   to->glacialCorr[2] = tempGlacial[2];
   to->glacialCorr[3] = tempGlacial[3];

   itoa(from->battState,tempBattState,BASE_DEC);

   to->battState[0] = tempBattState[0];
   to->battState[1] = tempBattState[1];
   to->battState[2] = tempBattState[2];
   to->battState[3] = tempBattState[3];
}

/**********************************************************************************************************
```

(mainTask.c continue 3/15)

```
*                     Parse Task
*
* VERSION:            1.0
* PROJECT:            Glacial Monitoring System
* MODIFIED:           May 29 2008
* AUTHOR:             Justin Reina
***********************************************************************************************************/
void far parseTask(void* data)
{
 unsigned char*                newMsgPtr            = (void*) 0;
 unsigned char                 i                    = 0;
 ParseOut_Msg*                 newParseMsgPtr       = (void*) 0;
 ParseData*                    myVars               = data;
 MY_OS_Q*                      SCPQ                 = serComOut_Event->OSEventPtr;

 unsigned char*                newOutBoxPtr         = NULL;

 for(;;)
 {
   /********************************************************************************************************
   *                              CHECK THE SERIAL COM MESSAGE QUEUE
   ********************************************************************************************************/
   newMsgPtr = myOSQPend(serComIn_Event,OS_TICKS_PER_SEC,&(myVars->parseErr));
   if(myVars->parseErr != OS_TIMEOUT)
   {
   /********************************************************************************************************
   *                          PROCESS THE SERIAL QUEUE
   ********************************************************************************************************/
       switch(*newMsgPtr)
      {
       case 'I':
               myOS_ErrReport[myOS_ErrCount++] = ERR_PARSE_INIT_CODE_INBOX;
            break;
         case 'E':
               myOS_ErrReport[myOS_ErrCount++] = ERR_PARSE_ERROR_CODE_INBOX;
         case 'D':
               myOSMboxPost(parseInbox_Event,&logOn);
            break;
         case 'L':
               myOSMboxPost(parseInbox_Event,&logOff);
            break;
         case 'M':
               myOSMboxPost(parseInbox_Event,&currMeas);
            break;
         case 'S':
               myOSMboxPost(parseInbox_Event,&measOn);
            break;
         case 'P':
               myOSMboxPost(parseInbox_Event,&measOff);
            break;
                      case 'W':
             myOS_ErrReport[myOS_ErrCount++] = WINDMILL_WINDMILL_WINDMILL;
             break;
         default:
               myOS_ErrReport[myOS_ErrCount++] = ERR_PARSE_UNKNOWN_COM_MESS;
      }
   }
   /********************************************************************************************************
   *                              CHECK THE PARSE OUTBOX (BUT DO NOT PEND)
   ********************************************************************************************************/
   if(!parseOutbox_Event->OSEventCnt)
   {
      /*****************************************************************************************************
      *                          DECODE THE COMMAND MESSAGE
      *****************************************************************************************************/
      newParseMsgPtr = myOSQPend(parseOutbox_Event,OS_TICKS_PER_SEC<<2,&nullErr);
      if(nullErr != OS_TIMEOUT)
      {
         newOutBoxPtr    = &serCom_DOutbox[SCPQ->queueIndex];
         switch(newParseMsgPtr->author)
         {
            case AUTHOR_COMMAND:
              *(newOutBoxPtr+0) = M_RESPONSE;
              *(newOutBoxPtr+1) = SPACE;
```

(mainTask.c continue 4/15)

```
               for(i = 0; i<4; i++)
               {
                  *(newOutBoxPtr +i + TEMP_IX)     = newParseMsgPtr->tempCorr[i];
                  *(newOutBoxPtr +i + FLOW_IX)     = newParseMsgPtr->flowCorr[i];
                  *(newOutBoxPtr +i + CARB_IX)     = newParseMsgPtr->carbonCorr[i];
                  *(newOutBoxPtr +i + SULF_IX)     = newParseMsgPtr->sulfurCorr[i];
                  *(newOutBoxPtr +i + THERMAL_IX)  = newParseMsgPtr->thermalCorr[i];
                  *(newOutBoxPtr +i + DIGIFLOW_IX) = newParseMsgPtr->digFlowCorr[i];
               }
               break;
            case AUTHOR_WARN_TRANSMIT:
               *(newOutBoxPtr+0) = W_RESPONSE;
               *(newOutBoxPtr+1) = SPACE;
               for ( i = 0; i < 4; i++)
               {
                  *(newOutBoxPtr +i + TEMP_IX)      = newParseMsgPtr->tempCorr[i];
                  *(newOutBoxPtr +i + FLOW_IX)      = newParseMsgPtr->flowCorr[i];
                  *(newOutBoxPtr +i + CARB_IX)      = newParseMsgPtr->carbonCorr[i];
                  *(newOutBoxPtr +i + SULF_IX)      = newParseMsgPtr->sulfurCorr[i];
               }
               break;
            default:
                myOS_ErrReport[myOS_ErrCount++] = ERR_PARSE_UNKNOWN_COM_MESS;
         }
      /****************************************************************************************************
      *                         POST THE MESSAGE
      ****************************************************************************************************/
         if(myOSQPeek(serComOut_Event) == Q_NOT_FULL)
            myOSQPost(serComOut_Event,newOutBoxPtr);
         else
            myOS_ErrReport[myOS_ErrCount++] = ERR_SER_COM_OUTBOX_FULL;
               }
    }
 }
}
/**********************************************************************************************************
*                     Measure Analog Task
*
* VERSION:            1.0
* PROJECT:            Glacial Monitoring System
* MODIFIED:           May 29 2008
* AUTHOR:             Justin Reina
**********************************************************************************************************/
unsigned int justin = 0;
void far measureTask(void* data)             /* Vars is the data pointer                                  */
{
 MeasureData*  myVars                 = data;
 Measure_Msg*  newMsgPtr              = NULL;
 OS_EVENT*     pevent                 = myVars->measure_Event;
 MY_OS_Q*      pq                     = pevent->OSEventPtr;
 unsigned short       recomputeFlag   = 0, i = 0, suspendFlag = 0;
 unsigned int         currMeas = 0, *newSamplePtr;
 for(;;)
 {
   OSSemAccept(myVars->msrAnlg_Sem);
   recomputeFlag = 0;
//---------Read form CH1 – CH3: Carbon and Sulfur read through CH3
   for(i=0;i<4;i++)
   {
      if(i == LAST_ANALOG)
      {
         outportb(MUX_CNTRL_PORT,MUX_SULF);
         ae_ad12(i);
         ae_ad12(i);
         currMeas = ae_ad12(i);
      }
      else
      {
         outportb(MUX_CNTRL_PORT,MUX_CARB);
         ae_ad12(CH0_CH2);
         ae_ad12(CH0_CH2);
         currMeas = ae_ad12(CH0_CH2);
      }
      if(currMeas - myVars->ADC_Counts[i])
```

(mainTask.c continue 5/15)

```
         recomputeFlag++;
      myVars->ADC_Counts[i] = currMeas;      //put in buffer for later proccess
   }
   myVars->ADC_Counts[1] = 0;                /* Temporary Rememdy For Broken Flow Potentiometer          */
 //--------------------

   if(myOSQPeek(myVars->measure_Event) != Q_FULL && recomputeFlag)
   {
      OSSemPend(myVars->msrMboxWrite_Sem,WAIT_FOREVER,&nullErr);

      newMsgPtr = &measure_Dbox[pq->queueIndex];

      newMsgPtr->author    = AUTHOR_MSR_ANALOG;
      newMsgPtr->timeStamp = OSTimeGet();
      newMsgPtr->tempRaw   = myVars->ADC_Counts[0];
      newMsgPtr->flowRaw   = myVars->ADC_Counts[1];
      newMsgPtr->carbonRaw = myVars->ADC_Counts[2];
      newMsgPtr->sulfurRaw = myVars->ADC_Counts[3];

      myOSQPost(measure_Event,&measure_Dbox[pq->queueIndex]);
   }
   //Storing data into buffer
   if(myOSMboxPeek(myVars->msrAnlg_Event) == MBOX_FULL)
   {
        newSamplePtr = myOSMboxAccept(myVars->msrAnlg_Event);
       switch(*newSamplePtr)
      {
        case SAMPLE_RATE_MESS:
            myVars->sampleRate = *newSamplePtr;
            break;
         case SUSPEND_REQ_MESS:
            suspendFlag++;
            break;
         default:
               myOS_ErrReport[myOS_ErrCount++] = ERR_MSR_ANLG_MESSAGE;
      }
   }
   if(suspendFlag)
   {
        suspendFlag  = 0;
       OSSemPost(myVars->msrAnlg_Sem);
       OSTaskSuspend(OS_PRIO_SELF);
   }
   else
        OSTimeDly(myVars->sampleRate);
 }
}

/***********************************************************************************************************
*                      Status Task
*
* VERSION:             1.0
* PROJECT:             Glacial Monitoring System
* MODIFIED:            May 29 2008
* AUTHOR:              Justin Reina
***********************************************************************************************************/
void far statusTask(void* data)       /* Decrements the battery                                          */
{                                     /*by one unit                                                      */
   StatusData* myVars         = data;
   void*       newMsgPtr      = NULL;
   MY_OS_Q*    Dpq            = display_Event->OSEventPtr;
   unsigned long lastDispPost = 0;

   newMsgPtr = newMsgPtr;
        for(;;)
   {
     (*myVars->battStatePtr)--;       /* Decrement battery status                                        */

     if(!*(myVars->battStatePtr))
        *(myVars->battStatePtr) = BATT_CAPACITY; /* Reset If Empty                                       */

     if(*(myVars->battStatePtr) < myVars->minBattState &&
         ((OSTimeGet()-lastDispPost) > (4*OS_TICKS_PER_SEC)))
     {
```

(mainTask.c continue 6/15)

```
         OS_ENTER_CRITICAL();
         if(myOSQPeek(display_Event) == Q_FULL )
            myOS_ErrReport[myOS_ErrCount++] = ERR_DISP_MBOX_FULL;
         else
         {
            OSSemPend(dispMboxWrite_Sem,WAIT_FOREVER,&nullErr);
            lastDispPost = OSTimeGet();
            display_Dbox[Dpq->queueIndex] = AUTHOR_STATUS;
            myOSQPost(display_Event,&display_Dbox[Dpq->queueIndex]);
         }
         OS_EXIT_CRITICAL();
     }
      OSTimeDly(myVars->battDrainPeriod);
   }
}

/************************************************************************************************************
*                      Digital Flow Task
*
* VERSION:             1.0
* PROJECT:             Glacial Monitoring System
* MODIFIED:            May 29 2008
* AUTHOR:              Justin Reina
************************************************************************************************************/

void far digitalFlowTask(void* data)
{
 char   tempFlowDisp[5], suspendFlag = 0;
 unsigned long tempFlow =0;
 unsigned int* newSamplePtr;
 DigitalFlowData* myVars = data;
 for(;;)
 {
       OSSemAccept(myVars->digFlow_Sem);
   /*********************************************************************************************************
   *                                                  INTERRUPT SETUP
   ********************************************************************************************************/
   OSSemPend(timer2_Sem,T2_WAIT_COUNT,&myVars->digFlowErr);                /* Wait for Timer2              */
   if(myVars->digFlowErr == OS_TIMEOUT)
      //myOS_ErrReport[myOS_ErrCount++] = ERR_DIG_FLOW_TIMEOUT_ON_T2;
      delay_ms(2);
   else
   {
      //outportb(INT1CON,INT1_UNMASK);                              /* Unmask the digital flow measurement */
      OSSemPend(digFlowResp_Sem,DIG_FLOW_WAIT,&myVars->digFlowErr);        /* Wait for Response           */
      if(myVars->digFlowErr == OS_TIMEOUT)
         //myOS_ErrReport[myOS_ErrCount++] = ERR_DIG_FLOW_TIMEOUT_ON_INT_RSP;
       delay_ms(2);
      else
      {
               //outportb(INT1CON,INT1_MASK);  /* Mask The Interrupt                                           */
               OSSemPost(timer2_Sem);          /* Return Timer2                                                */
   /***************************************************************************************************
   *                   PROCESS RESULT
   **************************************************************************************************/
               tempFlow = digFlowCounts[2] + digFlowCounts[3] + digFlowCounts[4] + digFlowCounts[5];
               tempFlow = tempFlow>>2;
               if(tempFlow)
         {
               tempFlow = 20000/tempFlow;
               tempFlow = *myVars->digFlowConvRate / tempFlow;
         }
         else
         {
               //myOS_ErrReport[myOS_ErrCount++] = ERR_DIG_MEAS_ZERO_CT;
            delay_ms(2);
         }
       itoa((unsigned int) tempFlow,tempFlowDisp,BASE_DEC);                /* Compute ASCII Disp          */
       *myVars->digitalFlow = (unsigned int) tempFlow;                     /* Store Value                */
       }
   }
   /********************************************************************************************************
   *                                  UPDATE DIGITAL SAMPLE PERIOD
   *********************************************************************************************************/
```

(mainTask.c continue 7/15)

```
   if(myOSMboxPeek(myVars->digFlow_Event))
   {
       newSamplePtr = myOSMboxAccept(myVars->digFlow_Event);
       switch(*newSamplePtr)
      {
       case SAMPLE_RATE_MESS:
               myVars->sampleRate = *newSamplePtr;
            break;
         case SUSPEND_REQ_MESS:
               suspendFlag++;
            break;
         default:
               myOS_ErrReport[myOS_ErrCount++] = ERR_DIG_FLOW_MESSAGE;
      }
   }
   /*******************************************************************************************************
   *                                                                       SUSPEND ON REQUEST
   *******************************************************************************************************/
   if(suspendFlag)
   {
       suspendFlag  = 0;
      OSSemPost(myVars->digFlow_Sem);
       OSTaskSuspend(OS_PRIO_SELF);
   }
   else
       OSTimeDly(myVars->sampleRate);
 }
}

/***********************************************************************************************************
*              DAQ Scheduler Task
*
* MAILBOXES:   TBD
* PARTNERS:    userProc, serialProc, msrAnlg, digFlow, glacialDepth, acqFFT
* PROJECT:     Glacial Monitoring System
* MODIFIED:    May 29 2008
* AUTHOR:      Justin Reina
***********************************************************************************************************/
void far DAQSchedulerTask(void* data)
{
 void  *newUserMsgPtr = NULL, *newSerMsgPtr = NULL;
 unsigned int  newUserMsg     = 0,    newSerMsg = 0;
 data = data;
 for(;;)
 {
       if(myOSQPeek(userProc_Event))                        // Check userProc_MBox
       {
               newUserMsgPtr = OSQAccept(userProc_Event);
               newUserMsg = *(unsigned int*) newUserMsgPtr;

      if(USER_GLACIAL_RQST == newUserMsg)                   // Request Glacial Depth Measurement
      {
         // Stuff
      }
      else if (USER_SAMPLE_RATE == newUserMsg)              // Request For Change To Sampling Rate
      {
         // Other Stuff
      }
      else
         myOS_ErrReport[myOS_ErrCount++] = ERR_DAQ_SCHED_DECODE_USER_PROC;
   }

       if(myOSQPeek(serialProc_Event))
   {
      newSerMsgPtr = OSQAccept(serialProc_Event);
      newSerMsg = *(unsigned int*) newUserMsgPtr;

      if(SER_GLACIAL_RQST == newUserMsg/*&!glacial Active*/)        // Request Glacial Depth Measurement
      {
         newSerMsg = newSerMsg;
      }
      else if (SER_OFF_RQST == newUserMsg/*&!AlreadyOff*/)         // Request For Change To Sampling Rate
      {
         newSerMsgPtr = newSerMsgPtr;
```

(mainTask.c continue 8/15)

```
      }
      else if (SER_ON_RQST  == newUserMsg /*&!AlreadyOn*/)
      {
       //Stuff
      }
      else
         myOS_ErrReport[myOS_ErrCount++] = ERR_DAQ_SCHED_DECODE_SER_PROC;
    }
 // Process the Glacial Flow Task if necessary
 }
}

/***********************************************************************************************************
*                      Thermal Measurement Task
*
* VERSION:             1.0
* PROJECT:             Glacial Monitoring System
* MODIFIED:            May 29 2008
* AUTHOR:              Justin Reina
**********************************************************************************************************/
void far thermalTask(void* data)
{
 signed int            real[THERMAL_SIZE],imag[THERMAL_SIZE];
 unsigned int          i,*count, maxIx;
 unsigned int*         signalFreq, bufIndex, *newSamplePtr;
 unsigned long         samplingFs;
 unsigned char         suspendFlag    = 0;
 Measure_Msg*          newMsgPtr      = NULL;
 MY_OS_Q*              Mpq             = measure_Event->OSEventPtr;

 ThermalData* myVars = data;
 signalFreq = myVars->thermalBufPtr;

 count    = (unsigned int*) myVars->timerCtrPtr;        /* Used to not be dereferenced!! Ask Thuat About  */
 for(;;)
 {
   /********************************************************************************************************
   *                                          SETUP AND SEMAPHORE ACQUISITION
   *******************************************************************************************************/
   OSSemAccept(myVars->acqFFT_Sem);

   OS_ENTER_CRITICAL();
   OSSemPend(myVars->timer2_Sem,T2_WAIT_COUNT,&(myVars->thermalErr));

   if(myVars->thermalErr == OS_TIMEOUT)
      myOS_ErrReport[myOS_ErrCount++] = ERR_THERMAL_T2_TIMEOUT;
   else
   {
      /*****************************************************************************************************
      *                                       FFT ACQUISITION
      ****************************************************************************************************/
      bufIndex   = *(myVars->thermalBufIXPtr);

      //setMaskAllExceptTimer1(1);
      for(i = 0; i<256; i++)                /* Reset The FFT Input Buffer                                */
      {
          real[i] = 0;
          imag[i] = 0;
      }

      *count = 0;                          /* Reset the Sampling Frequency Variables                    */
      samplingFs = 0;
      maxIx = 0;


      ae_ad12(FFT_DAC_PORT);               /* Flush the A/D Hardware Buffer                             */
     ae_ad12(FFT_DAC_PORT);
      ae_ad12(FFT_DAC_PORT);

      outport(TCUCON,TIMERS_UNMASK);        /* Unmask All Timers
                                                     */
      for(i = 0; i<THERMAL_SIZE; i++)       /* Acquire FFT Data                                          */
      {
       real[i]= ae_ad12(FFT_DAC_PORT);
```

(mainTask.c continue 9/15)

```
      }
      outport(TCUCON,TIMERS_MASK);           /* Mask All Timers                          */
      OSSemPost(myVars->timer2_Sem);         /* Release The Timer2 Sem


       for(i = 0; i <THERMAL_SIZE; i++)     /* Calibration                               */
       {
           real[i] = (real[i]>>DIV_64);
           real[i] = (real[i] - OFFSET);
       }
       maxIx = optfft(real,imag);            /* Perform FFT on the collected signal      */

       *count = (*count)*TICK_TIME;         /* Calculating sampling freq                 */
       *count = (*count)>>DIV_128;

       if(!*count)
         myOS_ErrReport[myOS_ErrCount++] = ERR_THERMAL_MEAS_NO_COUNT;
       else
       {
         samplingFs = FREQ_CONV/(*count);

         *(signalFreq + bufIndex) = (unsigned int) ((maxIx*samplingFs)>>DIV_256);         /* Calc signal */
         *(myVars->thermalBufIXPtr)= *(myVars->thermalBufIXPtr)+1;                        /* frequency   */
         *(myVars->thermalBufIXPtr) = *(myVars->thermalBufIXPtr)%THERMAL_BUF_SIZE;
       }
   }
   OS_EXIT_CRITICAL();

   /********************************************************************************************************
   *                                  POST TO MAILBOX
   ********************************************************************************************************/
   if(myOSQPeek(myVars->measure_Event))
   {
      OSSemPend(myVars->msrMboxWrite_Sem,WAIT_FOREVER,&nullErr);

      newMsgPtr = &measure_Dbox[Mpq->queueIndex];

      newMsgPtr->author        = AUTHOR_THERMAL;
      newMsgPtr->timeStamp  = OSTimeGet();
      newMsgPtr->thermalRaw = *signalFreq;
      myOSQPost(measure_Event,&measure_Dbox[Mpq->queueIndex]);
      OSSemPost(myVars->msrMboxWrite_Sem);
   }
   /********************************************************************************************************
   *                                  CHECK ACTIVITY BOX
   ********************************************************************************************************/
   if(myOSMboxPeek(myVars->acqFFT_Event))
   {
       newSamplePtr = myOSMboxAccept(myVars->acqFFT_Event);
      switch(*newSamplePtr)
      {
         case SAMPLE_RATE_MESS:
               myVars->sampleRate = *newSamplePtr;
               break;
         case SUSPEND_REQ_MESS:
               suspendFlag++;
               break;
         default:
               myOS_ErrReport[myOS_ErrCount++] = ERR_MSR_ANLG_MESSAGE;
      }
   }

   if(suspendFlag)
   {
      suspendFlag  = 0;
      OSSemPost(myVars->acqFFT_Sem);
      OSTaskSuspend(OS_PRIO_SELF);
   }
   else
       OSTimeDly(myVars->sampleRate);
   }
 }

/***********************************************************************************************************
```

(mainTask.c continue 10/15)

```
*                       Serial Communication Task
*
* VERSION:              1.0
* PROJECT:              Glacial Monitoring System
* MODIFIED:             May 29 2008
* AUTHOR:               Justin Reina
***********************************************************************************************************/
#define LENGTH_IX 1
#define BODY_IX 5
#define M_LENGTH 26
#define W_LENGTH 18
#define M_FRAME_LENGTH 34
#define W_FRAME_LENGTH 26
#define START  '\x01'
#define END    '\x0A'
#define ZERO   '\x30'
#define TWO            '\x32'
#define THREE  '\x33'
#define FOUR   '\x34'
#define SIX            '\x36'

unsigned char mResponse[M_LENGTH];
unsigned char wResponse[W_LENGTH];
void far serComTask(void* data)
{

    MY_OS_Q*   SCPq     = serComIn_Event->OSEventPtr;
    for(;;)
    {
    SerComData* myVars = data;
    unsigned char checkByte1, checkByte2;
    unsigned short length;
    unsigned char* rxBufPtr      = myVars->rxBufPtr;
    unsigned char* txBufPtr      = myVars->txBufPtr;
    unsigned char* tempCorrPtr   = myVars->tempCorrPtr;
    unsigned char* flowCorrPtr   = myVars->flowCorrPtr;
    unsigned char* carbonCorrPtr = myVars->carbonCorrPtr;
    unsigned char* sulfurCorrPtr = myVars->sulfurCorrPtr;
    unsigned char* newMgsPtr         = (void*)0;
    unsigned int   checkSum          = 0;



   unsigned char    commandBody = 0;

   void*             newMsgPtr  = (void*) 0;

   // checksum, length, & validation stuff

   unsigned int   checksumFrame = 0, checksumComp = 0, lengthFrame = 0;

   unsigned static short receive = 1;
   unsigned short             i = 0;

   unsigned char  CFrame[4], ackOrNakType, char_byte;
         /*************************************************************************************************
         *                              CHECK FOR RECEPTION ON THE SERIAL PORT
         *************************************************************************************************/
         serHit = 0;
         tempTime = OSTimeGet();
        // OSTimeDly(430);
         while((OSTimeGet()-tempTime)<OS_TICKS_PER_SEC && !serHit)
         {             /* Wait One Second For Serial Hit*/
               serHit = serhit1(c1);
               delay_ms(2);
         }

         if(serHit)
         {
         //------------------------------------------------------------------------------------------------
            //-------------------------------- Receiving from SERIAL PORT --------------------------------//

            while (serhit1(c1))                      // check if there is a character in serial buffer
            {
```

(mainTask.c continue 11/15)

```
      if (countChar > LOCAL_RXBUF_SIZE)
      {
         putser1(XOFF,c1);
         rxFull = 1;                                  // we can't take any more character
      }

      if (countChar >= RX_ALMOST_FULL)
      {
         putser1(XOFF,c1);                            // reached the first limit, send OFF
         overLimit = 1;                              // well, we're over the first limit
      } else if ((countChar <= (RX_ALMOST_FULL - BYTE_THRESHOLD)) && (1 == overLimit)) {
         putser1(XON,c1);                             // we now have at least 50 bytes left, send ON
         overLimit = 0;                              // continue to accept more characters
      }

      if (countChar <= LOCAL_RXBUF_SIZE)
      {
         char_byte = getser1(c1);
         localRXBuf[countChar] = char_byte;            // store this temporarily in local buffer
         countChar++;

         if (char_byte == '\x0A')
         {
            for (i = 0; i < countChar; i++) {
               (*(rxBufPtr + i)) = localRXBuf[i];
            }
            countChar = 0;
            receive   = 1;                             // sent to real buffer, let's start over
         }
      }
   }

   //--------------------------------- Check checksum, length -----------------------------------//

      if ((*(rxBufPtr)) == '\x01' && receive)
      {
      countChar = 0;
              while ((*(rxBufPtr + countChar)) != '\x0A'){         // cnt num of char
                     countChar++;
      }
      countChar++;                                                 // the x0A was not counted

      if (*(rxBufPtr+1) == '\x06')                                 // received ACK
      {
                     ackFlag = 1;
      } else if (*(rxBufPtr+1) == '\x15'){                         // received NAK
                                                                   // resend last command
      }

      // length and checksum pulled from the 4B-length frame, 2B-checksum frame
      lengthFrame   = (getInt(*(rxBufPtr+1))<<12) + (getInt(*(rxBufPtr+2))<<8) +
                                       (getInt(*(rxBufPtr+3))<<4)  + getInt(*(rxBufPtr+4));
              checksumFrame = (getInt(*(rxBufPtr+6)))*10  + getInt(*(rxBufPtr+7));

      // checksum calculation
              for (i = 0; i < 6; i++){
                     checksumComp = checksumComp^(*(rxBufPtr + i));
              }
      CFrame[0] = '\x01'; CFrame[1] = '\x06';              // by default, make the C-frame an ACK
      CFrame[2] = '0';    CFrame[3] = '\x0A';

      // check to see if the length AND checksum are good
              if (countChar == lengthFrame && checksumComp == checksumFrame){
         sendAckOrNak = 1;
              } else {
              // checksum or length was NOT valid
         CFrame[1] = '\x15';                               // send a NAK to Java
         sendAckOrNak = 1;
      }

      // if 1, the msg was received correctly, 0 was a nak
      if (sendAckOrNak && receive)   // no 'receive' before
      {
```

(mainTask.c continue 12/15)

```
         // transfer C-frame to aTXBuf using the txBufPtr
         for (i = 0; i < 4; i++)
         {
              *(txBufPtr + i) = CFrame[i];
            localTXEvent = 1;
         }
         // sendAckOrNak = 0;   // might need to be 1
         // receive      = 0;
      }

      // reset the nakCount to zero & send I-frame
      if (nakCount == 4)
      {
         // send back an unknown response
         // transfer E-frame to aTXBufPtr
         // error = 1
         localTXEvent = 1;
         nakCount    = 0;
      }

      // if there's something to send, and we've just receive the 'receive' signal
      // 'receive' prevents this task from sendering continuously the in_buff of serial
      if (localTXEvent && receive)
      {
         putsers1(txBufPtr,c1);
         localTXEvent = 0;
         // receive = 0;                                          // might need to be 1
      }

      // send body of msg to parse task
      if (sendAckOrNak && receive)
      {
              commandBody = *(rxBufPtr+5);
         serCom_DInbox[SCPq->queueIndex] = commandBody;
         myOSQPost(serComIn_Event,&serCom_DInbox[SCPq->queueIndex]);
         parseTask(&commandBody);
      }
      receive = 0;
   }
}

/**************************************************************************************************
*                           CHECK FOR RECEPTION FROM THE PARSE TASK
**************************************************************************************************/
newMsgPtr =  myOSQPend(serComOut_Event,OS_TICKS_PER_SEC>>1,&(myVars->serComErr));


serCom_DOutbox[SER_OUT_MESSQ_SIZE][MAX_SER_MESSQ_SIZE];
if (myVars->serComErr != OS_TIMEOUT)
{
   //Read from newOutbox;
   if(*newMgsPtr == 'M')
   {

      mResponse[0] = START;
      mResponse[1] = ZERO;
      mResponse[2] = ZERO;
      mResponse[3] = THREE;
      mResponse[4] = FOUR;
      for(i = 0; i< M_LENGTH; i++)
      {
              mResponse[i+BODY_IX] = *(newMgsPtr+i);
      }
      for( i = 0; i< M_FRAME_LENGTH - 3; i++)
      {
       checkSum = checkSum^mResponse[i];
      }
      // DETERMINE CHECKSUM HERE
      checkByte1 = (unsigned char)(checkSum>>8);
      checkByte2 = (unsigned char)(checkSum);
      mResponse[BODY_IX + M_LENGTH]    = checkByte1;
      mResponse[BODY_IX + M_LENGTH +1] = checkByte2;
      mResponse[BODY_IX + M_LENGTH +2] = END;
      putsers1(mResponse,c1);
```

(mainTask.c continue 13/15)

```
            }
            else if( *newMgsPtr == 'W')
            {
               wResponse[0] = START;
               wResponse[1] = ZERO;
               wResponse[2] = ZERO;
               wResponse[3] = TWO;
               wResponse[4] = SIX;
               for(i = 0; i< W_LENGTH; i++)
               {
                      wResponse[i+BODY_IX] = *(newMgsPtr+i);
               }
               for( i = 0; i< W_FRAME_LENGTH - 3; i++)
               {
                checkSum = checkSum^mResponse[i];
               }
               checkByte1 = (unsigned char)(checkSum>>8);
               checkByte2 = (unsigned char)(checkSum);


               wResponse[BODY_IX + W_LENGTH] = checkByte1;
               wResponse[BODY_IX + W_LENGTH +1] = checkByte2;
               wResponse[BODY_IX + W_LENGTH +2] = END;
               putsers1(wResponse,c1);
         }
      }
   }
}

/*********************************************************************************************************/
/* optfft.c
/*
/* An optimized version of the fft function using only 16-bit integer math.
/*
/* Optimized by Brent Plump
/* Based heavily on code by Jinhun Joung
/*
/* - Works only for input arrays of 256 length.
/* - Requires two arrays of 16-bit ints.  The first contains the samples, the
/*   second contains all zeros.  The samples range from -31 to 32
/* - Returns the index of the peak frequency
/*********************************************************************************************************/
#define ABS(x)  (((x)<0)?(-(x)):(x))

#define CEILING(x) (((x)>511)?511:(x))

signed int optfft(signed int real[256], signed int imag[256]) {

signed int i, i1, j, l, l1, l2, t1, t2, u;

#include       "fft_Tables.c"

       /* Bit reversal. */
       /*Do the bit reversal */
       l2 = 128;
       i=0;
       for(l=0;l<255;l++) {
              if(l < i) {
                  j=real[l];real[l]=real[i];real[i]=j;
              }
              l1 = l2;
              while (l1 <= i){
                     i -= l1;
                     l1 >>= 1;
              }
              i += l1;
       }
       /* Compute the FFT */
       u = 0;
       l2 = 1;
       for(l=0;l<8;l++){
              l1 = l2;
              l2 <<= 1;
              for(j=0;j<l1;j++){
```

(mainTask.c continue 14/15)

```
                        for(i=j;i<256;i+=l2){
                                i1 = i + l1;
                                t1 = (u1[u]*real[i1] - u2[u]*imag[i1])/32;
                                t2 = (u1[u]*imag[i1] + u2[u]*real[i1])/32;
                                real[i1] = real[i]-t1;
                                imag[i1] = imag[i]-t2;
                                real[i] += t1;
                                imag[i] += t2;
                        }
                        u++;
                }
        }

        /* Find the highest amplitude value */
        /* start at index 1 because 0 can hold high values */
        j=1;
        l=0;
        for ( i=1; i<(128); i++ ) {
                l1 = square[CEILING(ABS(real[i]))]+square[CEILING(ABS(imag[i]))];
                if (l1 > l) {
                        j = i;
                        l = l1;
                }
        }
        return (j);
}
int getInt(char c)
{
   int myInt = (int)(c-'\x30');
        if (c=='A')
        return 10;
   else if (c=='B')
        return 11;
   else if (c=='C')
        return 12;
   else if (c=='D')
        return 13;
   else if (c=='E')
        return 14;
   else if (c=='F')
        return 15;

   return myInt;
}
```

Figure 32. Main Task Source File

secondaryTask.h – Secondary Task Header File

```
//COMPUTE DEFINITIONS----------------------------------------------------------------------------------------

//DISPLAY DEFINITIONS----------------------------------------------------------------------------------------
#define ROW_1 0x80
#define ROW_2 0xc0
#define LCD_T 0x82
#define LCD_F 0x87
#define LCD_B 0x8d
#define LCD_S 0xc3
#define LCD_C 0xcb

#define AUTHOR_COMPUTE 0x44
#define AUTHOR_STATUS  0x55

//WARN DEFINITIONS-------------------------------------------------------------------------------------------
#define ONE_SEC_TONE 170                             //Equiv To One Sec in the Timer2 ISR
#define TWO_SEC_TONE 340                             //Equiv To Two Sec in the Timer2 ISR
#define MIN_BATTERY  20
//SERIAL HANDLER DEFINITIONS---------------------------------------------------------------------------------
//ALARM HANDLER STRUCTS/PROTOTYPES---------------------------------------------------------------------------
#define GREEN_TYPE            0
#define YELLOW_TYPE           1
#define RED_TYPE              2

#define SOLID_TYPE            0
```

(secondaryTask.h continue 1/2)

```
#define SLOW_TYPE             1
#define FAST_TYPE             2

#define OS_ONE_SEC            100
#define NORMAL_STATE          1
#define NOT_NORMAL_STATE      0

//COMPUTE STRUCTS/PROTOTYPES--------------------------------------------------------------------------------
#define TEMP_VAR  5
#define FLR_VAR   9
#define CLVL_VAR  7
#define SLVL_VAR  6

#define AUTHOR_MSR_ANALOG  0x11
#define AUTHOR_DIG_FLOW    0x22
#define AUTHOR_GLACIAL     0x33
#define AUTHOR_THERMAL     0x44

#define DATA_LOGGING_OFF   0
#define DATA_LOGGING_ON    1
#define JIMS_FACE          42

typedef struct
{
   unsigned int** tempBufPtr;                //Measure data pointers
   unsigned int** flowBufPtr;
   unsigned int** carbonBufPtr;
   unsigned int** sulfurBufPtr;

   unsigned char* battStatePtr;

   unsigned char* tempCorrPtr;               //Corrected Data pointers
   unsigned char* flowCorrPtr;
   unsigned char* carbonCorrPtr;
   unsigned char* sulfurCorrPtr;

   OS_EVENT*      measure_Event;
   unsigned short dataLogging;

   unsigned int*  tempADCMax;
   unsigned int*  tempADCMin;
   unsigned int*  flowADCMax;
   unsigned int*  flowADCMin;
   unsigned int*  sulfurADCMax;
   unsigned int*  sulfurADCMin;
   unsigned int*  carbonADCMax;
   unsigned int*  carbonADCMin;
} ComputeData;

void far computeTask(void*);
void copyComputeMsg(void*,void*);
//DISPLAY STRUCTS/PROTOTYPES--------------------------------------------------------------------------------
typedef struct
{
   unsigned char* tempCorrPtr;               // corrected data pointers
   unsigned char* flowCorrPtr;
   unsigned char* carbonCorrPtr;
   unsigned char* sulfurCorrPtr;

   unsigned char* battStatePtr;               // pointer to battery state
   OS_EVENT*      display_Event;
   void***        currSysStatusPtr;
} DisplayData;

void far displayTask(void*);                  //Helper function-display data on LCD
void updateVal(char*, char*,int);
//WARN STRUCTS/PROTOTYPES-----------------------------------------------------------------------------------
typedef enum {G = 0, Y = 1, R = 2} LED;                    //LED Type Enum
typedef enum {MYFALSE = 0, MYTRUE = 1  } myBool;           //Bool Type Enum

typedef struct
{
   unsigned int**      tempBufPtr;
   unsigned int**      flowBufPtr;
```

(secondaryTask.h continue 2/2)

```
   unsigned int**     carbonBufPtr;
   unsigned int**     sulfurBufPtr;
   unsigned char*     battStatePtr;

   unsigned char      tempOutRange;
   unsigned char      flowOutRange;
   unsigned char      carbonOutRange;
   unsigned char      sulfurOutRange;

   myBool             tempHigh;
   myBool             flowHigh;
   myBool             carbonHigh;
   myBool             sulfurHigh;

   unsigned short*    tempL0;
   unsigned short*    tempL1;
   unsigned short*    flowL0;
   unsigned short*    flowL1;
   unsigned short*    carbonL0;
   unsigned short*    carbonL1;
   unsigned short*    sulfurL0;
   unsigned short*    sulfurL1;

   unsigned int*      newLEDType;
   unsigned int*      newLEDState;
   unsigned int*      newAlarmState;
   unsigned int*      newAlarmDuration;
   unsigned int*      newInitDuration;
   unsigned int*      newInitCount;

   unsigned int*      alarmCycleActive;             //He Writes to to indicate active alarm cycle
   unsigned int*      normalState;
   void***             currSysStatusPtr;
} WarnData;

void far warnTask(void*);
void           adjustLevel(unsigned int*,int*,int,int*,int*,int);
void           LED_alarmDisp(LED,int,int,int);

//ALARM ACK STRUCTS/PROTOTYPES------------------------------------------------------------------------------
typedef struct
{
        int deleteMe;
} AlarmAckData;

void far alarmAckTask(void* myTCB);

//ALARM HANDLER STRUCTS/PROTOTYPES-------------------------------------------------------------------------
typedef struct
{
   unsigned int* currState;                          //Protected Variables
   unsigned int* currAlarmDuration;
   unsigned int* currInitDuration;
   unsigned int* remInitCount;
   //Flags
   unsigned int* alarmAcknowledge;                   //He reads to see if alarm was acknowledged
   unsigned int* alarmCycleActive;                   //He Writes to to indicate active alarm cycle
   unsigned int* normalState;

   unsigned int* newLEDType;                         //Input Values
   unsigned int* newLEDState;
   unsigned int* newAlarmState;
   unsigned int* newAlarmDuration;
   unsigned int* newInitDuration;
   unsigned int* newInitCount;
} AlarmHandlerData;

void far alarmHandlerTask(void*);
void setLED(unsigned int,unsigned int);
void setSpkr(unsigned int);
```

Figure 33. Secondary Task Header File

secondaryTask.c – Secondary Task Source File

```c
/***********************************************************************************************************
*                     Secondary Tasks
*
* FILE:               secondarytasks.c
* HEADER:             secondarytasks.h
* VERSION:            2.0
* PROJECT:            Glacial Monitoring System
* MODIFIED:           May 29 2008
* AUTHOR:             Justin Reina, Khoa Nguyen, Thuat Nguyen
***********************************************************************************************************/
#ifndef INCLUDES
    #define INCLUDES
    #include "includes.h"
#endif
unsigned char  mySignature = AUTHOR_COMPUTE;
unsigned int tempDisplayCount = 0;
unsigned int tempWarnCount           = 0;
char  tempChars[3][17] = {{'T',':','X','X','X','F',':','X','X','X','X','B',':','X','X','X','\0'},
                         {'S','L',':','X','X','X','X','X','C','L',':','X','X','X','X','X','\0'},
                         {'G','D',':','X','X','X','X','X','T','H',':','X','X','X','X','X','\0'}};
char    tempBatt[5];

/***********************************************************************************************************
*                     Compute Task
*
* VERSION:            1.0
* PROJECT:            Glacial Monitoring System
* MODIFIED:           May 29 2008
* AUTHOR:             Justin Reina
***********************************************************************************************************/
void far computeTask(void* data)
{
 Compute_Msg   currentStat;
 ComputeData*  myVars      = data;    //Pull Off Data
 Measure_Msg*  newMsgPtr   = NULL;
 OS_EVENT*     Mpevent     = myVars->measure_Event;
 MY_OS_Q*      Mpq         = Mpevent->OSEventPtr;
 MY_OS_Q*      Dpq         = display_Event->OSEventPtr;
 MY_OS_Q*      Cpq         = compute_Event->OSEventPtr;
 MY_OS_Q*      sysPq       = compute_Event->OSEventPtr;
 unsigned int  tempVal     = 0, callWarn = 0, oldVals[4] = {0,0,0,0};
 unsigned long lastDispPost = 0;

 Mpq = Mpq;
 currentStat.digFlowVal = 123;
 currentStat.thermalVal = 345;
 itoa(currentStat.digFlowVal,(char*) &(currentStat.digFlowCorr[0]),   BASE_DEC);
 itoa(currentStat.thermalVal,(char*) &(currentStat.thermalCorr[0]),   BASE_DEC);

 for(;;)
 {
   newMsgPtr = myOSQPend(Mpevent,WAIT_FOREVER,&nullErr);

       /****************************************************************************************************
      *                       DECODE NEW MEASURMENT MESSAGE AND UPDATE STATUS
      *****************************************************************************************************/
      callWarn = 0;
      switch(newMsgPtr->author)
      {
         case AUTHOR_MSR_ANALOG:
            OS_ENTER_CRITICAL();              /* Convert The Values Based Upon the ADC Scale              */
                                              /* and then update the analog fields                        */
            currentStat.timeStamp = OSTimeGet();
            currentStat.battState = *(myVars->battStatePtr);

            tempVal = newMsgPtr->tempRaw;      /* Temp                                                    */
            tempVal -= *myVars->tempADCMin;
            tempVal *= 100;
            tempVal /=(*myVars->tempADCMax - *myVars->tempADCMin);
            tempVal  = TEMP_VAR + (tempVal<<3)/10;
            currentStat.tempVal = tempVal;
            itoa(tempVal,(char*) &(currentStat.tempCorr[0]),   BASE_DEC);
            if(tempVal - oldVals[0])
```

(secondaryTask.c continue 1/10)

```
               callWarn++;
            oldVals[0] = tempVal;

            tempVal = newMsgPtr->flowRaw;       /* Flow                                                  */
            tempVal -= 0;
            tempVal *= 100;
            tempVal /=(*myVars->flowADCMax - *myVars->flowADCMin);
            tempVal = FLR_VAR + (tempVal<<1);
            currentStat.flowVal = tempVal;
            itoa(tempVal,(char*) &(currentStat.flowCorr[0]),   BASE_DEC);
            if(tempVal - oldVals[1])
               callWarn++;
            oldVals[1] = tempVal;

            tempVal = newMsgPtr->carbonRaw;     /* Carbon                                                */
            tempVal -= *myVars->carbonADCMin;
            tempVal *= 100;
            tempVal /=(*myVars->carbonADCMax - *myVars->carbonADCMin);
            tempVal = CLVL_VAR + tempVal + (tempVal<<1)/10;
            currentStat.carbonVal = tempVal;
            itoa(tempVal,(char*) &(currentStat.carbonCorr[0]), BASE_DEC);
            if(tempVal - oldVals[2])
               callWarn++;
            oldVals[2] = tempVal;

            tempVal = newMsgPtr->sulfurRaw;     /* Sulfur                                                */
            tempVal -= *myVars->sulfurADCMin;
            tempVal *= 100;
            tempVal /=(*myVars->sulfurADCMax - *myVars->sulfurADCMin);
            tempVal = SLVL_VAR + (tempVal<<3) + (tempVal/10);
            currentStat.sulfurVal = tempVal;
            itoa(tempVal,(char*) &(currentStat.sulfurCorr[0]), BASE_DEC);
            if(tempVal - oldVals[3])
               callWarn++;
            oldVals[3] = tempVal;

            /***********************************************************************************************
            *                        SEND SYSTEM STATUS MESSAGE
            ***********************************************************************************************/
            if(myVars->dataLogging)
            {
               if(sysPq->OSQEntries < sysPq->OSQSize)
               {
                  copyComputeMsg(&currentStat,&compute_Dbox[sysPq->queueIndex]);
                  OS_EXIT_CRITICAL();
                  myOSQPost(compute_Event,&compute_Dbox[sysPq->queueIndex]);
               }
               else
                       sysPq->OSQEntries = 0;
                       //myOS_ErrReport[myOS_ErrCount++] = ERR_COMPUTE_MBOX_FULL;
            }
            else
            {
               copyComputeMsg(&currentStat,&compute_Dbox[sysPq->queueOut]);
               OS_EXIT_CRITICAL();
               myOSQPostOverFront(compute_Event,&compute_Dbox[sysPq->queueOut]);
            }

            /***********************************************************************************************
            *                        REQUEST A REFRESH OF THE DISPLAY SCREEN
            ***********************************************************************************************/
            OS_ENTER_CRITICAL();
            if(!myOSQPeek(display_Event))
               myOS_ErrReport[myOS_ErrCount++] = ERR_DISP_MBOX_FULL;
            else if((OSTimeGet() - lastDispPost) > (OS_TICKS_PER_SEC) && (Dpq->OSQEntries < Dpq->OSQSize))
            {
               OSSemPend(dispMboxWrite_Sem,WAIT_FOREVER,&nullErr);
               lastDispPost = OSTimeGet();
               display_Dbox[Dpq->queueIndex] = AUTHOR_COMPUTE;
               myOSQPost(display_Event,&display_Dbox[Dpq->queueIndex]);
            }
            OS_EXIT_CRITICAL();
            break;
         case AUTHOR_DIG_FLOW:
```

(secondaryTask.c continue 2/10)

```
            break;
         case AUTHOR_GLACIAL:
            currentStat.glacialVal =newMsgPtr->glacialRaw;
            itoa(currentStat.glacialVal,&currentStat.glacialCorr,BASE_DEC);
            if(sysPq->OSQEntries < sysPq->OSQSize)
            {
               copyComputeMsg(&currentStat,&compute_Dbox[sysPq->queueIndex]);
               OS_EXIT_CRITICAL();
               myOSQPost(compute_Event,&compute_Dbox[sysPq->queueIndex]);
            }
            if(!myOSQPeek(display_Event))
               myOS_ErrReport[myOS_ErrCount++] = ERR_DISP_MBOX_FULL;
            if(Cpq->OSQEntries < Cpq->OSQSize)
            {

               copyComputeMsg(&currentStat,&compute_Dbox[sysPq->queueIndex]);
               OS_EXIT_CRITICAL();
               myOSQPost(compute_Event,&compute_Dbox[sysPq->queueIndex]);
            }
            else
               Cpq->OSQEntries = 0;

            OSSemPend(dispMboxWrite_Sem,WAIT_FOREVER,&nullErr);
            lastDispPost = OSTimeGet();
            display_Dbox[Dpq->queueIndex] = AUTHOR_COMPUTE;
            myOSQPost(display_Event,&display_Dbox[Dpq->queueIndex]);

         OS_EXIT_CRITICAL();
            break;
         case AUTHOR_THERMAL:
            currentStat.thermalVal =newMsgPtr->thermalRaw;
            itoa(currentStat.thermalVal,&currentStat.thermalCorr,BASE_DEC);

            if(sysPq->OSQEntries < sysPq->OSQSize)
            {
               copyComputeMsg(&currentStat,&compute_Dbox[sysPq->queueIndex]);
               OS_EXIT_CRITICAL();
               myOSQPost(compute_Event,&compute_Dbox[sysPq->queueIndex]);
            }
            if(!myOSQPeek(display_Event))
               myOS_ErrReport[myOS_ErrCount++] = ERR_DISP_MBOX_FULL;
            if(Cpq->OSQEntries < Cpq->OSQSize)
            {

               copyComputeMsg(&currentStat,&compute_Dbox[sysPq->queueIndex]);
               OS_EXIT_CRITICAL();
               myOSQPost(compute_Event,&compute_Dbox[sysPq->queueIndex]);
            }
            else
               Cpq->OSQEntries = 0;

            OSSemPend(dispMboxWrite_Sem,WAIT_FOREVER,&nullErr);
            lastDispPost = OSTimeGet();

            display_Dbox[Dpq->queueIndex] = AUTHOR_COMPUTE;
            myOSQPost(display_Event,&display_Dbox[Dpq->queueIndex]);
            break;
         default:
            myOS_ErrReport[myOS_ErrCount++] = ERR_MEASURE_AUTHOR_UNKNOWN;
      }
      /*********************************************************************************************
      *                              UPDATE THE WARN STATUS
      *********************************************************************************************/
      if(newMsgPtr->author == AUTHOR_MSR_ANALOG && callWarn && !warn_Event->OSEventCnt)
       myOSMboxPost(warn_Event,&warn_Mbox);

 }
}


/***************************************************************************************************
*                             COMPUTE MESSAGE COPY - HELPER FUNCTION
*
***************************************************************************************************/
```

(secondaryTask.c continue 3/10)

```c
void copyComputeMsg(void* from,void* to)
{
   Compute_Msg* msgTo = to;
   Compute_Msg* msgFrom = from;

   msgTo->timeStamp = msgFrom->timeStamp;

   msgTo->tempVal = msgFrom->tempVal;
   msgTo->flowVal = msgFrom->flowVal;
   msgTo->carbonVal = msgFrom->carbonVal;
   msgTo->sulfurVal = msgFrom->sulfurVal;

   msgTo->digFlowVal = msgFrom->digFlowVal;
   msgTo->glacialVal = msgFrom->glacialVal;
   msgTo->thermalVal = msgFrom->thermalVal;

   msgTo->tempCorr[0] = msgFrom->tempCorr[0];
   msgTo->tempCorr[1] = msgFrom->tempCorr[1];
   msgTo->tempCorr[2] = msgFrom->tempCorr[2];
   msgTo->tempCorr[3] = msgFrom->tempCorr[3];
   msgTo->tempCorr[4] = msgFrom->tempCorr[4];

   msgTo->flowCorr[0] = msgFrom->flowCorr[0];
   msgTo->flowCorr[1] = msgFrom->flowCorr[1];
   msgTo->flowCorr[2] = msgFrom->flowCorr[2];
   msgTo->flowCorr[3] = msgFrom->flowCorr[3];
   msgTo->flowCorr[4] = msgFrom->flowCorr[4];

   msgTo->carbonCorr[0] = msgFrom->carbonCorr[0];
   msgTo->carbonCorr[1] = msgFrom->carbonCorr[1];
   msgTo->carbonCorr[2] = msgFrom->carbonCorr[2];
   msgTo->carbonCorr[3] = msgFrom->carbonCorr[3];
   msgTo->carbonCorr[4] = msgFrom->carbonCorr[4];

   msgTo->sulfurCorr[0] = msgFrom->sulfurCorr[0];
   msgTo->sulfurCorr[1] = msgFrom->sulfurCorr[1];
   msgTo->sulfurCorr[2] = msgFrom->sulfurCorr[2];
   msgTo->sulfurCorr[3] = msgFrom->sulfurCorr[3];
   msgTo->sulfurCorr[4] = msgFrom->sulfurCorr[4];

   msgTo->digFlowCorr[0] = msgFrom->digFlowCorr[0];
   msgTo->digFlowCorr[1] = msgFrom->digFlowCorr[1];
   msgTo->digFlowCorr[2] = msgFrom->digFlowCorr[2];
   msgTo->digFlowCorr[3] = msgFrom->digFlowCorr[3];
   msgTo->digFlowCorr[4] = msgFrom->digFlowCorr[4];
   msgTo->digFlowCorr[5] = msgFrom->digFlowCorr[5];

   msgTo->thermalCorr[0] = msgFrom->thermalCorr[0];
   msgTo->thermalCorr[1] = msgFrom->thermalCorr[1];
   msgTo->thermalCorr[2] = msgFrom->thermalCorr[2];
   msgTo->thermalCorr[3] = msgFrom->thermalCorr[3];
   msgTo->thermalCorr[4] = msgFrom->thermalCorr[4];
   msgTo->thermalCorr[5] = msgFrom->thermalCorr[5];

   msgTo->glacialCorr[0] = msgFrom->glacialCorr[0];
   msgTo->glacialCorr[1] = msgFrom->glacialCorr[1];
   msgTo->glacialCorr[2] = msgFrom->glacialCorr[2];
   msgTo->glacialCorr[3] = msgFrom->glacialCorr[3];
   msgTo->glacialCorr[4] = msgFrom->glacialCorr[4];
   msgTo->glacialCorr[5] = msgFrom->glacialCorr[5];

   msgTo->battState = msgFrom->battState;
}

/****************************************************************************************************
*                     Display Task
*
* VERSION:            1.0
* PROJECT:            Glacial Monitoring System
* MODIFIED:           May 29 2008
* AUTHOR:             Justin Reina
****************************************************************************************************/
void far displayTask(void* data)
{
```

(secondaryTask.c continue 4/10)

```
DisplayData*        myVars  = data;
Compute_Msg*  currSysStatus = **(myVars->currSysStatusPtr);
MY_OS_Q*                   DPq = display_Event->OSEventPtr;
MY_OS_Q*                   CPq = compute_Event->OSEventPtr;
unsigned char  dispType = 0, i =0;
unsigned long  lastTypeSwitch  = 0;

unsigned int*       newMsgPtr  = NULL;

DPq = DPq;
CPq = CPq;
for(;;)
{
   newMsgPtr = myOSQPend(display_Event,WAIT_FOREVER,&nullErr);
   itoa(*(myVars->battStatePtr),tempBatt,BASE_DEC);
   OS_ENTER_CRITICAL();
   switch(*newMsgPtr)
   {
     case AUTHOR_COMPUTE:
        currSysStatus = **(myVars->currSysStatusPtr);
                                                         /* Must Ensure Atomic Access to              */
         switch(dispType)
         {
            case 0:
               updateVal((char*)&(currSysStatus->tempCorr[0]),&tempChars[0][2],3);
               updateVal((char*)&(currSysStatus->flowCorr[0]),&tempChars[0][7],4);
               updateVal(tempBatt,&tempChars[0][13],3);
               updateVal((char*)&(currSysStatus->sulfurCorr[0]),&tempChars[1][3],5);
               updateVal((char*)&(currSysStatus->carbonCorr[0]),&tempChars[1][11],5);
               lcd_movecursor(ROW_1);
                lcd_putstr(&tempChars[0][0]);
               lcd_movecursor(ROW_2);
               lcd_putstr(&tempChars[1][0]);
               break;
            case 1:
               updateVal((char*)&(currSysStatus->tempCorr[0]),&tempChars[0][2],3);
               updateVal((char*)&(currSysStatus->flowCorr[0]),&tempChars[0][7],4);
               updateVal(tempBatt,&tempChars[0][13],3);
               updateVal((char*)&(currSysStatus->glacialCorr[0]),&tempChars[2][3],5);
               updateVal((char*)&(currSysStatus->thermalCorr[0]),&tempChars[2][11],5);
               lcd_movecursor(ROW_1);
                lcd_putstr(&tempChars[0][0]);
               lcd_movecursor(ROW_2);
               lcd_putstr(&tempChars[2][0]);
               break;
         }

         if(myOS_ErrCount)
         {
            lcd_movecursor(0xCF);
            lcd_put('X');
         }
         OS_EXIT_CRITICAL();
         if(OSTimeGet()-lastTypeSwitch > OS_TICKS_PER_SEC)
         {
            dispType = (dispType+1)%2;
            lastTypeSwitch = OSTimeGet();
         }
        OSTimeDly(OS_TICKS_PER_SEC>>1);
         break;

      case AUTHOR_STATUS:
         for(i=0;i<2;i++)
         {
            OS_ENTER_CRITICAL();
            lcd_movecursor(0x80);              //Put it out to the LCD.
            lcd_put('%');
            lcd_putstr(itoa(*(myVars->battStatePtr)>>1,NULL,BASE_DEC));
            lcd_putstr(" remaining!   ");
            lcd_fillrow(2,' ');
            OS_EXIT_CRITICAL();

            OSTimeDly(OS_TICKS_PER_SEC>>1);
            lcd_fillrow(1,' ');
```

(secondaryTask.c continue 5/10)

```
            }
               break;
         case JIMS_FACE:
               OS_ENTER_CRITICAL();
            lcd_fillrow(1,' ');
            lcd_fillrow(1,' ');
            lcd_movecursor(0x80);
            lcd_putstr("   JIM'S FACE   ");
            lcd_movecursor(0xc0);
            lcd_putstr("     :-P        ");
            OS_EXIT_CRITICAL();
            OSTimeDly(OS_TICKS_PER_SEC);
         default:
         {
            myOS_ErrReport[myOS_ErrCount++] = ERR_DISP_MBOX_DECODE_UNKNOWN;
            OSTimeDly(10);
         }
      }
      OS_EXIT_CRITICAL();
       }
}

//Helper function-display data on LCD
void updateVal(char* newVal, char* oldVal,int type)
{
   unsigned char i = 0;

   while(*newVal!= '\0')
   {
      *oldVal++ = *newVal++;
      i++;
   }

   while(i<type)
   {
      *oldVal++ = ' ';
      i++;
   }
}

/**********************************************************************************************************
*                      Warn Task
*
* VERSION:             1.0
* PROJECT:             Glacial Monitoring System
* MODIFIED:            May 29 2008
* AUTHOR:              Justin Reina
*********************************************************************************************************/
void far warnTask(void* data)
{
 WarnData* myVars = data;
 unsigned int  temp_L0,temp_L1,flow_L0,flow_L1,    /* Initialize Warning Limits                          */
               carbon_L0,     carbon_L1,     sulfur_L0,     sulfur_L1;
 unsigned int  myLEDState     = 0,           myLEDType, myAlarmDuration = 0,
               myAlarmState = 0,      myInitialDuration = 0,
               myInitCount = 0,              myStatus = 0;

 myOSMboxPend(warn_Event,WAIT_FOREVER,&nullErr);  /* Initialize the P&R Mechanism                        */

 OSSemAccept(warn_Sem);

 for(;;)
 {
   /*******************************************************************************************************
   *                         RESET WARN STATE WATCH
   ******************************************************************************************************/
   myLEDState  = 0;
   myLEDType = 0;
   myAlarmDuration = 0;
   myAlarmState = 0;
   myInitialDuration = 0;
   myInitCount = 0;
   myStatus = 0;
```

(secondaryTask.c continue 6/10)

```
/*******************************************************************************************************
*                         MEASUREMENT LIMIT CALCULATIONS
********************************************************************************************************/
temp_L0   = *(myVars->tempL0+0);            //Establish Conversions of All
temp_L0   = *(myVars->tempL0+1);            //    Limit Containers
temp_L0   = 10**(myVars->tempL0+1)    + *(myVars->tempL0+0);
temp_L1   = 10**(myVars->tempL1+1)    + *(myVars->tempL1+0);

flow_L0   = 10**(myVars->flowL0+1)    + *(myVars->flowL0+0);
flow_L1   = 10**(myVars->flowL1+1)    + *(myVars->flowL1+0);

carbon_L0 = 100**(myVars->carbonL0+1) + 10**(myVars->carbonL0+0);
carbon_L1 = 100**(myVars->carbonL1+1) + 10**(myVars->carbonL1+0);

sulfur_L0 = 100**(myVars->sulfurL0+1) + 10**(myVars->sulfurL0+0);
sulfur_L1 = 100**(myVars->sulfurL1+1) + 10**(myVars->sulfurL1+0);

/*******************************************************************************************************
*                                                CHECK OF ALARM STATE
********************************************************************************************************/
if(temp_L1 <= **(myVars->tempBufPtr)&& !*(myVars->alarmCycleActive))    /* Temp Alarm                 */
{
   myLEDType         = RED_TYPE;
   myLEDState        = SOLID_TYPE;                                     //'Solid'
   myAlarmDuration   = TWO_SEC_TONE;
   myAlarmState      = 1;
   myInitialDuration = ONE_SEC_TONE;
   myInitCount       = 2;
   myStatus++;
}

if(flow_L1 <= **(myVars->flowBufPtr)&& !*(myVars->alarmCycleActive))    /* Flow Alarm                 */
{
    myLEDType          = RED_TYPE;
   myLEDState        = SOLID_TYPE; //'Solid'
   myAlarmDuration   = ONE_SEC_TONE;
   myAlarmState      = 1;
   myInitialDuration = TWO_SEC_TONE;
   myInitCount     = 2;
   myStatus++;
}
if(carbon_L1 <= **(myVars->carbonBufPtr)&& !*(myVars->alarmCycleActive))/* Carbon Alarm               */
{
   myLEDType         = RED_TYPE;
   myLEDState        = SOLID_TYPE; //'Solid'
   myAlarmDuration   = ONE_SEC_TONE;
   myAlarmState      = 1;
   myInitialDuration = TWO_SEC_TONE;
   myInitCount     = 0;
   myStatus++;

   myVars->carbonHigh      = MYTRUE;
   myVars->carbonOutRange  = **(myVars->carbonBufPtr);
}

if(sulfur_L1 <= **(myVars->sulfurBufPtr)&& !*(myVars->alarmCycleActive)) /* Sulfur Alarm              */
{
   myLEDType         = RED_TYPE;
   myLEDState        = SOLID_TYPE; //'Solid'
   myAlarmDuration   = ONE_SEC_TONE;
   myAlarmState      = 1;
   myInitialDuration = TWO_SEC_TONE;
   myInitCount     = 0;
   myStatus++;

   myVars->sulfurHigh      = MYTRUE;
   myVars->sulfurOutRange  = **(myVars->sulfurBufPtr);
}

/********************************************************************************************************
*                          CHECK OF ALARM STATE
********************************************************************************************************/
if(**(myVars->tempBufPtr)>= temp_L0 && !myAlarmState)           /* Temp Warn                           */
{
```

(secondaryTask.c continue 7/10)

```
      myLEDType               = GREEN_TYPE;
      myLEDState              = FAST_TYPE;
      myStatus++;

      myVars->tempHigh     = MYTRUE;
      myVars->tempOutRange = **(myVars->tempBufPtr);
   }
   if(**(myVars->flowBufPtr)>= flow_L0 && !myAlarmState)           /* Flow Warn                          */
   {
      myLEDType               = GREEN_TYPE;
      myLEDState           = SLOW_TYPE;
      myStatus++;

      myVars->flowHigh     = MYTRUE;
      myVars->flowOutRange = **(myVars->flowBufPtr);
   }
   if(**(myVars->carbonBufPtr) >= carbon_L0 && !myAlarmState)      /* Carbon Warn                        */
   {
      myLEDType  = YELLOW_TYPE;
      myLEDState = (myLEDState == 1) ?   myLEDState:FAST_TYPE ;

      myVars->carbonHigh = MYTRUE;
      myVars->carbonOutRange = **(myVars->carbonBufPtr);
   }
   if(**(myVars->sulfurBufPtr) >= sulfur_L0 && !myAlarmState)      /* Sulfur Warn                        */
   {
      myLEDType  = YELLOW_TYPE;
      myLEDState = (myLEDState == 1) ?   myLEDState:FAST_TYPE ;

      myVars->sulfurHigh     = MYTRUE;
      myVars->sulfurOutRange = **(myVars->sulfurBufPtr);

   }
   /********************************************************************************************************
   *                                         UPDATE WARN STATE VALUES
   ********************************************************************************************************/

      *myVars->newLEDType            = myLEDType;           /* Store Values To Warn Handler              */
      *myVars->newLEDState           = myLEDState;
      *myVars->newAlarmState         = myAlarmState;
      *myVars->newAlarmDuration      = myAlarmDuration;
      *myVars->newInitDuration       = myInitialDuration;
      *myVars->newInitCount          = myInitCount;
       *myVars->normalState          = (myStatus) ? NORMAL_STATE:NOT_NORMAL_STATE;

      if(myStatus)                                        /* Check if the current state is an Alarm State */
      {
       myOSMboxPost(warnState_Event,&warnState_Mbox);         /* Post To Warn State Mailbox uncond*/

         if(!warnHndlr_Sem->OSEventCnt)                        /* If !WarnHandler Active...              */
               myOSMboxPost(warnHndlr_Event,&warnHndlr_Mbox);  /* Post To WarnHandler                    */
         OSTimeDly(OS_TICKS_PER_SEC>>2);                       /* Delay and recheck values               */
      }
      else                                                     /* Else: <--Normal Operation              */
      {
       OSSemPost(warn_Sem);                                  /* Release My Active Flag                  */
         OSMboxPend(warn_Event,WAIT_FOREVER,&nullErr);         /* Wait For Another Warn Event Trigg      */
         OSSemAccept(warn_Sem);                                /* <- 2nd call; check again               */
      }
 }
}

/*********************************************************************************************************
*                     Alarm Handler Task
*
* VERSION:            1.0
* PROJECT:            Glacial Monitoring System
* MODIFIED:           May 29 2008
* AUTHOR:             Justin Reina
********************************************************************************************************/
void far alarmHandlerTask(void* data)
{
   AlarmHandlerData* myVars = data;
   unsigned int resumeLoop = 0;
```

(secondaryTask.c continue 8/10)

```
       for(;;)
   {
      if(*myVars->normalState)                  //Warn writes to this flag.
       *myVars->currState = 0;                  //If it is high, state is rst

     //Alarm Handler State Machine-------------------------------------------------------------------------
     do
     {
               switch(*myVars->currState)
     {
       case 0://DECODE---------------------------------
                              if(*myVars->newAlarmState)
            {                         //Go To Alarm
               *myVars->currAlarmDuration = *myVars->newAlarmDuration;
               *myVars->currInitDuration  = *myVars->newInitDuration;
               *myVars->remInitCount      = *myVars->newInitCount;

               *myVars->alarmCycleActive  = 1;

               *myVars->currState   = 1;                            //Set State
               resumeLoop                            = 1;           //Reenter Loop
            }
            else
               setLED(*myVars->newLEDType,*myVars->newLEDState);
            break;                                                  //Do Nothing
         case 1://ALARM STATE-------------------------------------------------------------------------------
            resumeLoop = 0;
            while(*myVars->remInitCount-- && !*myVars->normalState)
            {
                  setSpkr(ON);               //ON
                  OSTimeDly((*myVars->currInitDuration) * OS_ONE_SEC);
                  setSpkr(OFF);              //OFF
                  OSTimeDly((*myVars->currInitDuration) * OS_ONE_SEC);
               }
            if(!*myVars->normalState)
                *myVars->currState = 2;
               break;
         case 2://ALARM SPKR CYCLE--------------------------------------------------------------------------
               while(!*myVars->normalState && !*myVars->alarmAcknowledge)
            {
               setSpkr(ON);
               OSTimeDly(*myVars->currAlarmDuration * OS_ONE_SEC);
               setSpkr(OFF);
               OSTimeDly(*myVars->currAlarmDuration * OS_ONE_SEC);
            }
            *myVars->alarmAcknowledge = 0;
               break;
       }
      } while(resumeLoop);

      *myVars->currState        = 0;         //Reset State to Initial
      *myVars->alarmCycleActive = 0;         //Deassert ActiveAlarm Flag
       OSTaskSuspend(OS_PRIO_SELF);
   }
}

//Alarm Handler Helper Functions----------------------------------------------------------------------------
void setLED(unsigned int color,unsigned int state)
{
   //Pretty Self Explanatory Type Defs. Ask if need clarification :)
       switch(color)
   {
       case GREEN_TYPE:
       outportb(LED0_PIN,OFF);
         outportb(LED1_PIN,OFF);
         break;
      case YELLOW_TYPE:
       outportb(LED0_PIN,ON);
         outportb(LED1_PIN,OFF);
         break;
      case RED_TYPE:
       outportb(LED0_PIN,OFF);
         outportb(LED1_PIN,ON);
```

(secondaryTask.c continue 9/10)

```
         break;
   }

   switch(state)
   {
       case SOLID_TYPE:
       outportb(STATE0_PIN,OFF);
         outportb(STATE1_PIN,OFF);
         break;
      case SLOW_TYPE:
       outportb(STATE0_PIN,ON);
         outportb(STATE1_PIN,OFF);
         break;
      case FAST_TYPE:
       outportb(STATE0_PIN,OFF);
         outportb(STATE1_PIN,ON);
         break;
   }
}

void setSpkr(unsigned int state)
{
       switch(state)
   {
       case OFF:
       outportb(SPKR_PIN,OFF);
         break;
      case ON:
       outportb(SPKR_PIN,ON);
         break;
   }
}
```

Figure 34. Secondary Task Source File

userTask.h – User Task Header File

```
//USER HANDLER DEFINITIONS----------------------------------------------------------------------------------
//Frame Lengths
#define ZERO_LENGTH    0
#define DIG_LENGTH     3
#define ADC_LENGTH    4

//Header Identifiers
#define TEMP_LIM       2
#define FLOW_LIM       3
#define CARB_LIM       4
#define SULF_LIM       5

#define TEMP_ADC       6
#define FLOW_ADC       7
#define CARB_ADC       8
#define SULF_ADC       9

#define GL_HEAD        0
#define SER_HEAD      1
#define ALARM_ACK      11
#define DIG_TRANS     10

//Packet2 Identifiers
#define ON             0
#define OFF            1

#define WARN           0
#define ALARM         1

#define MIN            0
#define MAX            1

#define MORE           1
#define NO_MORE      0
```

(userTask.h continue 1/2)

```
//USER HANDLER STRUCTS/PROTOTYPES----------------------------------------------------------------------------
typedef struct
{
   unsigned short* userInputBufPtr;
   unsigned int*   userInBufHeadPtr;
   unsigned int*   userInBufTailPtr;

   unsigned short* tempL0Ptr;
   unsigned short* tempL1Ptr;
   unsigned short* flowL0Ptr;
   unsigned short* flowL1Ptr;
   unsigned short* carbL0Ptr;
   unsigned short* carbL1Ptr;
   unsigned short* sulfL0Ptr;
   unsigned short* sulfL1Ptr;

   unsigned int* tempADCMinPtr;
   unsigned int* tempADCMaxPtr;
   unsigned int* flowADCMinPtr;
   unsigned int* flowADCMaxPtr;
   unsigned int* carbADCMinPtr;
   unsigned int* carbADCMaxPtr;
   unsigned int* sulfADCMinPtr;
   unsigned int* sulfADCMaxPtr;
} UserHandlerData;

void far userHandlerTask(void*);

//GLACIAL DISPLAY STRUCTS/PROTOTYPES------------------------------------------------------------------------
#define GLACIAL_MESSAGE "Glacial Depth"
#define GLACIAL_PING_WAIT(3*OS_TICKS_PER_SEC)
typedef struct
{
        unsigned long *pingCount;
   unsigned long *latchedCount;
   unsigned int  *glacialDepth;
   UBYTE          glacialErr;
   OS_EVENT*      glacialResponse_Event;
} GlacialData;

void far glacialTask(void*);

//SERIAL BUFFER STRUCTS/PROTOTYPES--------------------------------------------------------------------------
typedef struct
{
  int deleteMe;
} SerBufData;

void far serBufTask(void*);

//EXTRA NOT SURE---------------------------------------------------------------------------------------------
typedef struct
{
   unsigned short* userInputBufPtr;
   unsigned int*   userInBufHeadPtr;
   unsigned int*   userInBufTailPtr;

   unsigned int* tempADCMinPtr;
   unsigned int* tempADCMaxPtr;
   unsigned int* flowADCMinPtr;
   unsigned int* flowADCMaxPtr;
   unsigned int* carbADCMinPtr;
   unsigned int* carbADCMaxPtr;
   unsigned int* sulfADCMinPtr;
   unsigned int* sulfADCMaxPtr;

} SetADCData;

typedef struct
{
   unsigned short* userInputBufPtr;
   unsigned int*   userInBufHeadPtr;
   unsigned int*   userInBufTailPtr;
```

(userTask.h continue 2/2)

```
   unsigned short* tempL0Ptr;
   unsigned short* tempL1Ptr;
   unsigned short* flowL0Ptr;
   unsigned short* flowL1Ptr;
   unsigned short* carbL0Ptr;
   unsigned short* carbL1Ptr;
   unsigned short* sulfL0Ptr;
   unsigned short* sulfL1Ptr;
} SetLimitsData;

void far setADCTask(void*);
void far setLimitsTask(void*);
void far sendValuesTask(void*);
void far dispGlacialTask(void*);


//MISSILE LAUNCH---------------------------------------------------------------------------------------------
#define MAX_MISSILE_RQSTS 15
void far missileDefenseTask(void*);


//Set LCD and Speaker
extern void setLED(unsigned int,unsigned int);
extern void setSpkr(unsigned int);

void far jimsFaceTask(void*);
```

Figure 35. User Task Header File

userTask.c – User Task Source File

```
#ifndef INCLUDES
   #define     INCLUDES
   #include "includes.h"
#endif

unsigned int iHate = 0;
//USER HANDLER TASK------------------------------------------------------------------------------------------
/*This function will check for requestion.
       Read data put into buffer [20][5]
       the processes according to type.
*/
void far userHandlerTask(void* data)
{
 UserHandlerData* myVars = data;
 Measure_Msg*         newMsgPtr                      = NULL;
 MY_OS_Q*                     Mpq                            = measure_Event->OSEventPtr;
 for(;;)
 {
    unsigned short*   dataIn = myVars->userInputBufPtr;
    unsigned short    head = 0,                                     //head and tail of data
                        tail = 0,
                        more = 0;                                   //end Frame?

    //---------------------------READ DATA-------------------------------------------------------------------
    do{
       unsigned int           dataL = 0,     i = 0;                //data length and interation index
       unsigned short header,
                        d3 = 0,       d2  = 0,       d1 = 0,        //Data-type and and data in BCD
                        d0 = 0,  rts = 1;                           //ready to send and clear to send line
      outportb(CTS_PIN, OFF);

      for(i = 0; i <=dataL+1; i++)
      {
        rts = 1;
        while(rts)
        {                                          //wait for data
         rts = read_PIO(RTS_PIN);
         delay_ms(1);
        }
        outportb(CTS_PIN, ON);                     //clear to send
        while (!rts)
        {
```

(userTask.c continue 1/7)

```
             rts = read_PIO(RTS_PIN);               //ready to send data
            delay_ms(1);
        }

        switch(i)
        {
         case 0:                                   //check for appropriate space in buffer
            header = pioShort(DATA_PIN);
            dataL = (GL_HEAD  == header || SER_HEAD == header ||
                     ALARM_ACK==header) ? ZERO_LENGTH:dataL;
            dataL = (TEMP_LIM == header || FLOW_LIM == header ||
                     CARB_LIM == header || SULF_LIM == header) ? DIG_LENGTH :dataL;
            dataL = (TEMP_ADC == header || FLOW_ADC == header
                     || CARB_ADC == header || SULF_ADC == header) ? ADC_LENGTH :dataL;
            dataL = (DIG_TRANS == header) ? DIG_LENGTH:dataL;
            break;

          case 1:        //first data packet & check for more data if necessary
            d0     = (1 != dataL+1) ? pioShort(DATA_PIN) : d0;    //read 1st bcd
            more = (1 == dataL +1 && MORE == pioShort(DATA_PIN)) ? MORE : NO_MORE; //check for more data
            break;

          case 2:       //second data packet & check for more data if necesssary
            d1     = (2 != dataL+1) ? pioShort(DATA_PIN):d1;      //read 2nd bcd
            more = (2 == dataL +1 && MORE == pioShort(DATA_PIN)) ? MORE: NO_MORE; //check for more data
            break;

          case 3:       //third data packet  & check for more data if necessary
            d2     = (3 != dataL+1) ? pioShort(DATA_PIN) :d1;     //read 3d bcd
            more = (3 == dataL +1 && MORE == pioShort(DATA_PIN)) ? MORE: NO_MORE; //check for more data
            break;

          case 4:      //fourth data packet if necessary & check for more data if necessary
            d3     = (4 != dataL+1) ? pioShort(DATA_PIN):d3;      //read 4th bcd
            more = (4 == dataL +1 && MORE == pioShort(DATA_PIN)) ? MORE: NO_MORE; //check for more data
            break;

          case 5:      //check for more data
            more = (5 == dataL +1 && MORE == pioShort(DATA_PIN)) ? MORE: NO_MORE;
            break;
         }
         outportb(CTS_PIN, OFF); //ready to recived data  !<- Should Turn off CTS after reading data
       }
       //-----------------------READ DATA END--------------------------------------------------------

          tail = ((*myVars->userInBufTailPtr)<<2) + *myVars->userInBufTailPtr; //check for current tail

          //--------------------INSERT DATA INTO BUFFER----------------------------------------------------
          //Insert temperature limit into buffer
          if(TEMP_LIM == header || FLOW_LIM == header || CARB_LIM == header || SULF_LIM == header ||
DIG_TRANS == header)
            {
            *(dataIn+ tail + 0) = header;
            *(dataIn+ tail + 1) = d0;
            *(dataIn+ tail + 2) = d1;
            *(dataIn+ tail + 3) = d2;
            *(dataIn+ tail + 4) = 0;
            }

          //Insert ADC limt  and Digital Transducer data  into buffer
          if(TEMP_ADC == header || FLOW_ADC == header || CARB_ADC == header || SULF_ADC == header)
           {
            *(dataIn+ tail + 0) = header;
            *(dataIn+ tail + 1) = d0;
            *(dataIn+ tail + 2) = d1;
            *(dataIn+ tail + 3) = d2;
            *(dataIn+ tail + 4) = d3;
           }

          //Insert Glacer  or Serial data into buffer
          if(GL_HEAD == header || SER_HEAD == header || ALARM_ACK == header)
            {
            *(dataIn+ tail + 0) = header;
            }
```

(userTask.c continue 2/7)

```
       //increment buffer tail.
       (*myVars->userInBufTailPtr)++;
       *myVars->userInBufTailPtr =  (*myVars->userInBufTailPtr)%20;

    }while(more);

    //--------------------PROCESSING DATA-------------------------------------------------------------------

    //OSSchedLock();
    while(*myVars->userInBufHeadPtr  < *myVars->userInBufTailPtr)
    {
     head = ((*myVars->userInBufHeadPtr)<<2) + *myVars->userInBufHeadPtr;
      if(TEMP_LIM == *(dataIn+head) || FLOW_LIM == *(dataIn+head) ||
                       CARB_LIM == *(dataIn+head) || SULF_LIM == *(dataIn+head))
      {
         setLimitsTask(&setLimitsData);
         //OSTaskResume(SETLIMITS_PRIORITY);                //resume setLimitsTask
         //OSTaskResume(WARN_PRIORITY);
      }
      else if(TEMP_ADC == *(dataIn+head) || FLOW_ADC == *(dataIn+head)
                           || CARB_ADC == *(dataIn+head) || SULF_ADC == *(dataIn+head))
         setADCTask(&setADCData);

      else if (GL_HEAD == *(dataIn+head))
         glacialTask(&glacialData);
        //OSTaskResume(GLACIAL_PRIORITY);                  //resume glacial depth task

      else if (SER_HEAD == *(dataIn+head) )
         serBufTask(&serBufData);                          //resume serial task

      else if ( DIG_TRANS == *(dataIn+head))
         digFlowConvRate = (*(dataIn+head+1)*100)+ (*(dataIn+head+2)*10) + *(dataIn+head+3);

      else if ( ALARM_ACK == *(dataIn+head))
      {
        alarmAcknowledge = 1;
        setSpkr(OFF);
        setLED(RED_TYPE,SLOW_TYPE);
      }
      //OSTaskResume(ALARM_ACK_PRIORITY);
     (*myVars->userInBufHeadPtr)++;                                //increment header
      *myVars->userInBufHeadPtr = (*myVars->userInBufHeadPtr)%20;  //reset if end of buffer
    }
    //OSSchedUnlock();

    *myVars->userInBufHeadPtr = 0;
    *myVars->userInBufTailPtr = 0;

    head = 0;
    tail = 0;

    outportb(INT_RQST,0);
    outportb(INT6CON,INT6_UNMASK);
    OSTaskSuspend(OS_PRIO_SELF);
 }
}

/**********************************************************************************************************
*                     Glacial Depth Task
*
* VERSION:            1.0
* PROJECT:            Glacial Monitoring System
* MODIFIED:           May 29 2008
* AUTHOR:             Justin Reina
*********************************************************************************************************/
void far glacialTask(void* data)
{
 unsigned long        tempDepth;
 char                 intResponse = 0, pingAttempts = 0;
 unsigned long*       newMsgPtr   = NULL;
 GlacialData          *myVars     = data;
 MY_OS_Q*             MPq         = measure_Event->OSEventPtr;
 for(;;)
```

(userTask.c continue 3/7)

```
 {
   /********************************************************************************************************
   *                          SEMAPHORE SETUP
   *********************************************************************************************************/
   OSSemPend(timer2_Sem,T2_WAIT_COUNT,&(myVars->glacialErr));/* Wait For Timer2 Semaphore
        */

        if(myVars->glacialErr == OS_TIMEOUT)
        myOS_ErrReport[myOS_ErrCount++] = ERR_GLACIAL_T2_TIMEOUT;
   else
   {
      /*****************************************************************************************************
      *                       SEND PING AND WAIT FOR RESPONSE
      *****************************************************************************************************/
        intResponse    = 0;
        pingAttempts   = 0;
        while(!intResponse)
        {
         outportb(INT3CON,INT3_MASK);
         *myVars->pingCount    = 0;
         *myVars->latchedCount = 0;
         outportb(INT3CON,INT3_UNMASK);

                       pio_wr(PING_PIO_OUT,ON);
        // Send 1ms Ping
        delay_ms(2);                                                       // Delay
        pio_wr(PING_PIO_OUT,OFF);                                          // Off

         newMsgPtr = myOSMboxPend(myVars->glacialResponse_Event,10*GLACIAL_PING_WAIT,&(myVars->glacialErr));
                       if(myVars->glacialErr != OS_TIMEOUT && *newMsgPtr)
                       {
               *myVars->latchedCount = *newMsgPtr;
                       intResponse = 1;
         }
               else if(pingAttempts == 5)
               intResponse = 1;
        pingAttempts++;
        }

      outportb(INT3CON,INT3_MASK);    /* Mask INT3                                                      */
      OSSemPost(timer2_Sem);          /* Release Timer2 Semaphore                                      */

        if(pingAttempts == 5)
      {
               myOS_ErrReport[myOS_ErrCount++] = ERR_GLACIAL_PING_NO_RESPONSE;
         intResponse = 0;
      }
        else
        {
         /***************************************************************************************************
         *                    PROCESS RESULTS
       *****************************************************************************************************/
         tempDepth                            = *myVars->latchedCount;
         tempDepth                            = tempDepth>>1;
         tempDepth                           *= 175;
         tempDepth                           /= 1000;
         *myVars->glacialDepth = (unsigned int) tempDepth;

         if(intResponse && myOSMboxPeek(glacialResponse_Event) != MBOX_FULL)
         {
            /************************************************************************************************
            *                  POST MESSAGE TO COMPUTE
            ************************************************************************************************/
            measure_Dbox[MPq->queueIndex].glacialRaw = (unsigned int) tempDepth;
            measure_Dbox[MPq->queueIndex].author = AUTHOR_GLACIAL;
            myOSQPost(measure_Event,&measure_Dbox[MPq->queueIndex]);
        }
      }
      //LEAVE-----------------------------------------------------------------------------------------------
               //OSSemPost(LCDSem);                        // Post the LCD Semaphore
   }
      OSTimeDly(6*OS_TICKS_PER_SEC);
   }
}
```

(userTask.c continue 4/7)

```
//SERIAL BUFFER DISPLAY TASK--------------------------------------------------------------------------------
void far serBufTask(void* data)
{
   data = data; // Turns off Annoying Warning
        for(;;)
   {
      delay_ms(50);
        OSTaskSuspend(OS_PRIO_SELF);
   }
}

void far setADCTask(void* data)
{
   SetADCData* myVars  = data;
   unsigned short* myBufPtr = myVars->userInputBufPtr;
   unsigned int* tempMin    = myVars->tempADCMinPtr;
   unsigned int* tempMax    = myVars->tempADCMaxPtr;
   unsigned int* flowMin    = myVars->flowADCMinPtr;
   unsigned int* flowMax    = myVars->flowADCMaxPtr;
   unsigned int* carbMin    = myVars->carbADCMinPtr;
   unsigned int* carbMax    = myVars->carbADCMaxPtr;
   unsigned int* sulfMin    = myVars->sulfADCMinPtr;
   unsigned int* sulfMax    = myVars->sulfADCMaxPtr;
   unsigned int adcHead, myADC ;
   for(;;)
   {
      //location of current data
      adcHead = ((*myVars->userInBufHeadPtr)<<2) + *myVars->userInBufHeadPtr;

      //converting bcd values into int.
      myADC   = (*(myBufPtr+adcHead+2)*100)+ (*(myBufPtr+adcHead+3)*10) + *(myBufPtr+adcHead+4);
      //--------------------PROCESSING ADC MINIMUN VALS-----------------------------------------------------
        if(MIN == *(myBufPtr+ adcHead +1))
        {  //Temperature Limit
         *tempMin = (TEMP_ADC == *(myBufPtr+ adcHead)) ? myADC: *tempMin;

        //FLow Rate
         *flowMin = (FLOW_ADC == *(myBufPtr+ adcHead)) ? myADC: *flowMin;

        //Carbon Level
        *carbMin  = (CARB_ADC == *(myBufPtr+ adcHead)) ? myADC: *carbMin;

        //Sulfur Level
        *sulfMin  = (SULF_ADC == *(myBufPtr+ adcHead)) ? myADC: *sulfMin;
               }

      //--------------------PROCESSING ADC MAXIMUN VALS-----------------------------------------------------
        if(MAX == *(myBufPtr + adcHead +1))
        {
         *tempMax = (TEMP_ADC == *(myBufPtr+ adcHead)) ? myADC: *tempMax;

        //FLow Rate
          *flowMax = (FLOW_ADC == *(myBufPtr+ adcHead)) ? myADC: *flowMax;

        //Carbon Level
        *carbMax  = (CARB_ADC == *(myBufPtr+ adcHead)) ? myADC: *carbMax;

        //Sulfur Level
        *sulfMax  = (SULF_ADC == *(myBufPtr+ adcHead)) ? myADC: *sulfMax;
        }
        OSTaskSuspend(OS_PRIO_SELF);
   }
}

//This function process the limits data according to type.
//Read the buffer at current postion interpret type (the header)
//Put data into according variable
void far setLimitsTask(void* data)
{
   SetLimitsData* myVars = data;

   unsigned short* myBufPtr = myVars->userInputBufPtr;
   unsigned short* myTempL0 = myVars->tempL0Ptr;    //Temperature warn pointer
```

(userTask.c continue 5/7)

```
   unsigned short* myTempL1 = myVars->tempL1Ptr;    //Temperature alarm pointer
   unsigned short* myFlowL0 = myVars->flowL0Ptr;    //Flow Rate warn pointer
   unsigned short* myFlowL1 = myVars->flowL1Ptr;    //FLow Rate alarm pointer
   unsigned short* myCarbL0 = myVars->carbL0Ptr;    //Carbon Lvl warn pointer
   unsigned short* myCarbL1 = myVars->carbL1Ptr;    //Carbon Lvl alarm pointer
   unsigned short* mySulfL0 = myVars->sulfL0Ptr;    //Sulfur Lvl warn pointer
   unsigned short* mySulfL1 = myVars->sulfL1Ptr;    //Sulfur Lvl alarm pointer
   unsigned int limHead = 0;                        //Location of data in the buffer
   for(;;)
   {
      limHead = ((*myVars->userInBufHeadPtr)<<2) + *myVars->userInBufHeadPtr;
        //--------------------PROCESSING WARN LIMITS---------------------------------------------------------
        if(WARN == *(myBufPtr+limHead + 1))

        {  //Temperature Limit
         *myTempL0     = (TEMP_LIM == *(myBufPtr+ limHead)) ? *(myBufPtr+limHead+2): *myTempL0;
         *(myTempL0+1) = (TEMP_LIM == *(myBufPtr+ limHead)) ? *(myBufPtr+limHead+3): *(myTempL0+1);

        //FLow Rate
         *myFlowL0     = (FLOW_LIM == *(myBufPtr+ limHead)) ? *(myBufPtr+limHead+2): *myFlowL0;
         *(myFlowL0+1) = (FLOW_LIM == *(myBufPtr+ limHead)) ? *(myBufPtr+limHead+3): *(myFlowL0+1);

        //Carbon Level
        *myCarbL0     = (CARB_LIM == *(myBufPtr+ limHead)) ? *(myBufPtr+limHead+2): *myCarbL0;
        *(myCarbL0+1) = (CARB_LIM == *(myBufPtr+ limHead)) ? *(myBufPtr+limHead+3): *(myCarbL0+1);

        //Sulfur Level
        *mySulfL0     = (SULF_LIM == *(myBufPtr+ limHead)) ? *(myBufPtr+limHead+2): *(mySulfL0);
        *(mySulfL0+1) = (SULF_LIM == *(myBufPtr+ limHead)) ? *(myBufPtr+limHead+3): *(mySulfL0+1);
         }

      //--------------------PROCESSING ALARM LIMITS---------------------------------------------------------
        if(ALARM == *(myBufPtr+ limHead +1))
        {
        //Temperature Limt
        *myTempL1     = (TEMP_LIM == *(myBufPtr+ limHead)) ? *(myBufPtr+limHead+2): *myTempL1;
        *(myTempL1+1) = (TEMP_LIM == *(myBufPtr+ limHead)) ? *(myBufPtr+limHead+3): *(myTempL1+1);

        //FLow Rate
        *myFlowL1     = (FLOW_LIM == *(myBufPtr+ limHead)) ? *(myBufPtr+limHead+2): *myFlowL1;
        *(myFlowL1+1) = (FLOW_LIM == *(myBufPtr+ limHead)) ? *(myBufPtr+limHead+3): *(myFlowL1+1);

        //Carbon Level
        *myCarbL1     = (CARB_LIM == *(myBufPtr+ limHead)) ? *(myBufPtr+limHead+2): *myCarbL1;
        *(myCarbL1+1) = (CARB_LIM == *(myBufPtr+ limHead)) ? *(myBufPtr+limHead+3): *(myCarbL1+1);

        //Sulfur Level
        *mySulfL1     = (SULF_LIM == *(myBufPtr+ limHead)) ? *(myBufPtr+limHead+2): *mySulfL1;
        *(mySulfL1+1) = (SULF_LIM == *(myBufPtr+ limHead)) ? *(myBufPtr+limHead+3): *(mySulfL1+1);
        }
      OSTaskSuspend(OS_PRIO_SELF);
   }
}

void far sendValuesTask(void* myTCB)
{
   myTCB = myTCB; // Turns off Annoying Warning
        for(;;)
   {
      pio_init(9,2);
      while(1)
      {
         pio_wr(9,ON);
         delay_ms(2);
         pio_wr(9,OFF);
         delay_ms(1000);
      }

   }
}
/***********************************************************************************************************
*                      Missile Defense Task
*
* VERSION:             2.0
```

(userTask.c continue 672)

```
* PROJECT:            Glacial Monitoring System
* MODIFIED:           May 29 2008
* AUTHOR:             Justin Reina, Khoa Nguyen, Thuat Nguyen
***********************************************************************************************************/
void far missileDefenseTask(void* data)
{
 //MissileData*        myVars          = data;
 unsigned int* newMsgPtr      = NULL;
 data = data;
 newMsgPtr = newMsgPtr;
 for(;;)
 {
   delay_ms(2);
        OSSemPend(missileLaunch_Sem,WAIT_FOREVER,&nullErr);
   delay_ms(2);
   OSTaskSuspend(OS_PRIO_SELF);
 }
}
/***********************************************************************************************************
*                     Jims' Face Task
*
* VERSION:            2.0
* PROJECT:            Glacial Monitoring System
* MODIFIED:           May 29 2008
* AUTHOR:             Justin Reina, Khoa Nguyen, Thuat Nguyen
***********************************************************************************************************/
void far jimsFaceTask(void* data)
{
MY_OS_Q* Dpq = display_Event->OSEventPtr;
        data = data;
   for(;;)
   {
      OSSemPend(dispMboxWrite_Sem,WAIT_FOREVER,&nullErr);
      display_Dbox[Dpq->queueIndex] = JIMS_FACE;
      myOSQPost(display_Event,&display_Dbox[Dpq->queueIndex]);

      OSTimeDly(15*OS_TICKS_PER_SEC);
   }
}
```

Figure 36.User Task Source File

mainTest.java – Java Main Class

```
/*************************************************
 * MAIN TEST (class)
 *
 * Michael Beauchamp
 * University of Washington (Teaching Assistant)
 * EE 472 - Summer 2006
 *
 * Modified by Walker Robb
 * University of Washington (Teaching Assitnt)
 * EE 472 - Spring 2007
 *
 * Wrapper class for Port Open.
 **********************************************************************************************************/

import java.io.*;
import javax.comm.*;

public class MainTest {

        public static void main(String[] args) {

               PortOpen myPort = null;

               try {
                      myPort = new PortOpen("COM2"); // specify here the port you wish to connect to...
               } catch (IOException e) {

               } catch (NoSuchPortException e) {
```

(userTask.c continue 7/7)

```
                } catch (PortInUseException e) {

                } catch (UnsupportedCommOperationException e) {

                }

                SimpleGUI myGUI = new SimpleGUI(myPort);
                myGUI.setSize(500,325);
                myGUI.setVisible(true);
                myPort.setGUI(myGUI);
```

(mainTest.java continue)

```
                try {
                        myPort.converse();
                } catch (IOException e) {

                }
        }
}
```

Figure 37. Java Main Class

simpleGui.java – simpleGui Class

```
************************
 * Author Walker Robb
 * University of Washington
 * EE 472 TA - Spring 2007
 *
 * SimpleGUI (class)
 * Modified: Khoa Nguyen
 **********************************************************************************************************/

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class SimpleGUI extends JFrame  implements ActionListener
{
        protected boolean     initialOn   = false;                  // not initialized
        protected boolean     measureOn   = false;                  // measure is initially off
        protected boolean     dataLogOn   = false;                  // logging is initially off
        protected boolean     errMesgOn   = false;                  // send random error initially off

        protected JButton     initialBut  = new JButton("Initialize System");
        protected JButton     measureBut  = new JButton("Measure OFF");
        protected JButton     dataLogBut  = new JButton("Logging OFF");
        protected JButton     displayBut  = new JButton("Current Measurement");
        protected JButton     errMesgBut  = new JButton("Bad Frame");

        protected JPanel      aPanel0     = new JPanel();
        protected JLabel      aLabel0     = new JLabel("INPUT:");
        protected JLabel      aLabel1     = new JLabel("OUTPUT:");

        protected JTextField  textField0  = new JTextField(40);
        protected JTextArea   textArea0   = new JTextArea(12,40);
        protected JScrollPane scrollPane0 = new JScrollPane(textArea0);
        protected Container   aContainer;

        protected PortOpen    aPort;

        public SimpleGUI(PortOpen port)
        {
                super("Simple Interface");                          // name the panel
                addWindowListener(new WindowAdapter()               // close program if window closes
                {
                        public void windowClosing(WindowEvent e)
                        {
                                dispose();
                                System.exit(0);                     // calling the method is a must
                        }
                });

                aContainer = this.getContentPane();                 // configure the frame and add a panel
```

(simpleGui.java continue 1/2)

```
                aContainer.setLayout(new GridLayout() );
                aContainer.add(aPanel0);

                aPanel0.add(scrollPane0);
                scrollPane0.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);
                scrollPane0.setViewportBorder(BorderFactory.createLoweredBevelBorder());

                /**** KHOA's ButtOn START ****/

                initialBut.addActionListener(this);
                measureBut.addActionListener(this);
                dataLogBut.addActionListener(this);
                displayBut.addActionListener(this);
                errMesgBut.addActionListener(this);

                initialBut.setActionCommand("initializeme");
                measureBut.setActionCommand("measureme");
                dataLogBut.setActionCommand("logme");
                displayBut.setActionCommand("displayme");
                errMesgBut.setActionCommand("errorme");

                initialBut.setMnemonic(KeyEvent.VK_I);
                measureBut.setMnemonic(KeyEvent.VK_M);
                dataLogBut.setMnemonic(KeyEvent.VK_L);
                displayBut.setMnemonic(KeyEvent.VK_C);
                errMesgBut.setMnemonic(KeyEvent.VK_E);

                initialBut.setToolTipText("Initializes serial com link between system and remote com.");
                measureBut.setToolTipText("Start all tasks and perform measurements.");
                dataLogBut.setToolTipText("Stop any running measurement tasks.");
                displayBut.setToolTipText("Display the most recent measurements.");
                errMesgBut.setToolTipText("Generate a bad message.");

                initialBut.setBackground(Color.LIGHT_GRAY);
                measureBut.setBackground(Color.LIGHT_GRAY);
                dataLogBut.setBackground(Color.LIGHT_GRAY);
                displayBut.setBackground(Color.LIGHT_GRAY);
                displayBut.setBackground(Color.LIGHT_GRAY);

                aPanel0.add(initialBut);
                aPanel0.add(measureBut);
                aPanel0.add(dataLogBut);
                aPanel0.add(displayBut);
                //aPanel0.add(errMesgBut);

                /**** KHOA's ButtOn END ****/

                this.setVisible(true);                      // make sure the user can see it!
                aPort = port;                               // save a reference to the port we are using
        }

        public void actionPerformed(ActionEvent anEvent)
        {
                updateButtons(anEvent);                     // update each button's status
                aPort.setButtons(((JButton)anEvent.getSource()).getText());
                aPort.setSendFlag(true);

        String paramString = anEvent.paramString();          // print out string object information for debug
            System.out.println(paramString);
        }

        private void updateButtons(ActionEvent anEvent)
        {
                String actionStr = anEvent.getActionCommand();

                if ("measureme".equals(actionStr)) {
                        if (false == measureOn) {
                                measureOn = true;
                                measureBut.setText("Measure ON");
                                measureBut.setBackground(Color.GREEN);
                        } else {
                                measureOn = false;
                                measureBut.setText("Measure OFF");
                                measureBut.setBackground(Color.LIGHT_GRAY);
```

(simpleGui.java continue 2/2)

```
                        }
                } else if ("logme".equals(actionStr)) {
                        if (false == dataLogOn) {
                                dataLogOn = true;
                                dataLogBut.setText("Logging ON");
                                dataLogBut.setBackground(Color.GREEN);
                        } else {
                                dataLogOn = false;
                                dataLogBut.setText("Logging OFF");
                                dataLogBut.setBackground(Color.LIGHT_GRAY);
                        }
                }
        }

        // enter in new data WITH a return carriage
        public void enterDataNewLine(String data)
        {
                textArea0.append(data);
                textArea0.append("\n");
                textArea0.setCaretPosition(textArea0.getDocument().getLength());
        }

        // enter in new data WITHOUT a return carriage
        public void enterDataNoNewLine(String data)
        {
                textArea0.append(data);
                textArea0.setCaretPosition(textArea0.getDocument().getLength());
        }
}
```

Figure 38. SimpleGUI Class

PortOpen.java – PortOpen Class

```
* Port Open (class)
 *
 * Michael Beauchamp
 * University of Washington (Teaching Assistant)
 * EE 472 - Summer 2006
 *
 * Modified by Walker Robb
 * University of Washington (Teaching Assitant)
 * EE 472 - Spring 2007
 *
 * Connects to a serial port specified by user.
 * Modified: Khoa Nguyen
 **********************************************************************************************************/

import java.io.*;
import javax.comm.*;
import java.util.*;

import java.util.Date;
import java.text.DateFormat;
import java.text.SimpleDateFormat;

public class PortOpen implements SerialPortEventListener {

        public static final int   TIMEOUTSECONDS = 30;     // How long to wait for the open to finish up.
        public static final int   BAUD           = 9600;//19200;  // The baud rate to use.

        protected DataInputStream is;        // The input stream
        protected PrintStream     os;        // The output stream
        protected SimpleGUI       gui;       // GUI to interact with

        public    String  scannedInput;      // The complete input from the serial port
        public    String  readyOutput;       // Data to be sent

        boolean   sendFlag    = false;       // Data ready to send flag
        boolean   receiveFlag = false;       // Data ready to process flag

        CommPortIdentifier thePortID = null; // The chosen Port Identifier
        CommPort           thePort;          // The chosen Port itself
```

(portOpen.java continue 1 /9)

```
/***** VARIABLES added BEGINS *****/

/* All checksums are pre-computed using capitalized letter
** IChecksum = 0x01^0x30^0x30^0x30^0x39^0x49;       // 65
** SChecksum = 0x01^0x30^0x30^0x30^0x39^0x53;       // 91
** PChecksum = 0x01^0x30^0x30^0x30^0x39^0x50;       // 88
** DChecksum = 0x01^0x30^0x30^0x30^0x39^0x44;       // 76
** MChecksum = 0x01^0x30^0x30^0x30^0x39^0x4D;  // 69
** LChecksum = 0x01^0x30^0x30^0x30^0x39^0x4C;  // 68
*************************************************************/

private final int maxNakCount = 4;    // max value of nak received consecutively before I-frame

private char[] cArray;                // contains the array of char received from serial
private int nakCount  = 0;
private int seqNumber = 0x30;

boolean ackFlag = true;

private String lastCommand;
private String buttonPressed;
private String preLength = "\u0030\u0030\u0030\u0039";
private String StartUni  = "\u0001", EndUni = "\n";

private String IChecksum = "\u0036\u0035", SChecksum = "\u0039\u0031",
                          PChecksum = "\u0038\u0038", DChecksum = "\u0037\u0036",
                          MChecksum = "\u0036\u0039", LChecksum = "\u0036\u0038";

private String ackFrame  = "\u0001\u0006\u0030\n";   // 0x01,0x06,0x30,0x0A
private String nakFrame  = "\u0001\u0015\u0037\n";    // 0x01,0x15,0x37,0x0A

private String IFrame    = StartUni + preLength + "I" + IChecksum + EndUni;
private String SFrame    = StartUni + preLength + "S" + SChecksum + EndUni;
private String PFrame    = StartUni + preLength + "P" + PChecksum + EndUni;
private String DFrame    = StartUni + preLength + "D" + DChecksum + EndUni;
private String MFrame    = StartUni + preLength + "M" + MChecksum + EndUni;
private String LFrame    = StartUni + preLength + "L" + LChecksum + EndUni;

/***************************** VARIABLES added ENDS  ***********************************************/


/***************************************************************************************************
 * PORT OPEN (Constructor)
 *
 * The constructor is complete and you should not have to modify it, but you
 * should take a look at it and see how it works.  The input is a string
 * consisting of either "COM1" or "COM2".  The constructor sets up the serial
 * communicaion port including the baud rate, number of data bits, the stop
 * bits, parity, and flow control.  It also sets up the input stream, output
 * stream, and the event listener.
 **************************************************************************************************/
public PortOpen(String desiredPort)

throws IOException, NoSuchPortException, PortInUseException,
UnsupportedCommOperationException {

        System.out.println("Serial Port by Michael Beauchamp's.");
        System.out.println("based on Ian Darwin's book Java Cookbook\n");

        System.out.println("Creating list of available ports ... ");
        System.out.println("Looking for port " + desiredPort + " ... ");

        // Get list of ports on this particular computer by calling static
        // method in CommPortIdentifier.  The ports, type (serial or parallel), and
        // ownership are listed.  If found, the desired port is noted.
        Enumeration portEnum = CommPortIdentifier.getPortIdentifiers();

        while (portEnum.hasMoreElements()) {

                CommPortIdentifier cpi = (CommPortIdentifier) portEnum.nextElement();

                System.out.print("\tPort " + cpi.getName());
```

(portOpen.java continue 2/9)

```
                        if ( cpi.getPortType() == CommPortIdentifier.PORT_SERIAL)
                                System.out.print(" - Serial Port   ");
                        else if ( cpi.getPortType() == CommPortIdentifier.PORT_PARALLEL)
                                System.out.print(" - Parallel Port ");
                        else
                                System.out.print(" - UNKNOWN Port  ");

                        if ( cpi.isCurrentlyOwned() )
                                System.out.println("Owned");
                        else
                                System.out.println("Unowned");

                        if ( desiredPort.compareTo(cpi.getName()) == 0 )
                                thePortID = cpi;
                }

                if (thePortID != null) // was desired port found?
                        System.out.println("Found port " + thePortID.getName() + " ... ");
                else
                        throw new IllegalStateException("ERROR: No such port found.");

                if (thePortID.getPortType() != CommPortIdentifier.PORT_SERIAL )     // is it a serial port?
                        throw new IllegalStateException("ERROR: Selected port is not a SERIAL port.");

                if ( thePortID.isCurrentlyOwned())    // is serial port in use?
                        throw new IllegalStateException("ERROR: Selected port is in use.");

                // Open the port. openPort takes an Application Name and a timeout.
                System.out.print("Trying to open port " + thePortID.getName() + " ... ");
                thePort = thePortID.open("EE472 DataComm", TIMEOUTSECONDS * 1000);

                SerialPort myPort = (SerialPort)thePort;
                System.out.println("Done");

                // set up the serial port
                // BaudRate
                // DataBits
                //              DATABITS_5      5 data bit format.
                //      DATABITS_6      6 data bit format.
                //      DATABITS_7      7 data bit format.
                //      DATABITS_8      8 data bit format.
                // StopBits
                //      STOPBITS_1      Number of STOP bits - 1.
                //      STOPBITS_1_5     Number of STOP bits - 1-1/2.
                //      STOPBITS_2      Number of STOP bits - 2.
                // Parity
                //      PARITY_EVEN      EVEN parity scheme.
                //      PARITY_MARK      MARK parity scheme.
                //      PARITY_NONE      No parity bit.
                //      PARITY_ODD      ODD parity scheme.
                //      PARITY_SPACE            SPACE parity scheme.
                System.out.print("Setting " + thePortID.getName() + "s parameters ... ");
                try {
                        myPort.setSerialPortParams(BAUD,
                                        SerialPort.DATABITS_8,
                                        SerialPort.STOPBITS_1,
                                        SerialPort.PARITY_NONE);
                } catch (UnsupportedCommOperationException e) {
                        System.out.println("FAILED");
                        throw new UnsupportedCommOperationException(
                                                                "ERROR: Incorrectly specified parameters.");
                }

                //  Set up the flow control.
                //  FLOWCONTROL_NONE  Flow control off.
                //  FLOWCONTROL_RTSCTS_IN RTS/CTS flow control on input.
                //  FLOWCONTROL_RTSCTS_OUT RTS/CTS flow control on output.
                //  FLOWCONTROL_XONXOFF_IN XON/XOFF flow control on input.
                //  FLOWCONTROL_XONXOFF_OUT XON/XOFF flow control on output.
                try {
                        myPort.setFlowControlMode(SerialPort.FLOWCONTROL_NONE);
                } catch (UnsupportedCommOperationException e) {
                        System.out.println("FAILED");
                        throw new UnsupportedCommOperationException(
```

(portOpen.java continue3/9)

```
                                                                "ERROR: Incorrectly specified flowcontrol.");
                }

                System.out.println("Done");

                // Attempt to get the input stream.
                System.out.print("Getting input stream ... ");
                try {
                        is = new DataInputStream(thePort.getInputStream());
                } catch (IOException e) {
                        is = null;
                        System.out.println("FAILED");
                        throw new IOException("ERROR: Can not open input stream.");
                }
                System.out.println("Done");

                // Create the output stream.
                System.out.print("Creating output stream ... ");
                os = new PrintStream(thePort.getOutputStream(), true);
                System.out.println("Done");

                // Add the event listener.
                System.out.print("Adding event listener ... ");
                try {
                        myPort.addEventListener(this);
                } catch (TooManyListenersException e) {
                        System.out.println("FAILED");
                        throw new IllegalStateException("ERROR: Too many listeners.");
                }
                myPort.notifyOnDataAvailable(true);
                System.out.println("Done");

                // Now ready to read and write to the serial port.
                System.out.println("\nReady to read and write port.\n");
        }

        /****************************************************************************************************
         * CONVERSE
         *
         * The is the main block of the code that will control the serial
         * communication.  The data ready flag is set in the serial event, which
         * takes information from the input stream and places it in the
         *  scannedInput.  Based on the input you can write to the output stream
         * using "os.write".
         **************************************************************************************************/
        protected void converse() throws IOException {

                boolean quit         = false;
                int            arrayLength   = 0;

                while (!quit) {

                        // The data ready flag is set by the serial event when there is data in the
                        // read buffer. THIS IS WHERE WE receive DATA from Tern Serial I/O to Java Comm
                        if (receiveFlag) {

                                cArray      = scannedInput.toCharArray();    // make this into char array
                                arrayLength = cArray.length;                 // compute the length, need later

                                if ((int)cArray[0] == 0x01)           // is first character a "start header"
                                {
                                        int arrayIndex1 = cArray[1];  // for c-frame, 0x06 = ACK, 0x15 = NAK

                                        //             if NAK was received || array length < 9
                                        //                     increment nakCount
                                        //                     if nakCount == 4
                                        //                             send an IFrame to Tern board
                                        //                     else
                                        //                             resend the last command
                                        //             else if ACK was received
                                        //                     set ackFlag  = true
                                        //                     set nakCount = 0
                                        //             else if predict whether Iframe was sent
                                        //                     call parseData with char array
```

(portOpen.java continue4/9)

```
               //      if error
               // invalid dat
               //                             send NAK
               //                     else
               //                             send ACK
               //                             call displayData
               //              else
               //                     send NAK
               //              set sendFlag false

               if (arrayIndex1 == 0x15) {     // check for a NAK response


                       nakCount++;
                       if (nakCount == maxNakCount)
                       {              // max NAK reached, re-initialize serial com
                               if (gui != null){
                                      gui.enterDataNewLine("NAK has reached max of 4
                                                             or greater tries.");
                               }
                               readyOutput = IFrame;  // since it's a NAK == 4,
                                                      //send an I-frame
                               os.print(readyOutput); // response from I-frame is
                                                      //"weird" --> nothing
                               sendFlag = true;
                               nakCount = 0;
                       }

                       readyOutput = lastCommand;     // this was saved previously
                       os.print(readyOutput);         // resending the last command
                       sendFlag = true;

                       if (gui != null){
                               gui.enterDataNoNewLine("NAK received: " + scannedInput);
                       }
               } else if (arrayIndex1 == 0x06){       // ACK was received
                       ackFlag = true;                        // set actFlag to true
                       nakCount = 0;                          // reset the nakCounter
                       if (gui != null){
                               gui.enterDataNoNewLine("ACK received: " + scannedInput);
                       }
               } else if (arrayLength >= 9 && (cArray[5] == 'M' ||
                                               cArray[5] == 'W' || cArray[5] == 'E')){
                       if (parseData(cArray)){
                               gui.enterDataNewLine("ACK sent
                                             (response recognized): " + scannedInput);
                               sendACK();     // length & checksum are correct
                               displayAndCommand(cArray);
                       } else {
                               gui.enterDataNewLine(
                                 "NAK sent (unrecognized response): " + scannedInput);
                               sendNAK();     // length and/or checksum wrong

                       }
               } else { // MIGHT NEED TO BE REMOVED
                       gui.enterDataNewLine(
                               "NAK sent (something is very wrong): " + scannedInput);
                       sendNAK();
               }
        } else {
               gui.enterDataNoNewLine("NAK sent
                                       (first byte is wrong): " + scannedInput);
               sendNAK();
        }
        receiveFlag = false;
}

// 1. Determine button pressed
// 2. Display the determined command on the gui text area
// 3. Set readyOutput = to frame type
// 4. Print readyOutput to serial out
// 5. Set ackFlag = false, sendFlag = false
if (sendFlag) {
        if (ackFlag)
```

(portOpen.java continue5/9)

```
                       {
                              if (buttonPressed.equals("Initialize System")) {
                                      if ( gui != null)
                                             gui.enterDataNewLine(getDateTime()
                                              + " - System is initialized.");
                                      readyOutput = IFrame;
                              } else if (buttonPressed.equals("Measure ON")) {
                                      if ( gui != null)
                                             gui.enterDataNewLine(getDateTime()
                                             + " - Measuring is turned ON.");
                                      readyOutput = SFrame;
                              } else if (buttonPressed.equals("Measure OFF")) {
                                      if ( gui != null)
                                             gui.enterDataNewLine(getDateTime()
                                              + " - Measuring is turned OFF.");
                                      readyOutput = PFrame;
                              } else if (buttonPressed.equals("Logging ON")) {
                                      if ( gui != null)
                                             gui.enterDataNewLine(getDateTime()
                                              + " - Logging is turned ON.");
                                      readyOutput = DFrame;
                              } else if (buttonPressed.equals("Logging OFF")) {
                                      if ( gui != null)
                                             gui.enterDataNewLine(getDateTime()
                                               + " - Logging is turned OFF.");
                                      readyOutput = LFrame;
                              } else if (buttonPressed.equals("Current Measurement")) {
                                      if ( gui != null)
                                             gui.enterDataNewLine(getDateTime()
                                            + " - Display the most recent measurements.");
                                      readyOutput = MFrame;
                              } else if (buttonPressed.equals("Bad Frame")) {
                                      if ( gui != null)
                                             gui.enterDataNewLine(getDateTime()
                                               + " - Generating an error. Response should be NAK.");
                                      readyOutput = "\u0001khoanguyen\n";

                              }
                              lastCommand = readyOutput;
                              os.print(readyOutput);

                              ackFlag     = false;
                              sendFlag    = false;
                       }
               }
        }
        is.close();           // Clean up streams
        os.close();

        System.out.println("\nInput and output streams closed.");
}


/********************************************************************************
 * Description: parses the length, checksum, make sure they match
 * Parameter:   the char array of response
 * Returns:     true if length & checksum match, false otherwise
 *******************************************************************************/
private boolean parseData(char[] value)
{
        int lengthReceived   = cArray.length;               // length for M-response = 30
        int checksumFrame    = 0;                           //  for W-response = 26
        int checksumComputed = 0;
        int lengthComputed   = 0;


        int indexInt1 = asciiToInt(cArray[1]);              // integers of the LENGTH frame
        int indexInt2 = asciiToInt(cArray[2]);
        int indexInt3 = asciiToInt(cArray[3]);
        int indexInt4 = asciiToInt(cArray[4]);

        int checkByte1 = ((int)cArray[lengthReceived-3]);
        int checkByte2 = ((int)cArray[lengthReceived-2]);
```

(portOpen.java continue 6/9)

```
                checkByte1 = checkByte1<<8;
                checksumFrame = checkByte1|checkByte2;

                // int lastThreeIndex = asciiToInt(cArray[lengthReceived-3]);  //first byte of the CHECKSUM
                // int lastTwoIndex   = asciiToInt(cArray[lengthReceived-2]);  // second byte

                // the length that Serial I/O task computed and placed in IFrame
                // the checksum calculated from the checksum frame received
                // checksumFrame  = 16*lastThreeIndex + 1*lastTwoIndex;
                lengthComputed = 4096*indexInt1    + 256*indexInt2 + 16*indexInt3 + 1*indexInt4;


                char myChar = cArray[5];      // response type in index5

                // Measure, Warn, Logging, Error responses
                if (myChar != 'M' || myChar != 'W' || myChar != 'E'){
                        return false;
                }

                // We compute the checksum here - yup!
                for (int i = 0; i < lengthReceived - 3; i++){
                        checksumComputed = checksumComputed^cArray[i];
                }

                // If lengths and checksums match, that is good, continue to explore
                if (lengthComputed == lengthReceived && checksumComputed == checksumFrame ){
                        return true;
                }

                // Something didn't fan out, the hardware must be broken
                return false;
        }

        /****************************************************************************************************
         * Description: Display the data from IFrame body
         * Parameter:   IFrame in char array
         * Returns:     none, make pretty pictures
         ***************************************************************************************************/

        private void displayAndCommand(char[] c)
        {
                switch (c[5])
                {
                        case 'M':
                                gui.enterDataNewLine(getDateTime() + "Showing most recent measurements.");
                                gui.enterDataNewLine("Temperature   [Far]: " + c[7]  + c[8]  + c[9]  + c[10]);
                                gui.enterDataNewLine("Flow Rate     [L/s]: " + c[11] + c[12] + c[13] + c[14]);
                                gui.enterDataNewLine("Carbon Level  [ppb]: " + c[15] + c[16] + c[17] + c[18]);
                                gui.enterDataNewLine("Sulfur Level  [ppm]: " + c[19] + c[20] + c[21] + c[22]);
                                gui.enterDataNewLine("Thermal Image [FS ]: " + c[23] + c[24] + c[25] + c[26]);
                                gui.enterDataNewLine("Digital Flow  [L/s]: " + c[27] + c[28] + c[29] + c[30]);
                                break;

                        case 'W':
                                gui.enterDataNewLine(getDateTime() + "WARNING: One or more limits exceeded.");
                                gui.enterDataNewLine("Temperature   [Far]: " + c[7]  + c[8]  + c[9]  + c[10]);
                                gui.enterDataNewLine("Flow Rate     [L/s]: " + c[11] + c[12] + c[13] + c[14]);
                                gui.enterDataNewLine("Carbon Level  [ppb]: " + c[15] + c[16] + c[17] + c[18]);
                                gui.enterDataNewLine("Sulfur Level  [ppm]: " + c[19] + c[20] + c[21] + c[22]);
                                break;

                        case 'E':
                                gui.enterDataNewLine("Error frame was received. Command not recognized.");

                                break;

                        default:
                                gui.enterDataNoNewLine("Default case was executed: " + scannedInput);
                                break;
                }
        }

        /***************************************************************************************************
         * Description: send an ACK frame to Tern Board, response valid
```

(portOpen.java continue 7/9)

```
 * Parameter:   none
 * Returns:     none
 *************************************************************************************************/
private void sendACK()
{
        seqNumber = (seqNumber)%8;
        readyOutput = ackFrame;
        os.print(readyOutput);
        sendFlag = true;
}

/**************************************************************************************************
 * Description: send an NAK frame to Tern Board, response was invalid
 * Parameter:   none
 * Returns:     none
 **************************************************************************************************/
private void sendNAK()
{
        seqNumber   = (seqNumber)%8;
        readyOutput = nakFrame;
        os.print(readyOutput);
        sendFlag = true;
}

/**************************************************************************************************
 * Description: Converts an integer to an ASCII character
 * Parameter:   integer, MUST ensure integers between 0-15
 * Returns:     a char array of length 1, (0-9,A-F) in uppercase
 **************************************************************************************************/
private char[] intToAscii(int value)
{
        String hexInt = Integer.toHexString(value);  // char array of length 1
        return hexInt.toUpperCase().toCharArray();    // valid: 0-15 (0-9,A-F)
}

/**************************************************************************************************
 * Description: Converts an ascii character to an integer
 * Parameter:   char, MUST ensure between 0-9, A-F uppercase ONLY
 * Returns:     the corresponding integer value between 0-15
 **************************************************************************************************/
private int asciiToInt(char value)
{
        int tempInt = (int)value;
        if (tempInt >= 0x30 && tempInt <= 0x39){
                return (tempInt - 0x30);
        } else if (tempInt >= 0x41 && tempInt <= 0x46){
                if (tempInt == 0x41){
                        return 10;
                } else if (tempInt == 0x42){
                        return 11;
                } else if (tempInt == 0x43){
                        return 12;
                } else if (tempInt == 0x44){
                        return 13;
                } else if (tempInt == 0x45){
                        return 14;
                } else if (tempInt == 0x46){
                        return 15;
                }
        }
        return -1; // the value was not in range
}

/**************************************************************************************************
 * Description: Provides the current date and time of the machine
 * Parameter:   none
 * Returns:     a formatted date/time string "MM/dd/yyyy hh:mm aa" (AM/PM)
 **************************************************************************************************/

private String getDateTime() {
        DateFormat dateFormat = new SimpleDateFormat("MM/dd/yyyy hh:mm aa");
        Date date = new Date();
        return dateFormat.format(date);
}
```

(portOpen.java continue8/9)

```
        protected void setButtons(String value)  { buttonPressed = value; } // set the button pressed

        public void setSendFlag(boolean value)   { sendFlag = value; }     // set new value for sendFlag
        protected void setSendData(String value) { readyOutput = value; }  // set new data for readyOutput
        protected void setGUI(SimpleGUI object)  { gui = object; }         // set the GUI to display stuff

        /****************************************************************************************************
         * SERIAL EVENT
         *
         * Serial event is an event listener that will take data, as available,
         * from the input stream and place it in a string called scannedInput.
         * ScannedInput is what you want to parse in Converse.  When the end of
         * the data has been received (a newline character -> 0x0A) the receiveFlag
         *  flag will be set.  Depending on how you set up your code you might
         *  want to modify this code.
         ***************************************************************************************************/
        public void serialEvent(SerialPortEvent event) {

                int FRAME_END = 0x0A;

                switch(event.getEventType()) {
                case SerialPortEvent.BI:
                case SerialPortEvent.OE:
                case SerialPortEvent.FE:
                case SerialPortEvent.PE:
                case SerialPortEvent.CD:
                case SerialPortEvent.CTS:
                case SerialPortEvent.DSR:
                case SerialPortEvent.RI:
                case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
                        break;
                case SerialPortEvent.DATA_AVAILABLE:
                        StringBuffer readBuffer = new StringBuffer();
                        int c;
                        try {

                                // Read the transmission into a string buffer until the end
                                // of the transmission is found.
                                while ( ( c = is.read() ) != ( (char) FRAME_END ) ) {
                                        readBuffer.append((char) c);
                                }

                                // Read the last char of the transmission into a string buffer.
                                readBuffer.append((char) c);

                                // Once the end of the transmission has been found, convert
                                // the string buffer into a string and set the data ready
                                // flag.
                                scannedInput = readBuffer.toString();
                                receiveFlag  = true;

                        } catch (IOException e) {}
                        break;
                }
        }
}
```

Figure 39. PortOpen Class

main.h – Main Task Header File for the MicroController (AVR)

```
#define F_CPU                 18432000
#define BUF_SIZE 40

//Keypad Pin Assigments
#define ENTER                 14
#define SEND                  12
#define A_BUTTON              3
#define B_BUTTON              7
#define C_BUTTON              11
#define D_BUTTON              15
#define NO_KEY_PRESSED        -1
```

(portOpen.java continue 9/9)

```
//PORT ASSIGNMENTS
#define RTS_PIN               2
#define RTS_PORT              'B'
#define CTS_PIN               3
#define CTS_PORT              'B'
#define DATA_START            4
#define DATA_PORT             'B'

#define PING_PIN              2
#define PING_PORT             'D'

#define DIG_FLOW_PIN          3
#define DIG_FLOW_PORT         PORTD
#define PING_OUT_PIN          4
#define PING_OUT_PORT        'D'

#define SPKR_CLK_PIN          0
#define SPKR_CLK_PORT       PORTB

//LED PORT ASSIGNMENTS
#define GR_LED_PIN            5
#define YELLOW_LED_PIN 6
#define RED_LED_PIN           7
#define LED_PORT              PORTC

//ALARMSTATE DEFINITIONS
#define LED0_PIN              0
#define LED1_PIN              1
#define STATE0_PIN            2
#define STATE1_PIN            3
#define SPKR_STATE_PIN  4
#define ALARM_PORT            PORTC

#define GLACIAL_PERIOD_MIN    35
#define TIMER1_COMP_VAL       OCR1A
#define LED_PRESCALER         40

#define RTS_ON  0
#define RTS_OFF 1

#define ON     1
#define OFF    0

#define TIMER1_INIT_COUNT     17500

#define INITIAL_GLACIAL_DEPTH 1000    //2000m

//Glacial Definitions
#define GLACIAL_PING_VELOCITY 3500   // m/s
#define GLACIAL_MELT_RATE             120    // m/min or 2m/s
extern unsigned int myJustin;
extern unsigned short  packetBuf[BUF_SIZE][10];
extern unsigned char   tempCol,tempRow;
extern unsigned int    packetBufHead;
extern unsigned int    packetBufTail;

extern unsigned char   keyPadVals [16];
extern int             keyPadNums[16];

extern unsigned short  identifiers[5];


extern unsigned short  currType, warnAlarm;

extern unsigned int    i;
extern int             myRow, myCol, myPoll, RTS, CTS;

extern unsigned int    currUserState;

unsigned short         digFreq[3];

unsigned long  currGlacialPingPeriod;
```

```
//GLOBAL VARS II--------------------------------------------------------------------------------------------
-
extern unsigned int   tempRaw,
                      flowRaw,
                      carbonRaw,
                      sulfurRaw;

extern unsigned short tempADC_Max[3],
                      tempADC_Min[3],
                      flowADC_Max[3],
                      flowADC_Min[3],
                      carbonADC_Max[3],
                      carbonADC_Min[3],
                      sulfurADC_Max[3],
                      sulfurADC_Min[3];

extern unsigned short tempL0[2],
                      tempL1[2],
                      flowL0[2],
                      flowL1[2],
                      carbonL0[2],
                      carbonL1[2],
                      sulfurL0[2],
                      sulfurL1[2];

extern unsigned short digitalFlowRate[3];
extern unsigned short glacialDepth[3];

extern unsigned int   newInput;
extern unsigned int   tempUserData[20];
extern unsigned int   tempUserCount;

extern int            x, rePrint;

extern unsigned char  printStatusBuf[100];                         //Buffer Used to store information for
printStatusBuf()
extern unsigned int   printStatusBufPtr;                    //Index Of the '\0' Character

extern unsigned int   pingFlag;
extern unsigned short digitalFlowConvRate[3];

extern unsigned short  simGlacialDepth[3];
unsigned int          tempGlacialDepthBuf[3];
extern unsigned int   simGlacialDepthInt;

unsigned short        digTransdFreq[3];

//PROTOTYPES------------------------------------------------------------------------------------------------
void           ser0Str(char*);
void           charToStr(unsigned char, unsigned char*);
unsigned int   pollKeyPad(void);

void           setPinB(int,int);
void           flipPinB(int);

void           USART0_Init(void);
void           USART0_Transmit(unsigned char);

unsigned char  intToASCII(unsigned int);
void           announceBuffer(void);
void           checkTern(void);
void           checkUser(void);

unsigned int   readPin(unsigned char, unsigned int);
void           writeNibble(unsigned char, unsigned int, unsigned short);

void           checkForIdentifier(int);
void           checkForAlarmWarn(int);
int            checkForData(int);
void           setPin(unsigned char, int, int);
void           ternInput(void);
int            ternOutput(void);
```

```c
void           print2(unsigned short*);
void           print3(unsigned short*);

void           handleNewLimit(void);

void           checkAlarmState(void);
```

Figure 40. Main Task Header File for the AVR

main.c – Main Task Source File for the MicroController (AVR)

```c
//PRE-PROC--------------------------------------------------------------------------------------------------
#include "main.h"
#include "myAVR.h"
#include "menuHandler.h"

#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <avr/io.h>
#include <avr/iom324.h>
#include <avr/wdt.h>

unsigned char tempChars[25];
//GLOBAL VARS-----------------------------------------------------------------------------------------------
unsigned short        packetBuf[BUF_SIZE][10];
unsigned char  tempCol,tempRow;
unsigned int   packetBufHead = 0;
unsigned int   packetBufTail = 0;

unsigned char  keyPadVals [16] = {'1','2','3','A','4','5','6','B','7','8','9','C','*','0','#','D'};
int            keyPadNums [16] = { 1 , 2 , 3 , 10, 4 , 5 , 6 , 11, 7 , 8 , 9 , 12, -1, 0 ,-1 , 13};

unsigned short identifiers[5] = {0x0, 0x2, 0x3, 0x4,0x5};

unsigned short currType = 0, warnAlarm = 0;

unsigned int   i = 0;
int            myRow, myCol, myPoll, RTS =0, CTS = 0;

unsigned int   currUserState =0;

//GLOBAL VARS II--------------------------------------------------------------------------------------------
unsigned int   tempRaw   = 200,
               flowRaw   = 48,
               carbonRaw = 345,
               sulfurRaw = 290;

unsigned short tempADC_Max[3]   = {1,2,5},
               tempADC_Min[3]   = {1,2,6},
               flowADC_Max[3]   = {1,2,7},
               flowADC_Min[3]   = {1,2,8},
               carbonADC_Max[3] = {1,2,9},
               carbonADC_Min[3] = {1,3,9},
               sulfurADC_Max[3] = {1,3,0},
               sulfurADC_Min[3] = {1,3,1};

unsigned short tempL0[2]   =  {4,8},
               tempL1[2]   =  {2,9},
               flowL0[2]   =  {3,5},
               flowL1[2]   =  {2,7},
               carbonL0[2] =  {8,8},
               carbonL1[2] =  {6,5},
               sulfurL0[2] =  {0,2},
               sulfurL1[2] =  {2,4};

unsigned short digitalFlowRate[3] = {4,5,0};
unsigned short glacialDepth[3]    = {9,9,9};




unsigned short  digitalFlowConvRate[3]   = {9,9,9};
```

(main.c[AVR] continue 1/5)

```
unsigned int    digitalFlowConvRateInt   =  999;
unsigned short  simdigitalFlowRate[3]    = {1,1,1};

unsigned short  digFreq[3] = {9,9,9};

unsigned short  simGlacialDepth[3]    = {1,7,5};
unsigned int    tempGlacialDepthBuf[3] ={0,0,0};
unsigned int    simGlacialDepthInt    = 42;

unsigned int    newInput = 0;
unsigned int    tempUserData[20];
unsigned int    tempUserCount = 0;

int            x = 0, rePrint = 0;

unsigned char  printStatusBuf[100];                  //Buffer Used to store information for printStatus()
unsigned int   printStatusBufPtr = 0;                //Index Of the '\0' Character

unsigned int   pingFlag = 0;                         //For Troubleshooting: Delete...
unsigned int   LEDType  = 0;
unsigned int   LEDState = 0;
unsigned int   myLEDScaler = LED_PRESCALER;
unsigned short digTransdFreq[3] = {5,0,0};
unsigned long  currGlacialPingPeriod = INITIAL_GLACIAL_DEPTH;

//MAIN------------------------------------------------------------------------------------------------------
unsigned char tempCharsThatSuck[20];

int main(void)
{
        tempCol = 0;
        tempRow = 0;
        printStatusBuf[0] = '\0';                                          //Initialize The Buffer To 'Empty'
        rePrint = 0;

//PORT DECLARATIONS----------------------------------------------------------
    //PORTA is KeyPad IO; A0-A7 Are lined up according to keypad schematic
        //     A0-A3. Keypad Row Inputs
        //  A4-A7. Keypad Col Inputs
        DDRA  = 0x0F; // Top Half Are Outputs, to the keypad rows?
        PORTA = 0x0F; // A0-3 Are Initialized High
        PINA  = 0x00; // Bottom Half Are Initialized low

        //PORT B is the timers and the TERN COM
        //     B0. Timer0 - Used For the LED Clock (~2 Hz)
        //     B1. Timer1 - Used For the Speaker Frequency (~5kHz)
        //     B2. RTS Pin
        //     B3. CTS Pin
        //     B4. Data0
        //     B5. Data1
        //     B6. Data2
        //     B7. Data3
        DDRB  = 0b11110111;                          //11110111 <- Pin 3 Is Input
        PINB  = 0x00;                                //CTS is tied low
        PORTB = 0b00000100;                          //All Outputs initial Low

        //PORTC is the AlarmStatus Port.
        //     C0. LED0
        //     C1. LED1
        //     C2.     STATE0
        //     C3.     STATE1
        //     C4.     SPEAKER STATE
        //     C5.     GREEN LED
        //     C6.     YELLOW LED
        //     C7.     RED LED
        DDRC  = 0b11100000;
        PORTC = 0x00;
        PINC  = 0x00;


        //PORTD currently uses the USART0
        //     D0. RXD0
        //     D1.     TXD0
        //     D2. Glacial Ping Line         (Requires ~83 Ohm Resistor To Vcc)
```

(main.c[AVR] continue 2/5)

```
        //      D3.     Digital Flow Simulation
        //      D4. Glacial LED (Reserv For Future Use)
        //      D5. Busy Pin (RTS Decode Indicator)
        DDRD  = 0b11111000;
        PORTD = 0x00;
        PIND  = 0x00;

//INITIALIZATIONS----------------------------------------------------------------------
        USART0_Init();                              //USART0 Intialization
        timer0_Init();                              //Timer0 (8-Bit) For LED CLK
        timer1_Init();                              //Timer1 (16-Bit) For Digital Flow Sim
        timer2_Init();                              //Timer2 (8-Bit) For SPKR CLK
        int0_Init();                                //Initializze the Glacial Ping Interrupt.
        sei();                                      //Enable Global Interrupts

        lab4Preamble();
        checkAlarmState();

//EXECUTION LOOP-----------------------------------------------------------------------
        while (1) {
        //The Code Shall Continually Execute The Loop. There are 3 primary serviceable tasks, all with flags:
        //      1. Reprint the Menu. This shall call printMain(), and reset it's flag. It is also responsible
         //        for reprinting the current user prompt and status
        //      2. Tend to The Tern. It needs to continually check the Tern
        //      3. Check the User For an input
        //         !RTS Defines the AVR  Sending Mode
        //         !CTS Defines the Tern Sending Mode
                checkAlarmState();

                //1. Reprinting the menu
                if(rePrint)
                {
                        printMain();   //Print the Main Section
                        printStatus(); //Print the current Prompt
                                        //(A Buf of Chars Terminated with a Null - 'printStatusBuf')
                        rePrint = 0;   //Reset Flag
                }

                //      2.Send Data If You Have It (Should Exit if !CTS)
                if(RTS)
                {
                        writeNibble(DATA_PORT,DATA_START,packetBuf[packetBufHead][1]);
                        setPin(RTS_PORT,RTS_PIN,RTS_ON);      //Assert Busy Pin <-! Place Inside ternOutput...
                        if(readPin(CTS_PORT,CTS_PIN))
                                ternOutput();                                //Empty Out Buffer
                }

                //      3. Check User (Should Run Straight Through if none are pressed)
                x = pollKeyPad();
                if(x != NO_KEY_PRESSED)
                        processKey(x);
        }
}

int ternOutput() //1 means he's done, 1 means no CTS yet...
{
   int currCol = 0;

   while(packetBufHead !=packetBufTail)
   {
        //ser0Str("\n\rHead: ");
        //USART0_Transmit(packetBufHead+'0');
        for(currCol = 1; currCol <= (packetBuf[packetBufHead][0]+1); currCol++)
        {
                //Put Out First Data, Assert RTS--------------------------
                if(currCol<=packetBuf[packetBufHead][0])
                        writeNibble(DATA_PORT,DATA_START,packetBuf[packetBufHead][currCol]);
                else
                {
                        if(packetBufHead == (packetBufTail-1))
                                writeNibble(DATA_PORT,DATA_START,0);
                                //ser0Str("\n\rNo More\n\r");
                        else
                                writeNibble(DATA_PORT,DATA_START,1);
```

(main.c[AVR] continue 3/5)

```
                }

                setPin(RTS_PORT,RTS_PIN,RTS_ON);

                //Wait For CTS to Go High----------------------------------
                while(!readPin(CTS_PORT,CTS_PIN));    // Wait for CTS to go High

                //Acknowledge with RTS Low-----------------------------
                setPin(RTS_PORT,RTS_PIN,RTS_OFF);              //Ack By Setting RTS Low

                //Wait For CTS to Go Low-------------------------------
                while(readPin(CTS_PORT,CTS_PIN));      //Wait for Tern Ack   (means he has read it) CTS Goes
Low

        }
        packetBufHead = (packetBufHead+1)%BUF_SIZE;
   }

   if(packetBufHead == packetBufTail)
                RTS =0;
        return 0;
}

//ALARM HANDLER-------------------------------------------------------------------------
void checkAlarmState()
{
        char myChars[20];
        static int      oldSpkrState   = 3,             //Records the prev spkrstate
                                oldLEDState    = 3,     //Records the prev LEDstate
                                oldLEDType             = 3,    //Records the prev LEDType
                                oldRead                = 0x1F; //Records The Prev Value
        int myRead;


        myRead = (PINC&0x1F);                           //Read in PinC (and Mask Port Bits)

        //INITIAL TEST----------------------------------------------------------------------
        if(myRead != oldRead)
        {
                oldRead = myRead;                       //Set oldRead

                //Speaker Check------------------------------------------
                if( (myRead&(1<<4))>>4 != oldSpkrState)//Speaker
                {
                        oldSpkrState = (myRead&(1<<4))>>4;

                        if(!oldSpkrState)               //A Zero Designates Off
                                TIMSK2 = 0;             //Turn it Off!
                        else
                                TIMSK2 |= (1 << OCIE2A);        //Turn it On!
                }

                //LED State Check----------------------------------------
                if((myRead&(3<<2))>>2 !=oldLEDState)                    //State
                {
                        oldLEDState = (myRead&(3<<2))>>2;
                        switch(oldLEDState)
                        {
                                case 0:
                                        TIMSK0 = 0;                     //Mask Flashing
                                        switch(LEDType)
                                        {
                                                case 0:
                                                        PORTC |=  (1 << 5);
                                                        PORTC &= ~(1<<6 | 1<<7);
                                                        break;
                                                case 1:
                                                        PORTC |= (1 << 6);
                                                        PORTC &= ~(1<<5 | 1<<7);
                                                        break;
                                                case 2:
                                                        PORTC |= (1 << 7);
                                                        PORTC &= ~(1<<6 | 1<<5);
                                                        break;
```

(main.c[AVR] continue 4/5)

```
                                        }
                                        break;
                                case 1:
                                        myLEDScaler = LED_PRESCALER<<1;
                                        TIMSK0 |= (1 << TOIE0);
                                        break;
                                case 2:
                                        myLEDScaler = LED_PRESCALER;
                                        TIMSK0 |= (1 << TOIE0);
                                        break;
                        }
                }
                //LED Type Check
                if((myRead&3) != oldLEDType)
                {
                        oldLEDType = (myRead&3);
                        switch(oldLEDType)
                        {
                                case 0:
                                        LEDType = 0;
                                        PORTC |=  (1 << 5);
                                        PORTC &= ~(1<<6 | 1<<7);

                                        break;
                                case 1:
                                        LEDType = 1;
                                        PORTC |= (1 << 6);
                                        PORTC &= ~(1<<5 | 1<<7);
                                        break;
                                case 2:
                                        LEDType = 2;
                                        PORTC |= (1 << 7);
                                        PORTC &= ~(1<<6 | 1<<5);
                                        break;
                        }
                }
        }
}
```

Figure 41.Main Task Source File for the MicroController (AVR)

myAVR.h - My AVR Task Header File

```
#define BAUD0                   115200
#define USART0_RX               0x0028
#define F_CPU                   18432000

#define ENTER_KEY               14
#define STAR_KEY                12

void timer0_Init(void);
void timer1_Init(void);
void timer2_Init(void);

void USART0_Transmit(unsigned char);
void ser0Str(char*);

unsigned int numericHit(int);

void int0_Init(void);

void myDelay(int);
```

Figure 42. MyAVR Task Header File

myAVR.c – myAVR Task Source Code

```
#include "myAVR.h"
#include "main.h"
#include <avr/interrupt.h>
#include <util/delay.h>
#include <avr/io.h>
```

(myAVR..c continue 1/5)

```
#include <avr/iom324.h>
#include <avr/wdt.h>

extern unsigned int myLEDScaler;
extern unsigned int LEDType;
//TIMER CODE------------------------------------------------------------------------------------------------

//___TIMER0 (8-Bit w/PWM Capability)_______________________________________________

void timer0_Init() //"LED CLK"
{
   //TCCR0A |= (1 << WGM01);                           //Set To Normal Mode

   //TCCR0B |=  (1 << FOC0A);                          // Configure timer 0 for CTC mode

   TCCR0B &= ~(1 << CS01 | 1<<CS00);
   TCCR0B |=  (1 << CS02);                             // Set The Prescaling to 1024

   TIMSK0 |= (1 << TOIE0);                             //Enable The 'Timer Overflow' Interrupt
}
ISR(TIMER0_OVF_vect)
{
        static int myCount = 0;

        if(myCount == myLEDScaler)
        {
                switch(LEDType)
                {
                        case 0:
                                PORTC ^=  (1 << 5);
                                PORTC &= ~(1<<6 | 1<<7);
                                break;
                        case 1:
                                PORTC ^= (1 << 6);
                                PORTC &= ~(1<<5 | 1<<7);
                                break;
                        case 2:
                                PORTC ^= (1 << 7);
                                PORTC &= ~(1<<6 | 1<<5);
                                break;
                }
        }
        myCount = (myCount+1)%(myLEDScaler+1);
}

//___TIMER1 (16-Bit w/PWM Capability_________________________________________________
void timer1_Init()     //"DIG CLK"
{
   TCCR1B |= (1 << WGM12);                             // Configure timer 1 for CTC mode (normal w/out PWM)

   TCCR1B &= ~(1 << CS11 | 1 << CS10);
   TCCR1B |=  (1 << CS12 );                            // Set The Prescaling to 256

   OCR1A   = TIMER1_INIT_COUNT;                        // Set 'Timer 1 Compare A' To GLACIAL_FREQ_MAX (i.e.
1khz)
   TIMSK1 |= (1 << OCIE1A);                            // Enable CTC interrupt
}

ISR(TIMER1_COMPA_vect)
{
        DIG_FLOW_PORT ^= (1 << DIG_FLOW_PIN); // Toggle the DIG FLOW Clk
}

//___TIMER2 (8-Bit)____________________________________________________________________
void timer2_Init()     //SPKR CLK"
{
   //TCCR2B |= (1<<FOC2A);
   //TCCR2B |= (1 << WGM22);                    // Configure timer 1 for CTC mode (normal w/out PWM)

   TCCR2A |= (1<<WGM21);
   TCCR2B &= ~(1 << CS21 | 1 << CS20);
   TCCR2B |=  (1 << CS22);                      // Set The Prescaling to 256

   OCR2A   = 250;                       //Set CTC compare value to 1Hz at 1MHz AVR clock, with a prescaler of
```

(myAVR.c continue 2/5)

```
64
   TIMSK2 |=(1 << OCIE2A);                   // Enable CTC interrupt
}

void timer2_Initb(int num)    //SPKR CLK"
{
        //TCCR2B |= (1<<FOC2A);
   //TCCR2B |= (1 << WGM22);                  // Configure timer 1 for CTC mode (normal w/out PWM)

   TCCR2A |= (1<<WGM21);
   TCCR2B &= ~(1 << CS21 | 1 << CS20);
   TCCR2B |=  (1 << CS22);                   // Set The Prescaling to 256

   OCR2A   = num;                     //Set CTC compare value to 1Hz at 1MHz AVR clock, w/ a prescaler of 64
   TIMSK2 |=(1 << OCIE2A);                   // Enable CTC interrupt
}

ISR(TIMER2_COMPA_vect)
{
        SPKR_CLK_PORT ^= (1 << SPKR_CLK_PIN); // Toggle SPKR Clk
}

/*int setPrescaler(int Timer,int prescaler)
{
        if(!(prescaler == 1024 | prescaler == 256 | prescaler == 64 | prescaler == 8 | prescaler == 1)
}FOR LATER...*/

//EXTERNAL INTERRUPTS---------------------------------------------------------------------------------------

//___INT0
void int0_Init()
{
    EIMSK  &= ~(1<<INT0);                    //Mask the Bit

        EICRA = (1<<ISC01) | (1<<ISC00);     //Trigger On Rising Edge

        EIMSK  |= (1<<INT0);                 //Unmask the Bit

        PCICR = 0;
}

//This is the interrupt to service the Glacial Ping Functionality.
//      Current:       Simulates a 30ms Return Ping.
//      Future:        Interrupt Driven Switch
//                            Variable Depth (i.e. 'melting glacier')
ISR(INT0_vect)
{
        unsigned int i,currGlacialDepth = currGlacialPingPeriod;
                //Mask Interrupt
        EIMSK = 0;
        //Delay Loop. Compiler is Optimizing the _delay_ms and can't
        //      establish why; will use this clunky structure instead for now.
        //      If there is time, this functionality will be performed correctly. But Not For The Moment...
        for(i =0;i<currGlacialDepth;i++)
                _delay_ms(1);

        setPin(PING_PORT,PING_OUT_PIN,ON);           //The Busy Pin Is Just A Flag
        _delay_ms(16);
        setPin(PING_PORT,PING_OUT_PIN,OFF);          //The Busy Pin Is Just A Flag

        //Reset Flag
        EIFR |= 1<<INT0;

        //UnMask Interrupt
        EIMSK |= 1<<INT0;
}       //Leave!!!


//USART CODE--------------------------------------------------------------------------
void USART0_Init() {

        UBRR0H = (unsigned char)((F_CPU/(BAUD0*16UL)-1)>>8); /* Set baud rate */
        UBRR0L = (unsigned char)(F_CPU/(BAUD0*16UL)-1);
```

(myAVR.c continue 3/5)

```
        UCSR0B = (1<<RXEN0)|(1<<TXEN0);             /* Enable receiver and transmitter */
        UCSR0C = (0<<USBS0)|(3<<UCSZ00);            /* Set frame format: 8data, 2stop bit 1 stop bit*/
        UCSR0B |= (1<<RXCIE0);                      // enable usart to recieve interrupts
        sei();                                      //Signal EOI
}

void USART0_Transmit(unsigned char data)
{
        while ( !( UCSR0A & (1<<UDRE0)) );
        UDR0 = data;
}
void ser0Str(char* str)
{
        while(*str)
        {
                USART0_Transmit(*str++);
                _delay_ms(500);
        }
}

void transmitBin0(unsigned char myChar)
{
        int i =7,currBit;

        for(i=7;i>=0;i--)
        {
                currBit = (myChar&(1<<i)) >>i;
                USART0_Transmit(currBit + '0');
        }
}

//INTERRUPT INTIALIZATION AND ISR---------------------------------------------------------------------------
#define INT0_VECT 0x02

//PORT/PIN API----------------------------------------------------------------------------------------------
void setPinB(int pin,int state)
{
        if(state)
                PORTB |= 1<<pin;
        else
                PORTB &= ~(1<<pin);

}

void setPin(unsigned char port, int pin, int state)
{
        switch(port)
        {
                case 'A':
                        if(state)
                                PORTA |= 1<<pin;
                        else
                                PORTA &= ~(1<<pin);
                        break;
                case 'B':
                        if(state)
                        PORTB |= 1<<pin;
                        else
                        PORTB &= ~(1<<pin);
                        break;
                case 'C':
                        if(state)
                        PORTC |= 1<<pin;
                        else
                        PORTC &= ~(1<<pin);
                        break;
                case 'D':
                        if(state)
                        PORTD |= 1<<pin;
                        else
                        PORTD &= ~(1<<pin);
                        break;
        }
}
```

(myAVR.c continue 4/5)

```
unsigned int readPin(unsigned char port, unsigned int pin)
{
        unsigned char myPort;
        unsigned int myPin;

        switch(port)
        {
                case 'A':
                        myPort = PINA;
                        break;
                case 'B':
                        myPort = PINB;
                        break;
                case 'C':
                        myPort = PINC;
                        break;
                default:
                        myPin = -1;
        }

        myPin = (myPort & (1<<pin))>>pin;
        return myPin;
}
void writeNibble(unsigned char port, unsigned int startBit, unsigned short data)
{
        unsigned int bit0,bit1,bit2,bit3;
        bit0 = (data&(1<<0))>>0;
        bit1 = (data&(1<<1))>>1;
        bit2 = (data&(1<<2))>>2;
        bit3 = (data&(1<<3))>>3;

        setPin(port,startBit+0,bit0);
        setPin(port,startBit+1,bit1);
        setPin(port,startBit+2,bit2);
        setPin(port,startBit+3,bit3);
}

void myDelay(int x)
{
        int i =0;
        for(i=0;i<x;i++)
        {
                _delay_ms(10);
                i =i ;
        }
}
```

Figure 43. myAVR Task Source Code

keypad.c – keyPad Task Source File

```
#include "myAVR.h"

#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <avr/io.h>
#include <avr/iom324.h>
#include <avr/wdt.h>

int pollKeyPad()
{
        int             myRow = -1, myCol = -1;

    unsigned int        readOne, readTwo, readThree, readFour;
        unsigned char   newPin;

        //Pull In KeyPad Value
        newPin = (PINA & 0xF0)>>4;

        switch(newPin)
        {
```

(keyPad.c continue)

```
                case 1:
                        myRow = 0;
                        break;
                case 2:
                        myRow = 1;
                        break;
                case 4:
                        myRow = 2;
                        break;
                case 8:
                        myRow = 3;
                        break;
                default:
                        myRow = -1;
        }

                readOne   = 0;
                readTwo   = 0;
                readThree = 0;
                readFour  = 0;

                PORTA     = 0x08;
                _delay_ms(10);
                readOne   = (PINA & 0xF0)>>4;

                PORTA     = 0x04;
                _delay_ms(10);
                readTwo   = (PINA & 0xF0)>>4;

                PORTA     = 0x02;
                _delay_ms(10);
                readThree = (PINA & 0xF0)>>4;

                PORTA     = 0x01;
                _delay_ms(10);
                readFour  = (PINA & 0xF0)>>4;


                PORTA     = 0x0F;
                if(readOne == newPin)
                        myCol = 3;
                else if (readTwo == newPin)
                        myCol = 2;
                else if (readThree == newPin)
                        myCol = 1;
                else if (readFour == newPin)
                        myCol = 0;
                else
                        myCol = -1;

                if(myRow>=0 && myCol >=0)
                        return (myRow)*4 + myCol;
                else
                        return -1;
}

unsigned int numericHit(int x) //Will Return a 1 if a key 0-9 was pressed, otherwise 0
{
        if(x == 0|| x == 1 || x == 2 || x == 4 || x == 5 || x == 6 || x == 8 || x == 9 || x == 10 || x == 13)
                return 1;
        else
                return 0;
}
```

Figure 44. keyPad Task Source File

CRITICAL SECTION CODE

pingResponse.asm – Ping Response Assemble Source Code

```
; AUTHOR:    Justin Reina
; CREATED:   5 May 2008
; PROJECT:   EE472 - Lab 4
; MODULE:    Interrupt For The User Request
```

```
.model small

PUBLIC _pingRespond

extrn _OSTaskResume:proc
extrn _OSIntEnter:proc
extrn _OSIntExit:proc
extrn _pingCountPtr:word
extrn _latchedCountPtr:word

;Defines The Code Section
.code

;This is The Name of the ISR
_pingRespond proc

push bp
push dx
push ax
;push si
;push di
mov  bp,sp

mov dx, 0FF3Eh
mov ax, 0000Fh
out dx,ax

call _OSIntEnter

mov si,_pingCountPtr
mov di,_latchedCountPtr

mov ax, [si]
mov dx, [si+2]

mov [di], ax
mov [di+2],dx

push 8h
call _OSTaskResume
pop ax

mov dx,0FF22h
mov ax,0000Fh         ;Bh is the EOI for INT6
out dx,ax

call _OSIntExit

; End The Interrupt (EOI)

;      Restore Registers
;pop di
;pop si
pop ax
pop dx
pop bp

iret

_pingRespond endp
end
```

Figure 45. Ping Response Assembler Source File

Timer0.asm – Timer 0 Assembly Source Code

```
; AUTHOR:    Justin Reina
; CREATED:   5 May 2008
; PROJECT:   EE472 - Lab 4
; MODULE:    Interrupt For The User Request
```

```
.model small

PUBLIC _Timer0_ISR


;     Defines The Code Section
.code

;  This is The Name of the ISR
_Timer0_ISR proc

push bp
push dx
push ax
mov  bp,sp

; End The Interrupt (EOI)
mov dx,0FF22h
mov ax,00008h           ;Bh is the EOI for INT6
out dx,ax

;     Restore Registers
pop ax
pop dx
pop bp

iret

_Timer0_ISR endp
end
```

Figure 46. Timer 0 Assembly Code

Timer2.asm – Timer 2 Assembly Source Code

```
; AUTHOR:    Justin Reina
; CREATED:   5 May 2008
; PROJECT:   EE472 - Lab 4
; MODULE:    Interrupt For The User Request

.model small

PUBLIC _timer2isr

extrn _pingCountPtr:word

;     Defines The Code Section
.code

_timer2isr proc

push bp
push dx
push ax
push si
push di
mov  bp,sp

mov si,_pingCountPtr
mov di,_pingCountPtr

mov ax, [si]
mov dx, [si+2]

add ax,1
adc dx,0

mov [di], ax
mov [di+2],dx

; End The Interrupt (EOI)

mov dx,0FF22h
mov ax,00008h           ;Bh is the EOI for INT6
out dx,ax

;     Restore Registers
pop di
pop si
pop ax
pop dx
pop bp

iret

_timer2isr endp
end
```

Figure 47. Timer 2 Assembyl Code

ACKNOWLEDGEMENTS

The following sources were used in the development of the system source code:

- µC/OS operating system, courtesy of Jean Labrosse
    - Educational license through the University of Washington Electrical Engineering Department
- TD40 library
    - Courtesy of the Tern Corporation
    - Educational license through the University of Washington Electrical Engineering Department
- FFT Analysis source code.
    - Courtesy of Brent Plumb, former UW EE student