**EE 472 Lab 5 Report**

Jitesh Bhogal, Thomas Wei, Xinyu Sui

EE 472

Microcomputer Systems

Summer Quarter 2015

August 21st, 2015

# 

Contents

[ABSTRACT 4](#_Toc427934227)

[1.0 INTRODUCTION 4](#_Toc427934228)

[1.1 DISCUSSION OF THE LAB 5](#_Toc427934229)

[1.1.1 System Kernel: 7](#_Toc427934230)

[1.1.1.1 Design Specification: 7](#_Toc427934231)

[1.1.1.2 Software Implementation: 7](#_Toc427934232)

[1.2 Communication Subsystem: 9](#_Toc427934233)

[1.2.1 Design Specification: 9](#_Toc427934234)

[1.2.2 Software Implementation: 10](#_Toc427934235)

[1.3 Display and Annunciation Subsystem: 11](#_Toc427934236)

[1.3.1 Design Specifications: 11](#_Toc427934237)

[1.3.2 Software Implementation: 11](#_Toc427934238)

[1.4 Warning Alarm System: 14](#_Toc427934239)

[1.4.1 Design Specification: 14](#_Toc427934240)

[1.4.2 Software Implementation: 14](#_Toc427934241)

[1.5 Power Management System: 16](#_Toc427934242)

[1.5.1 Design Specification 16](#_Toc427934243)

[1.5.2 Software Implementation 16](#_Toc427934244)

[1.6 Thruster System 19](#_Toc427934245)

[1.6.1 Design Specification 19](#_Toc427934246)

[1.6.2 Software Implementation 20](#_Toc427934247)

[1.7 Vehicle Communication System 22](#_Toc427934248)

[1.7.1 Design Specification 22](#_Toc427934249)

[1.72 Software Implementation 22](#_Toc427934250)

[1.8 Solar Panel Control System 23](#_Toc427934251)

[1.8.1 Design Implementation 23](#_Toc427934252)

[1.8.2 Software Implementation 23](#_Toc427934253)

[1.9 Keypad Control 24](#_Toc427934254)

[1.9.1 Design Implementation 24](#_Toc427934255)

[1.9.2 Software Implementation 24](#_Toc427934256)

[1.10 Pirate Detection and Management 24](#_Toc427934257)

[1.10.1 Design Implementation 24](#_Toc427934258)

[1.10.2 Software Implementation 24](#_Toc427934259)

[1.11 Transport Distance 25](#_Toc427934260)

[1.11.1 Design Implementation 25](#_Toc427934261)

[1.11.2 Software Implementation 25](#_Toc427934262)

[1.12 Command Manage 25](#_Toc427934263)

[1.12.1 Design Implementation 25](#_Toc427934264)

[1.12.2 Software Implementation 25](#_Toc427934265)

[1.13 Image Capture 26](#_Toc427934266)

[1.13.1 Design Implementation 26](#_Toc427934267)

[1.13.2 Software Implementation 26](#_Toc427934268)

[2.0 PRESENTATION, DISCUSSION, AND ANALYSIS OF THE RESULTS 26](#_Toc427934269)

[3.0 ANALYSIS OF ANY ERRORS 27](#_Toc427934270)

[4.0 ANALYSIS OF WHY THE PROJECT MAY NOT HAVE WORKED AND WHAT EFFORTS WERE MADE TO IDENTIFY THE ROOT CAUSE OF ANY PROBLEMS 27](#_Toc427934271)

[5.0 TEST PLAN 27](#_Toc427934272)

[6.0 TEST SPECIFICATION 28](#_Toc427934273)

[6.1 PWM 28](#_Toc427934274)

[6.2 GPIO 28](#_Toc427934275)

[6.3 Task Timing 28](#_Toc427934276)

[6.4 Ethernet 28](#_Toc427934277)

[6.5 Distance 28](#_Toc427934278)

[6.6 Pirate 28](#_Toc427934279)

[7.0 TEST CASES 29](#_Toc427934280)

[8.0 SUMMARY 30](#_Toc427934281)

[9.0 CONCLUSION 30](#_Toc427934282)

[10.0 WORK DIVISION 31](#_Toc427934283)

[11.0 APPENDIX 32](#_Toc427934284)

[11.1 Source Code 32](#_Toc427934285)

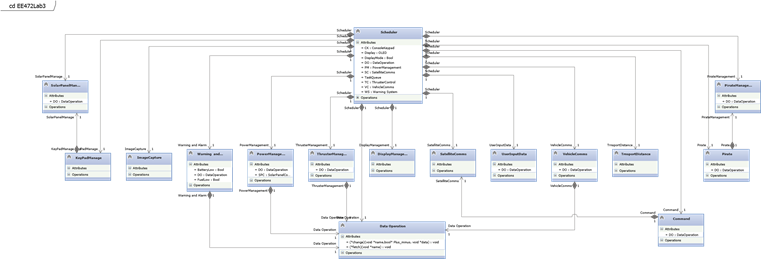
# 

# ABSTRACT

In this lab the final phase of an exploratory mining satellite is developed. This phase focuses on replacing the previously designed system to a Real Time Operating System that utilizes a preemptive, priority based scheduler to do the same job. Additionally we have added several capabilities to the initial design such as drivers to support Vehicle Distance tracking, Image Capture, Pirate Detection, and an Internet Based Communication System. The upgraded subsystem is now capable of managing the satellite, collecting data from several different types of sensors, processing the data from those sensors, displaying it locally, using some of the data to control the tools for performing the necessary mining operations, as well as defending itself against pirates or incoming dangerous obstacles. The following report outlines the initial deliverables which include full specification and design documentation for the portions of the system under current development, the high-level system architecture, the ability to perform a subset of the necessary control functions, and portions of the display and the annunciation components

# 1.0 INTRODUCTION

In this lab we improve upon the basic system of a satellite mining system, built with C language on Texas Instruments Stellaris EKILM3S8962. A Satellite Management and Control System that was modelled will now become the basis of development. The overall function of the subsystem is improved by the by switching to a RTOS system and hard and soft timing constraints. Previous additions of a startup task, implementing a hardware system time base, incorporating safety features and using internal and external interrupts with GPIO to measure battery level, drive solar panel and a bidirectional serial communication channel between the satellite and the land based mining vehicle have been supplemented with a bidirectional remote communication system, extended mining vehicle and transport communications link and introduces several tasks to drive peripheral devices. For this prototype, the console will be modeled with the OLED display on the Stellaris board. Commands will be received from and pertinent mission information will be sent to an earth station via internet communications protocol. Figure 1 shows the Class diagrams outlining a high level view of the subsystems to be discussed in the remainder of the report.

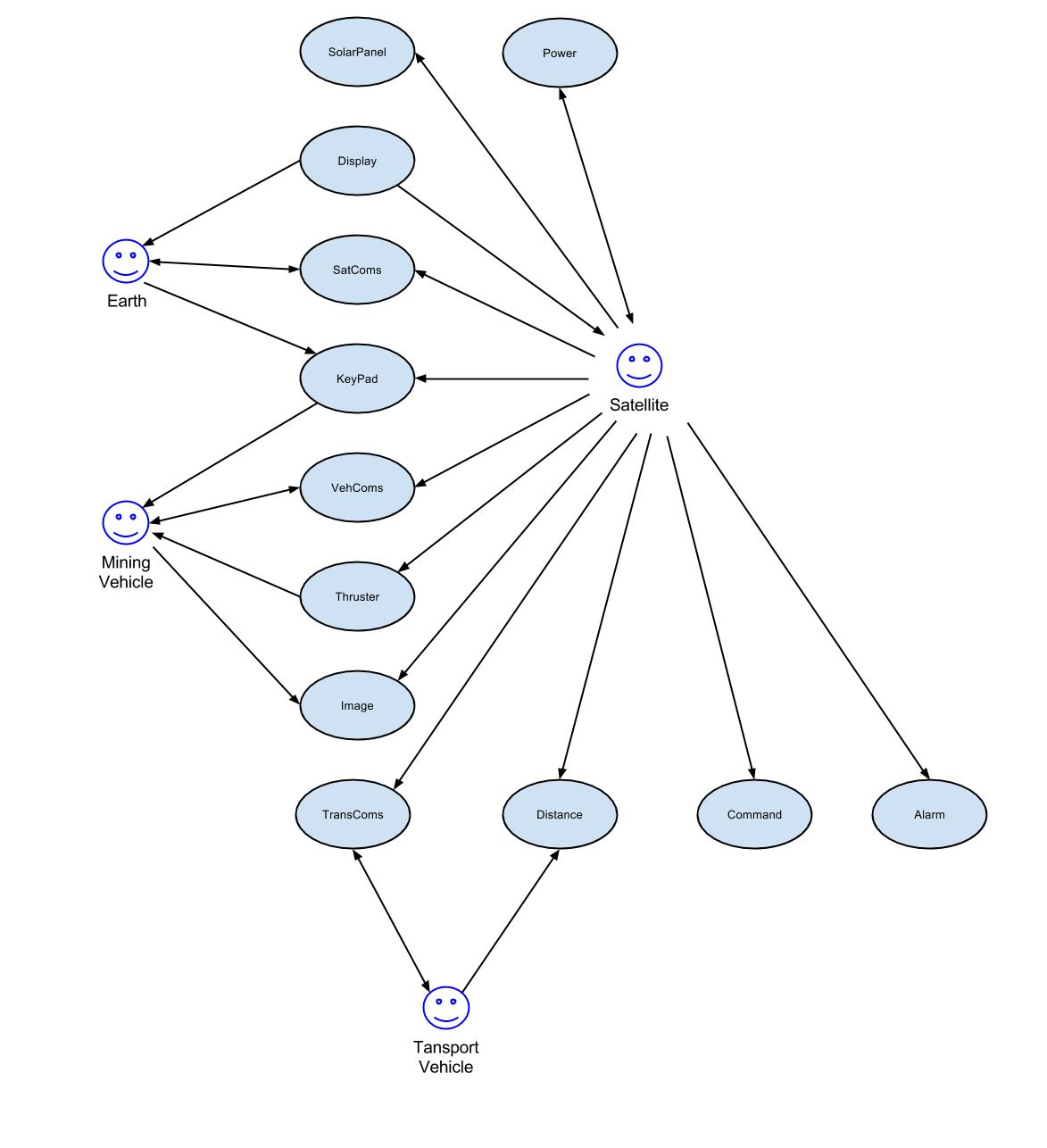


**Figure 1: UML class diagram**

The Satellite Management and Control System is designed and developed. The overall function of the system is to be able to control surface based mining equipment from an orbiting command center and to be able to communicate with various earth stations. Status Warning and alarm information will be displayed on a system console modelled with the OLED display on the Stellaris board. Beyond the capabilities of the first phase, the second phase implements many of the subsystem that were modeled during the initial phase. Additional capabilities are also incorporated and are the main focus of this report.

# 1.1 DISCUSSION OF THE LAB

Figure 2 shows the Use Case Diagram related to the major sub systems of the project. The use cases are further outlined within their respective sequence diagrams.



**Figure 2: UML use case diagram**

## 

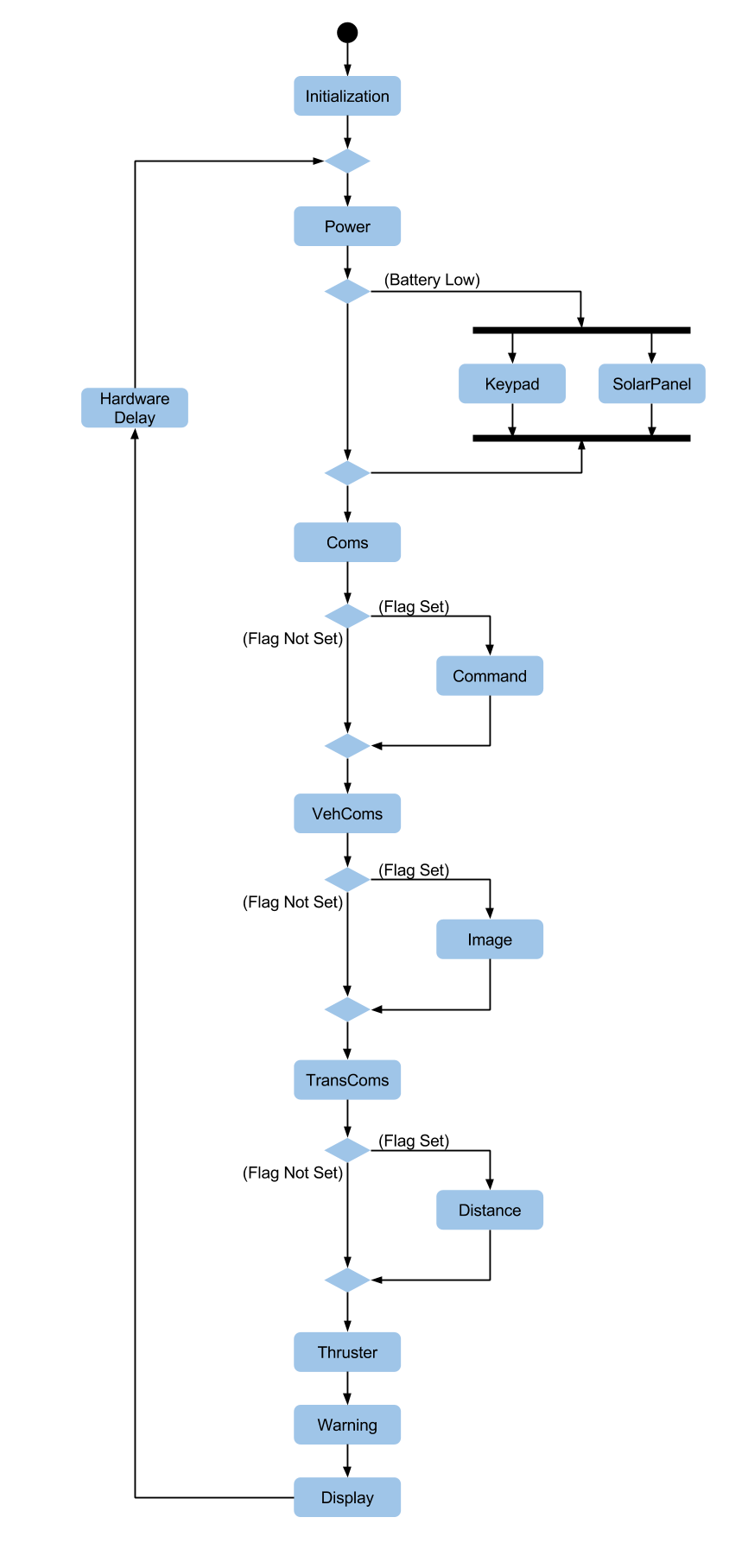
## 1.1.1 System Kernel:

### 1.1.1.1 Design Specification:

FreeRTOS is used as the system kernel in this lab, i.e., FreeRTOS dynamically schedules and deletes the tasks. TCB is still used to construct the task queue. The nine static TCB elements in the queue are: Manage Power Subsystem, Manage Solar Panel, User Input Data, Manage Thrusters, Satellite Communications, Landbased Mining Vehicle Communications, Transport Vehicle Communications Status and Annunciation Display, Warning and Alarm. Two optional tasks are static TCB elements Pirate Detection and Pirate Discouragement. Four tasks, the Command, Image Capture, solarPanelControl, and transportDistance and one subtask, batteryTemperature are to be added or deleted as needed. In each minor cycle, the system will go through the task queue once. The period of the major cycle is 5 seconds. Within the major cycle, a number of minor cycles should be executed.

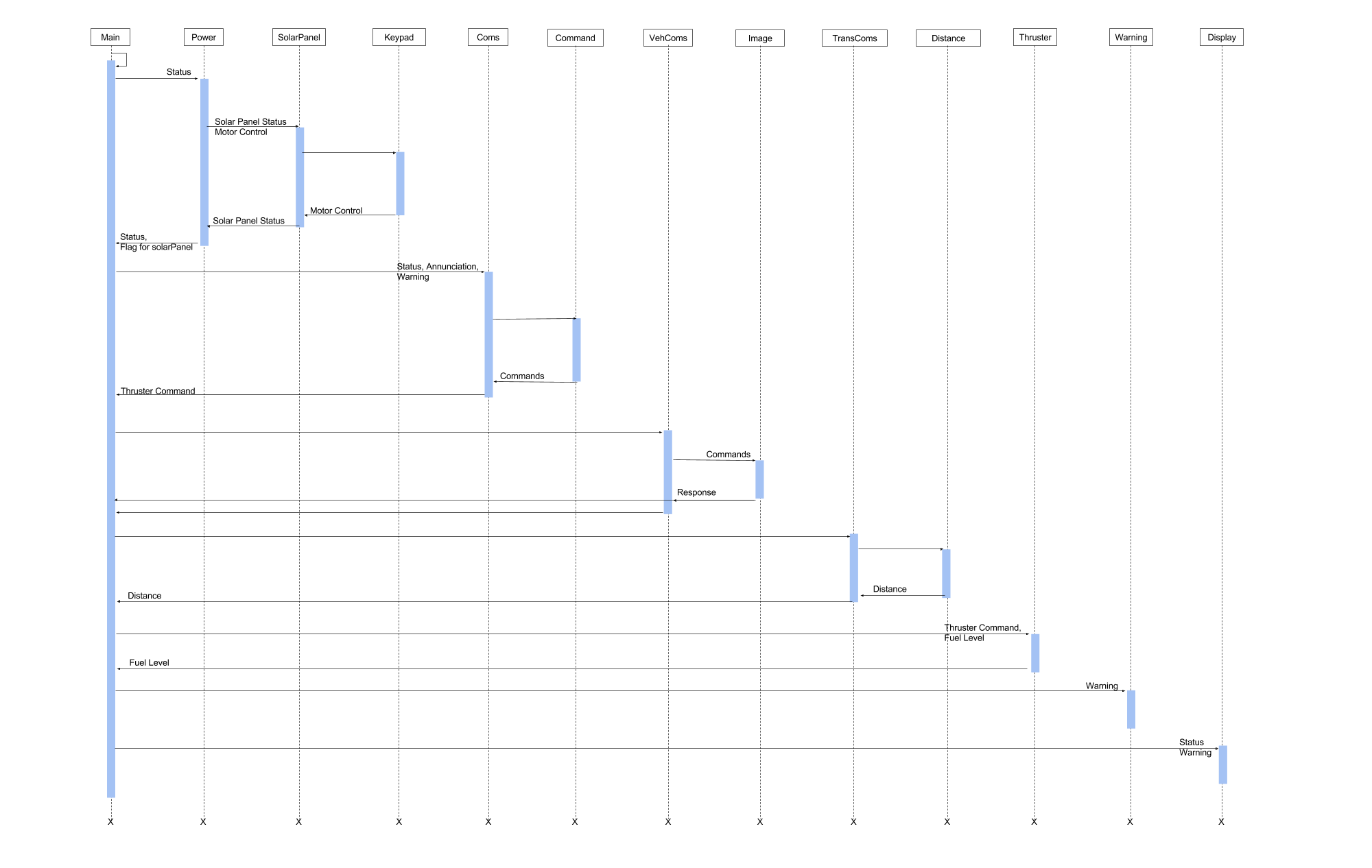
### 1.1.1.2 Software Implementation:

A data structure is built to construct a general purpose task queue in FreeRTOS. The system goes through the task queue infinitely to do its major job, mining, and keep track of the time the program has run. Every task is a subsystem. For each of the subsystems, a data structure, containing pointers to some of the global variables, is used for communication with the system. Figure 3 shows a general overview of the entire system and flow of control.



**Figure 3: System task flow**

A TCB data structure consists of a pointer to a function and a general pointer. There are several global variables defined and initialized before as task starts, as shown in the specification. A data structure containing pointers to some of the global variables are built for each of the subsystems as media for communication between the systems. Data of every implementation of subsystems are stored in its data structure, address of which is stored in the general pointer in the TCB data structure consisting task queue. Pointers in data structures are also initialized before going through the task queue. The system iterates infinitely and goes through task queue once each iteration, i.e., a minor cycle. The order of tasks in the task queue is determined by the delay time specified in each task, and the priority passed to the task creator. All tasks which needed to be dynamically scheduled and suspend are brought in or out of the task queue by different flags.

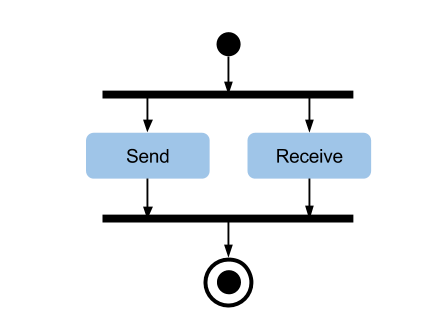


**Figure 4: Main sequence diagram**

## 1.2 Communication Subsystem:

### 1.2.1 Design Specification:

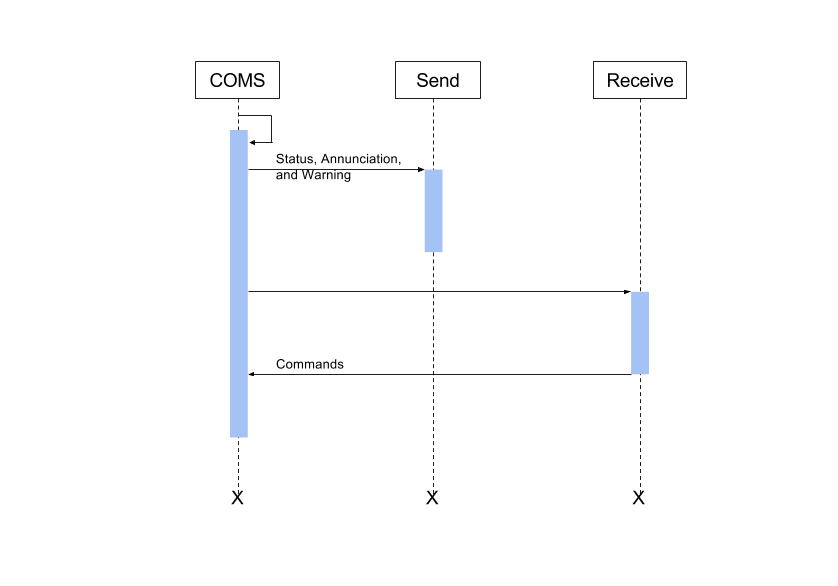
The satelliteComs handles communications with the earth. The function accepts a pointer to void with a return of void. A web server should be set up to communicate with a remote browser. Data transfer from the satellite to earth consists of status and alarm information, with other specified input such as the date. Some specific command can be received and handled properly. Data communication between the earth and the satellite follows the following scheme:



**Figure 5: Communications system activity diagram**

### 1.2.2 Software Implementation:

Firstly, the html files provided in the sample code provided by FreeRTOS is modified to produce the proper behavior. Specifically, there are two files added: display.shtml and command.shtml. Makefsfile is used to produce the fsdata file. There are two major modifications applied to the sample code. The first modification is modifying the handler links the code and the html file, so that the proper display information can be sent by the Ethernet. The second modification to the code is in the ISR for input data by the web browser. When the data is received, the ISR stores the data to global variables and resumes command task. After the command has done its work, a flag is set so that proper response can be sent out by the Ethernet.



**Figure 6: Display system activity diagram**

## 1.3 Display and Annunciation Subsystem:

### 1.3.1 Design Specifications:

Manages the display of the satellite status and alarm information. Function accepts a pointer to void with a return of void. In the implementation of the function the pointer is re-cast as a pointer to a display data structure type before it is referenced. The function also uses an external push button with an asynchronous interrupt to switch between the two displays: Satellite Status and Annunciation.

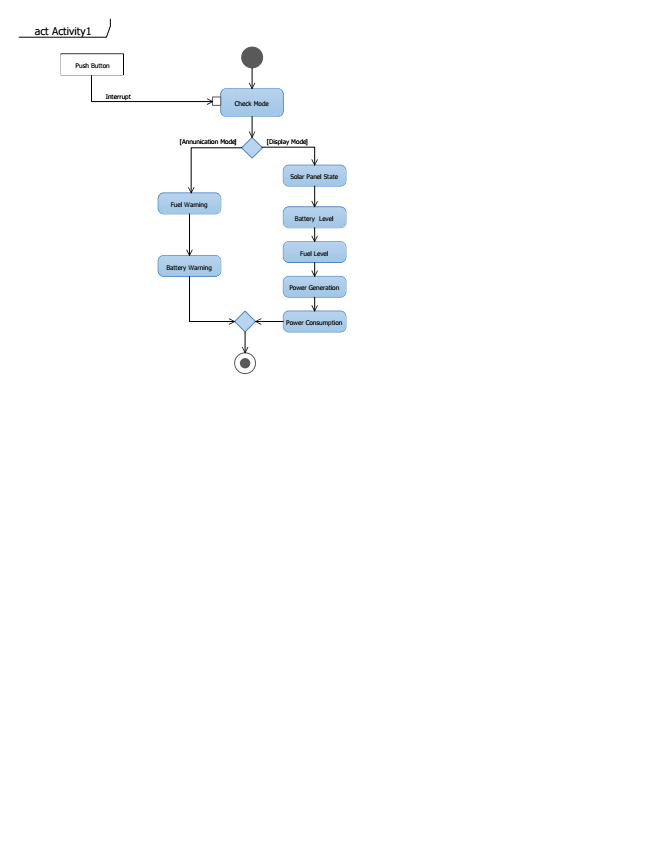
Satellite Status mode displays: Solar Panel State, Battery Level, Fuel Level, Power Consumption and Power Generation, Battery Temp Warning and Vehicle Disatnce

Annunciation mode displays: Fuel Low warning, Fuel Level, Battery Low Warning, Battery Level, Battery Temp Warning

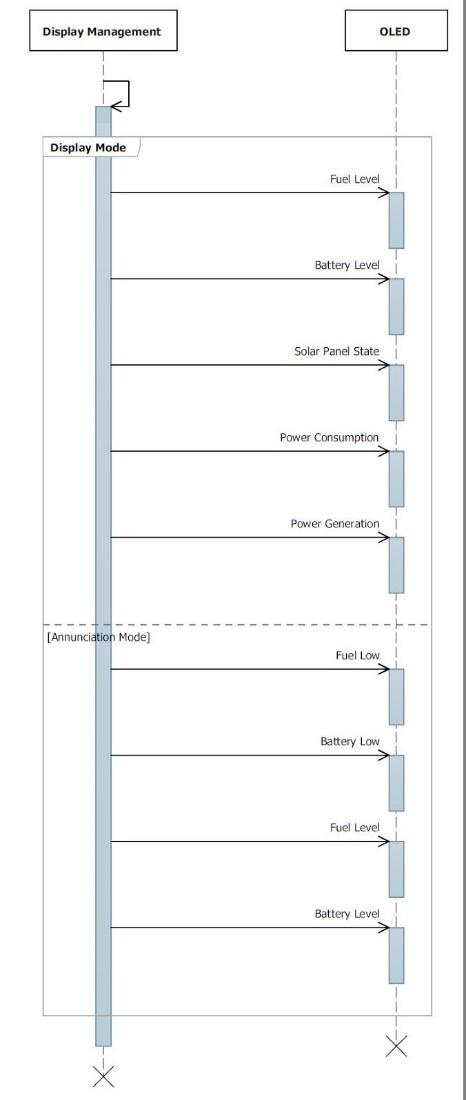
The Display Subsystem run on the minor cycle and consequently will provide the most up to date information to the display panel. The addition of Power generation to the Satellite Status mode is to convey functionality of the solar panels in case of potential damage or malfunction. This feature will give future capability to manage remaining power to critical subsystems or initiate alternate methods of Power generation. Veh Distance shows how far an incoming vehicle is whereas Battery Temp shows the current temp of the Battery.

### 1.3.2 Software Implementation:

The Figure 7 shows the activity diagram of the subsystem. The Push Button triggers an interrupt which sets a flag to be acknowledged by the routine. The Push Button cannot trigger another interrupt until the flag has been reset by the system. The system implements several nested IF ELSE statements which call the OLED print function to display appropriate information onto the display. Additionally earth station can implement a Display or Enable command to the system which will turn the OLED on or off.



**Figure 7: Display system activity diagram**



**Figure 8: Display system sequence diagram**

FIGURE 8 shows the sequence diagram of the subsystem.

This is parallel to the activity diagram and provides an alternate look at the subsystem. during the Display mode the subsystem sequentially write information to the OLED. The various variables are converted into an acceptable form with the use of the sprintf() function call. The largest variable passed into this function is an int, of size 16, and the char array is sized with an excess of 4 bits to provide a buffer in case of faulty or accidental input into the function with a size greater than that of 16. The sequence diagram shows the signals (strings) sent to the OLED from within the Display Management subroutine. The final code can be found in the Appendix.

## 

## 1.4 Warning Alarm System:

### 1.4.1 Design Specification:

The warningAlarm function accepts a pointer to void with a return of void. In the implementation of the function, this pointer is re-cast as a pointer to the warningAlarm task’s data structure type. The warningAlarm task interrogates the state of the battery and fuel level to determine if they have reached a critical level.

* If both are within range, the green LED on the annunciation panel shall be illuminated and on solid.
* If the state of the battery level reaches 50%, the yellow LED on the annunciation panel shall flash at a 1 second rate.
* If the state of the fuel level reaches 50%, the yellow LED on the annunciation panel shall flash at a 2 second rate.
* If the state of the battery level reaches 10%, the red LED on the annunciation panel shall flash at a 1 second rate.
* If the state of the fuel level reaches 10%, the red LED on the annunciation panel shall flash at a 2 second rate.
* If the over temperature event occurs, an audible alarm shall be set and shall remain active until acknowledged.
* If the alarm is unacknowledged for more than 15 seconds, the red and yellow LEDs shall flash with the pattern: flash for 5 seconds at a 10Hz rate – remain solid on for 5 seconds and the cycle will repeat

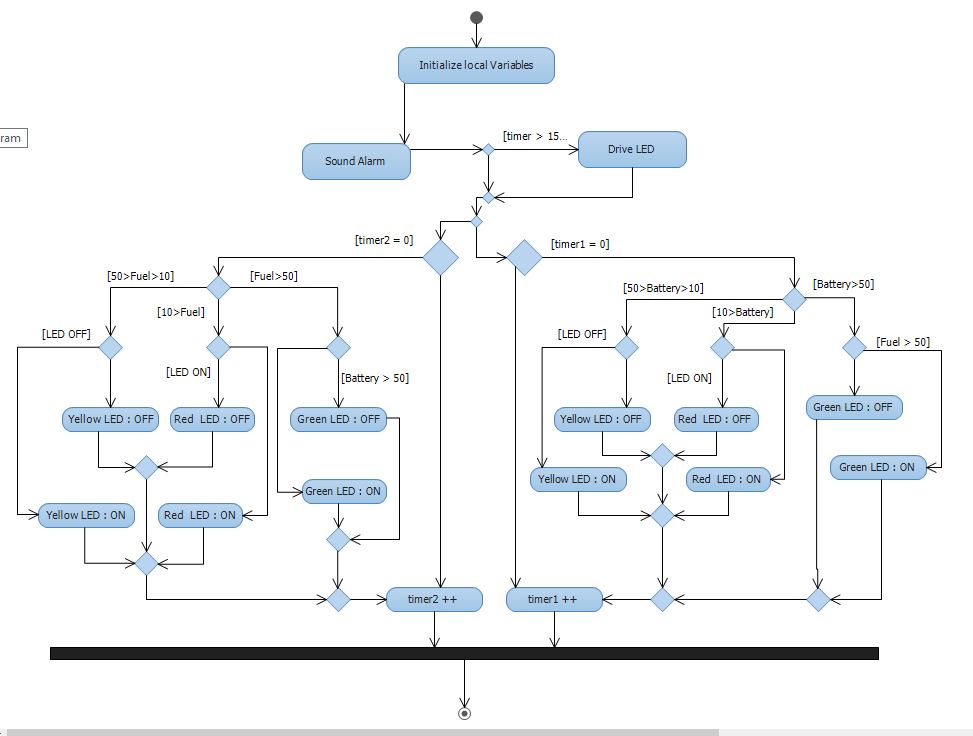
### 1.4.2 Software Implementation:

Figure 9 shows the activity diagram for the Warning subsystem. Please NOTE the flow of control is NOT PARALLEL. The activity diagram shows the functionality of the subsystem only. There are nine possible states as described by the design specification. The implementation internally tracks the action taken and duration of each individual LED. These are initialized when the function is first called and are static variables. This provides the Warning subsystem the ability to maintain a local clock which is driven by the external Global Clock every minor cycle. The subsystem evaluates Fuel and Battery level using AND “&&” conditions within IF Loops and break to exit.

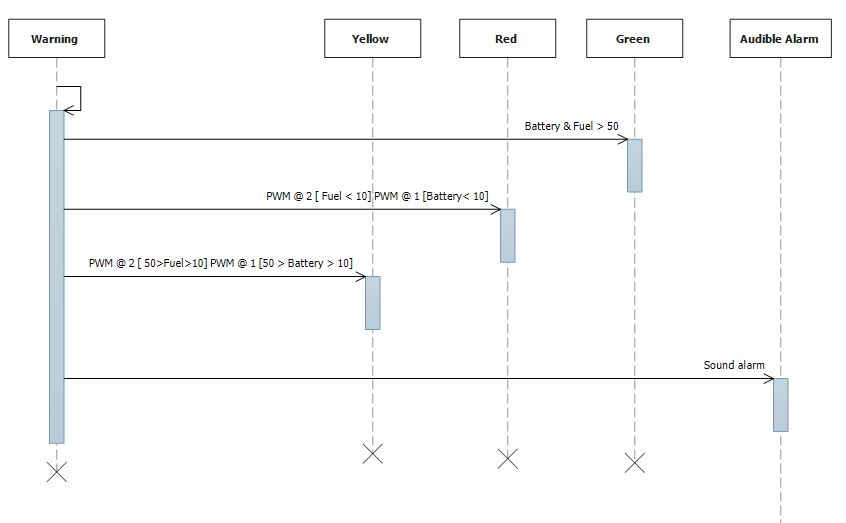
If Battery Temp alarm is active the taks deals with it first and starts a local timer. If the timer is not acknowledged for an extended period of time the Battery Temp Warning takes priority over the rest of this section.

The subsystem sends a HIGH or LOW signal to one of three GPIO pins which connect to appropriate LED on a peripheral board. Figure 10 is condensed to show only the relationship between the subsystem and the peripheral LED’s. The Green LED does not blink; the Yellow and Red LED’s are modulated with designed constraints using a PWM signal of either one and/or two seconds.

Additionally the LED ON/OFF variables within each loop are individual variables tied to the Fuel or Battery warning conditions. These DO NOT check the state of the LED but are variables to keep track of previous action taken. In example LED OFF translates to the function having switched off the LED during the previous walkthrough in relation to either Fuel or Battery LOW. Both Fuel and Battery Low INDIVIDUALLY track their respective actions. The activity diagram is not show as a sequential diagram for simplicity and readability. The final code can be found in the Appendix.



**Figure 9: Warning system activity diagram**



**Figure 10: Warning system sequence diagram**

## 1.5 Power Management System:

### 1.5.1 Design Specification

The powerManagement task takes in a void pointer to void and returns void. The void pointer is cast as a pointer, myPower, to the power data struct. In addition to the previous implementation power management now reads an analog signal on A/D channel 0 to measure the battery level. Because the signal takes time to settle following the connection of the solar panel output and the measurement equipment to the battery the measurement is made 600 usec following an external event signalling the connection to ensure a valid reading. powerManage now also measures and records the battery temperature of the satellite by probing the battery at 2 different locations (A/D channel 1 and 2) and storing the temperature values in a buffer. If either of the two temperatures exceeds the maximum of the two in a single update, a warning is triggered to alert the satellite of possible overheating.

### 1.5.2 Software Implementation

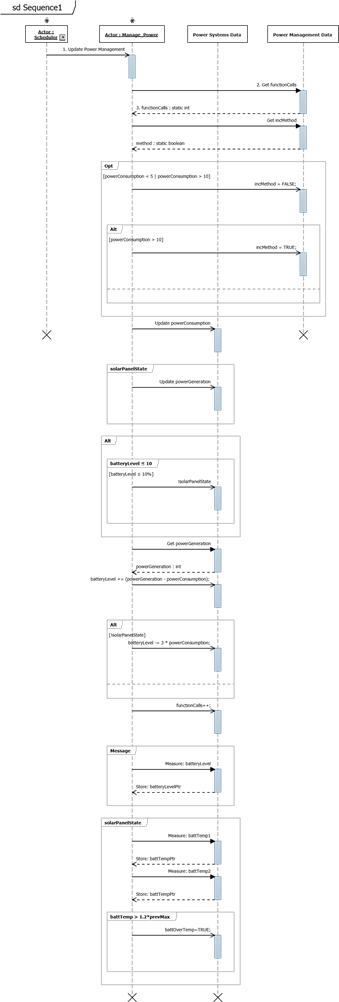
The powerManagement task is set to update once every major cycle (5 seconds). The powerManagement task manages the global variables: powerConsumption, powerGeneration, solarPanelState solarPanelDeploy, solarPanelRetract and batteryLevel. The task begins by initializing two static variables: funcCount and incMethod. FuncCount keeps track of the number of function calls and incMethod keeps track of the current method of incrementation based on previous calls. These two local variables determine the change in power consumption through a series of if-else statements.

The global variable, powerGeneration, also depends on the local variable funcCount to determine the incrementation value. However, the change in powerGeneration also depends on global variables, solarPanelState and batteryLevel. A series of if-else statements is implemented to either deploy a retracted solar panel, retract a deployed solar panel, or increment powerGeneration. PowerManagement now triggers an interrupt to signal the solar panel deployment and waits till full deployment or retraction before charging the battery.

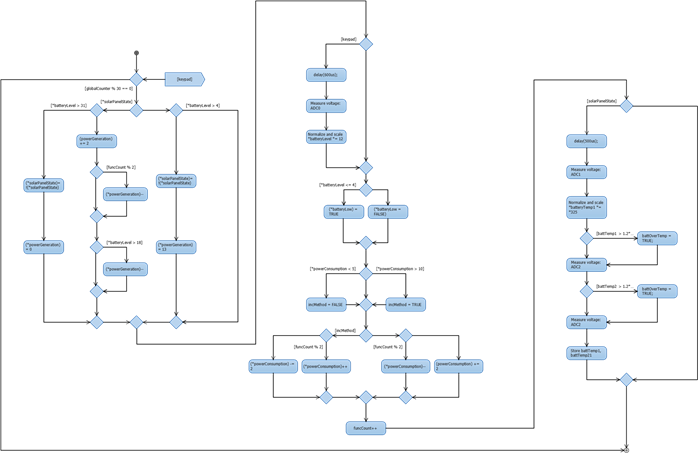
A deployed solar panel and high batteryLevel ( > 31) results in retraction of the solar panels and a reset of powerGeneration. If the batteryLevel is below 31 while the solar panel is deployed the powerGeneration is incremented according to the funcCount and batteryLevel. If the solar panel is not deployed and batteryLevel falls below or equal to 10, the solar panels will deploy and powerGeneration is set to 13. Setting powerGeneration to 13 guarantees that powerGeneration always exceeds powerConsumption which reaches a maximum of 12.

The batteryLevel is read as the output voltage from an ADC. The value output is normalized by its reference voltage (3V) and scaled to a 32V scale. In the current implementation, the battery level is controlled by an adjustable source voltage (the laboratory power supply). The state, deployment, and retraction of solar panels remain dependent on the current state of the battery level. Battery level measurement is not initialized until external input is provided to the system followed by a 600us delay to ensure a stable connection.

The batteryTemp subtask is to be triggered only while the battery is charging. The battery is charging if and only if the solar panel is currently deployed. The state of the solar panel is used as a condition in an if statement to begin temperature measurement. A software delay of 500us is used to enforce periodic measurements. The system’s battery temperature is read as the output from A/D channel 1 and 2. The input range of the A/Ds remains the same. In this case, the outputs are scaled to an output range of 325mV. The normalized input range (0 - 3.25V) is taken and converted to a temperature in celsius. The temperature is then converted to Kelvin in order to obtain higher accuracy warning temperatures. Both outputs are read, outputted and stored into a shared circular buffer.



**Figure 11: Power system sequence diagram**

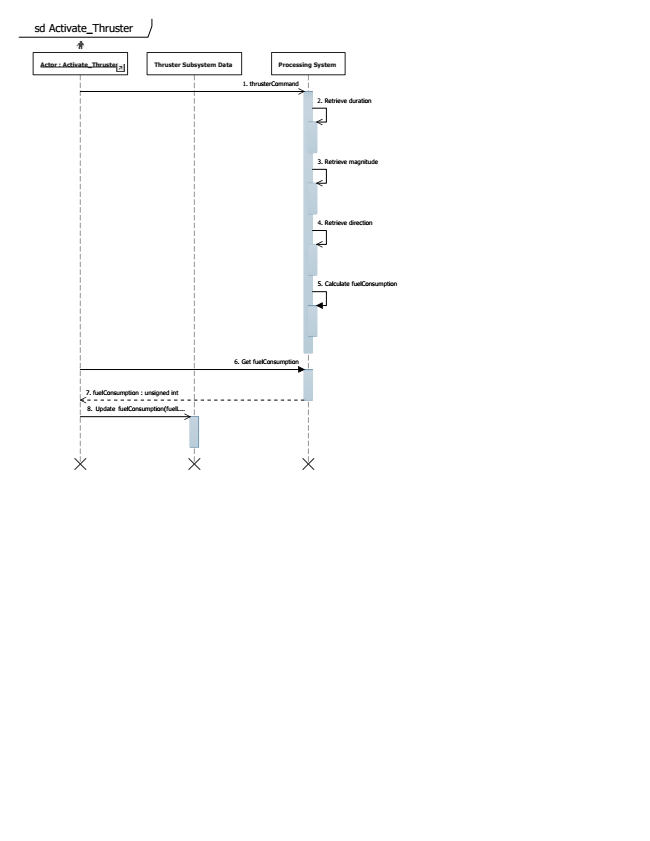


**Figure 12: Power system activity diagram**

## 1.6 Thruster System

### 1.6.1 Design Specification

The thursterManage task takes in a void pointer to void and returns void. The void pointer is cast as a pointer, myThruster, to the thruster data struct. The thrusterManage task deciphers thruster commands sent from Earth and powers the thrusters accordingly. While the thrusters are active, the thrusters consume fuel at the rate set by the specifications (6 month lifetime at a constant 5% magnitude). For demo purposes the rate of fuel consumption was increased by a factor of 864 which equates to 1/10th of a day every second. As the thrusters run, a PWM signal is output from the satellite indicative of the duration and magnitude.

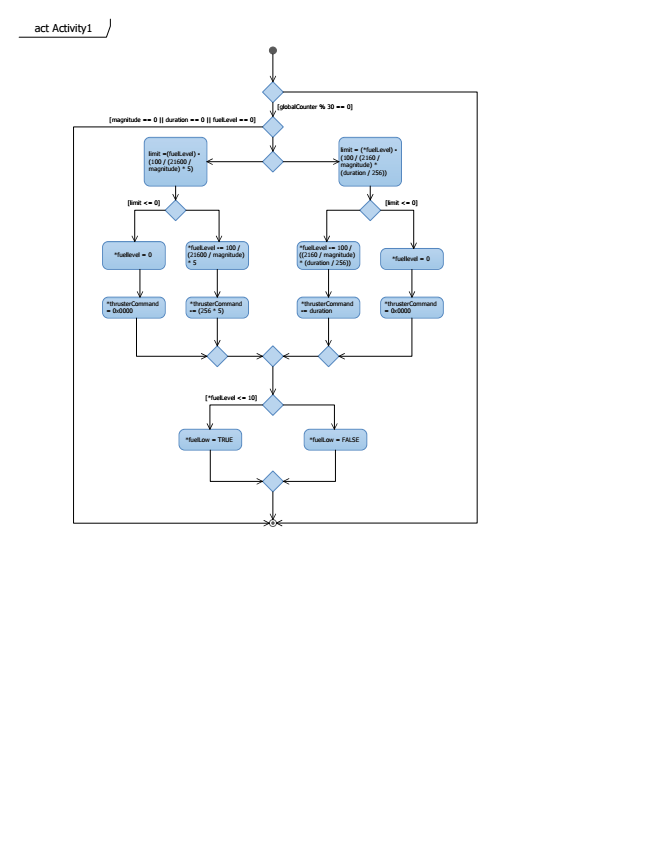


**Figure 13: Thruster system sequence diagram**

### 1.6.2 Software Implementation

The thrusterMangement task is set to update once every major cycle (5 seconds). The global variables, thrusterCommand and fuelLevel, are read and modified by the task. From thrusterCommand, the magnitude, duration, and direction of travel are determined. Each of the aforementioned properties is extracted using the & operand with initialized bits of ones at the desired bit ranges. For example, duration is extracted by (\*thrusterCommand) & 0xff00.

After extracting the desired properties, the fuelLevel is decremented according to magnitude and remaining duration. If either magnitude, duration or fuelLevel are 0, the fuel level goes unchanged. Otherwise, the fuelLevel function is split into 2 major cases depending on the remaining duration. As long as the remaining duration is longer than 5 seconds (int : 1280) the fuelLevel will decrement according to the magnitude which determines the rate of change over time. This decrement will take into account the change over a major cycle and decrement both the duration and fuelLevel accordingly.



**Figure 14: Thruster system activity diagram**

The duration is decremented by altering the first 8 bits of the thrusterCommand by 5 seconds (int : 1280). If the remaining duration does not warrant 5 more seconds of fuel use, the fuelLevel will drain according to the current magnitude and remaining duration. Since there is no longer ample time for another cycle, the duration will decrement to 0. Within these 2 major cases, local limits are calculated prior to the decrement to prevent the fuelLevel from dropping below 0. Therefore, if the fuelLevel would be less than 0 after the following decrement, the fuelLevel is set to 0.

The PWM output signal has a set period of 500ms. The output signal runs for the duration of the thruster command and ceases once the duration has elapsed. The duty cycle of the PWM signal indicates the magnitude of the thruster command. The duty cycle is calculated by scaling the normalized magnitude of the thruster command to the period length.

## 

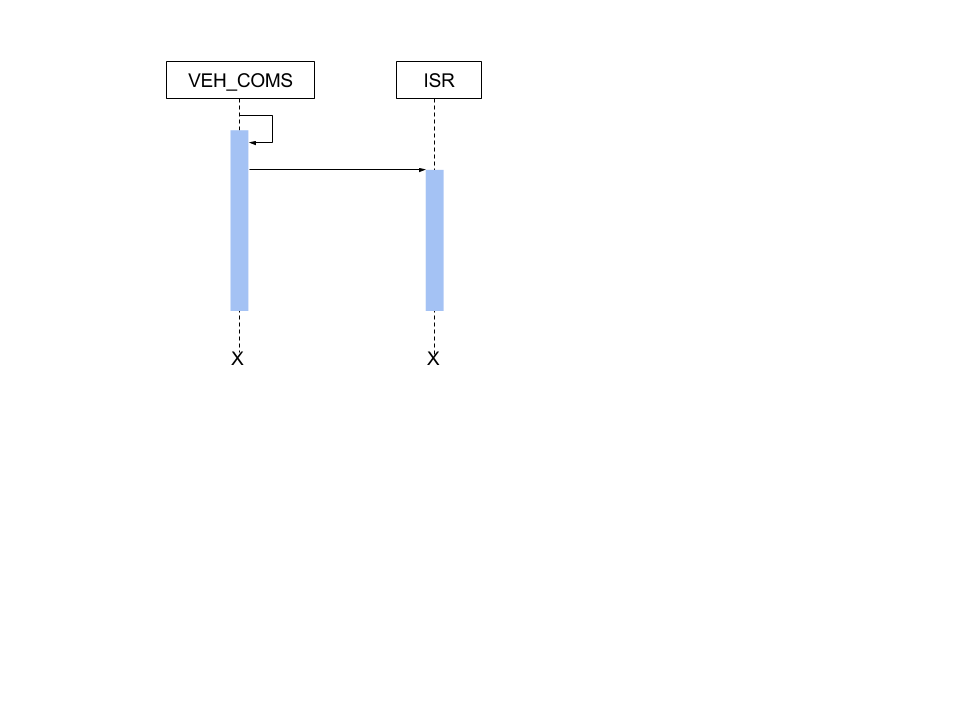
## 1.7 Vehicle Communication System

### 1.7.1 Design Specification

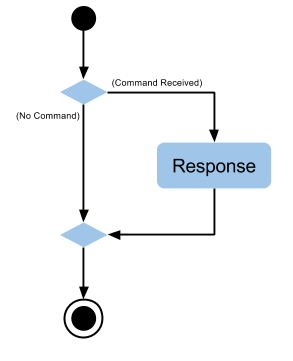
The vehicleComs takes care of a bidirectional communication between the satellite and the land based vehicle. The function accepts a pointer to void with a return of void. Response from the vehicle will be displayed using Hyperterm. Commands to vehicle, F, B, L, R, D S, I and H, and requests from vehicle, T and D, should be implemented.

### 1.72 Software Implementation

Vehicle coms subsystem detects messages from Hyperterm as interrupts. In vehicle coms task, the interrupt is enabled. In the ISR for vehicle coms subsystem, the message received is stored in the global variable command. Then the content of the command will be checked. If the command is one of F, B, L, R, D, S, I and H, the response A<sp Command sent> will be send to the Hyperterm display. Besides, when S or I is received, specific operations, i.e., start image capture and send image data, are performed. If the command is not an appropriate input, letter “N” will be send to the Hyperterm display. The send function is from the uart-echo example. A sequence diagram and a flow diagram are shown below.



**Figure 15: Vehicle communications sequence diagram**



**Figure 16: Vehicle communications sequence diagram**

## 1.8 Solar Panel Control System

### 1.8.1 Design Implementation

The solar panel task accepts a pointer to void and returns void. The solar panel control task manages the deployment and retraction of the satellite’s solar panels. During deployment or retraction the system outputs a PWM signal with period 500 ms to drive an electric motor. The motor drive ranges from full ON to full OFF in 5% increments based off keypad inputs. Once the solar panels are fully deployed or retracted an interrupt signal is sent to the system to terminate the drive command.

### 1.8.2 Software Implementation

The solar panel task is initiated from the power management task. Once a change in solar panel state is initiated the solar panel manage task is brought into the task queue. The PWM signal is configured, enabled, set and output using functions from the pwm.c file in driver-lib.a. The period is set at 125000 ticks for a 250kHz PWM clock. Initial trials proved 4000000 ticks with a 8MHz clock non-functional. In order to compensate for the high amount of ticks, a PWM clock is created using the global clock as a reference. The duty cycle of the PWM signal changes according to increment or decrement commands from the keypad. The duty cycle and period of the PWM signal are used to calculate the local variable, motorDrive. The motor drive determines the rate of deployment or retraction for the solar panel.The solar panel tasks tracks the current state of deployment and retraction using a local variable. Once the solar panel is fully retracted or deployed a flag is raised for the scheduler to remove the task from the task queue.

## 1.9 Keypad Control

### 1.9.1 Design Implementation

The Console Keypad function accepts a pointer to void with a return of void. The pointer in the task argument is re-cast as a pointer to the Console Keypad task’s data structure type before it can be dereferenced. The console keypad is used to manually control the solar panel drive motors in increments of ±5.0%. The keypad uses interrupts to scan for new key presses on a single press basis; The button must be released to trigger the next key press. The task is only scheduled during deployment or retraction of the solar panels.

### 1.9.2 Software Implementation

Using two GPIO pins as input type pins driving interrupts we set flags for Motor speed increment or Motor Speed Decrement. The Keypad task has no other functionality to ensure that the input is taken in one at a time. The keypad task sets either an increment flag or a decrement flag and waits for the solar panel sub system to reset these flags before accepting a new interrupt. By doing so we prevent triggering multiple presses caused by signal bounce or by holding down a corresponding input key.

## 1.10 Pirate Detection and Management

### 1.10.1 Design Implementation

Pirate Detection task accepts a pointer to void with a return of void. the pointer is re-case as a pointer to its data type. The system manages obstacles detected within 100 m and takes action if they are within 30 meters and drastic action if within 5 meters.

### 1.10.2 Software Implementation

The pirate detection task uses an external system with employs the use of an arduino UNO board and a sonar distance sensor. The board pings the sensor which returns a high signal with high time corresponding to the distance of the object detected. If the detected object is within range the board outputs a high signal to the main system signaling a warning interrupt. The kernel then schedules a pirate manage task which output a confirmation signal to the external system to take defensive action and fire on command via a serial input. The system fires phasors within 30 meters and photos within 5 meters.

## 1.11 Transport Distance

### 1.11.1 Design Implementation

The Transport Keypad function accepts a pointer to void with a return of void. The pointer in the task argument is recast as a pointer to the task’s data structure type before it can be dereferenced. The output detects falling edge of transport signal to determine the distance from the satellite. The task is only scheduled during with an interrupt triggered if the vehicle is within 1Km of the satellite.

### 1.11.2 Software Implementation

Using one GPIO pins as input type pin driving interrupt we set flags for scheduling and measuring the vehicle distance. The Distance task has no other functionality to ensure that the input is taken in one at a time and displayed on the console. The Distance task sets either a flag waits for the a new interrupt to complete the measurement. We use a global timer to determine the period of the incoming signal.

## 1.12 Command Manage

### 1.12.1 Design Implementation

The Command manage task accepts a pointer to void with a return of void. the pointer is recast as a pointer to the commands data struct. The task interprets incoming messages and formats outgoing messages in preparation for transportation. It transmits an error when receiving faulty commands. Additionally it builds the body of the transmit task with the relevant data needed.

### 1.12.2 Software Implementation

Using Switch case statements we organize our incoming data according to the the specification found in the Appendix. The first letter of the incoming data is read as the command type and the rest is read as the payload. using atoi() function we convert incoming strings of data into relevant data for thruster or any other subsystem. using sprintf() we parse outgoing data into string and output this data to the communications subsystem.

## 

## 1.13 Image Capture

### 1.13.1 Design Implementation

The ImageCapture task is to be scheduled in by the satellite. The task receives an AC signal output of a function generator in A/D channel 3. The AC signal represents the image data obtained by the mining vehicle. ImageCapture extracts the significant component frequency values using FFT. The frequency values are stored in a circular buffer that maintains old image data.

### 1.13.2 Software Implementation

The ImageCapture task is scheduled by the satellite. Once the satellite initiates ImageCapture with an ‘S’ command, the task processes the signal currently input into A/D channel 3. The task takes 256 samples of the input signal, spaced apart using a software delay. The software delay plays a huge role in determining the frequency range and frequency resolution of the readings. The software delay sets the sampling frequency for the ImageCapture task. The highest frequency signal that may be captured without aliasing is half the sample frequency. Through a few trial and error tests, the highest frequency that could be obtained is approximately 14000, with a sample frequency over 28000. The frequency resolution was approximately 110Hz. After the signal is sampled, the buffer containing the 256 AC samples and another buffer set to 0 are input into the optfft function for FFT processing. The function performs FFT on the signal data and returns an index containing the highest frequency value. The index, sampling frequency, and sample size of 256 for the optfft are used to calculate the signals frequency. This value is stored in a size 16 circular buffer and transmitted to the satellite or Earth.

# 2.0 PRESENTATION, DISCUSSION, AND ANALYSIS OF THE RESULTS

As per the design specification of the individual subsystems, the final version of the functional Satellite system performed according to what was expected. As showcased in the DEMO the OLED was successfully displaying all the target information and was able to switch between annunciation and display mode. The variables such as, Fuel Level and Battery Level, were constantly being updated at an accelerated rate for demonstration purposes. The Power Generation, Power Consumption, and Thruster systems changed appropriately and impacted associated Fuel Level, however battery level was now modelled by using a variable Power Supply with a range [0V-3.3V]. The interrupts designed to signal external and internal events activated accordingly and were able to trigger appropriate responses. The distance sensor worked using a function generator and the distance changed as the frequency changed.

In summary our design performed all the necessary tasks with synergy and compliance with the overall goal of the system. Each individual task acted within its designated parameters and performed its duty with robust precision.

# 3.0 ANALYSIS OF ANY ERRORS

An error occurred during the demo where the communications task caused the OLED to constantly switch between states when the incoming command was to toggle the OLED state. This error occurred because there were no safeguards implemented with the incoming command buffer. This has since been fixed.

# 4.0 ANALYSIS OF WHY THE PROJECT MAY NOT HAVE WORKED AND WHAT EFFORTS WERE MADE TO IDENTIFY THE ROOT CAUSE OF ANY PROBLEMS

The major cause to error was the lack of working pin on the board. additionally another fault was found during demo where the incoming command would constantly execute. Although we have no means to fix the board pin availability, we fixed the input command by using a flag to indicate if a command has been input and then the flag resets once it has been acknowledged. This fixed the error from happening again.

# 5.0 TEST PLAN

Testing was done to not only assess subsystem functionality but also empirically measure task execution time.

1. Use oscilloscope to measure period and duty cycle of PWM signals.
2. Apply variable voltage to ADC CH0 to test appropriate battery signal measurements
3. Trigger GPIO interrupts using PULL DOWN voltage circuit to ensure a single interrupt is triggered per button press and measure resulting outcome in the PWM signal for Solar Panel Motor Drive signal.
4. Trigger GPIO pin for battery interrupt with Ground signal to signal proper battery connection
5. Trigger Distance taks with Function generator to determine Upper (2.31Hz) and Lower(0.19Hz) bounds of the signal
6. Test pirate task with varying distance to measure if distance changes
7. Test Battery ADC’s to ensure proper sampling and accurate battery temp change
8. Test the connection of Internet
9. Trigger ethernet interrupt by sending data to web server and check the received data.

# 

# 6.0 TEST SPECIFICATION

## 6.1 PWM

Color entire screen white to check for faulty pixels

Assign random values to each variable within respective functions

Track changes to ensure accessibility and functionality within the debugger and by displaying on the OLED

## 6.2 GPIO

Measure output signal, using oscilloscope, to ensure period is according to specification

Apply 5V to All LEDs individually to ensure functionality

Apply varying voltage to ADC CH0 to ensure proper measurement is being taken

Apply Ground to GIOP interrupt pin for battery connection

## 6.3 Task Timing

The task timing were approximated by using a GPIO output HIGH when entering a task and output LOW when exiting the task. Each subsystem was put on a WHILE(1) loop with the appropriate system delay to measure the active time of the tasks. On average each task took approximately 2 - 10 milliseconds. Which resulted in approximately a 1% offset for PWM signals to the warning lights at the most compounded stage. The PWM period of 1 and 2 seconds adjusted to 1.01 and 2.02 seconds during assimilated version of the program.

## 6.4 Ethernet

Open the webpage to check the connection. Send a single character to the web server and display it on the OLED screen. Send a string to the web server and display it on the OLED screen.

## 6.5 Distance

Using a function generator we output a square wave to the appropriate pin to measure the associated distance. We varied the freq up and down by 0.1 Hz to determine the upper and lower bound of the signal. Beyond this range the output distance was constant.

## 6.6 Pirate

Using a sonar sensor we moved closer and farther away from a flat wall and measured the distance. We varied the distance between 40 and 0 meters to determine the system functionality.

# 7.0 TEST CASES

Using the OLED printing function Print the uppercase letter “I” over the entire screen and visually inspect OLED for any defects or missing pixels. Using the Stringdraw function of the OLED Library to ensure proper functionality. Additionally using the Clear() function to ensure all pixels clear appropriately.

Using the rand() function assign values when functions are called within each sub-system via the pointers within respective structs are passed to the functions. Using the Debugger track the changes along with printing to the OLED for easier and quicker readability. If fault is found during visual inspection perform a finer check using the debugger and stepping through each line.

Using the Oscilloscope measure the PWM signal of the LED output signal from the GPIO port, first without any additional subsystem running then with all the subsystems running. Calculate the difference in the period (approx 1% variation depending on specific activity of subroutine) and adjust the values within the code to accommodate for the variation. The PWM signal to the LED’s for the warning subsystem alone was 1 sec and 2 sec but was adjusted to 1.01 and 2.02 seconds when combined with the other subsystems. Additionally the motor drive signal was also measured using the oscilloscope in conjunction with the keypad interrupts to measure how the signal behaved when instructed to change based on the keypad input. We initially found that the keypad caused the signal to degrade severely due to signal bounce as well as the interrupt resetting well before the major cycle or minor cycle is complete. We adjusted to this by employing a hardware delay before resetting the interrupt flag effectively only taking in one button press at a time. A multi press was configured to ensure one could no longer hold a button and shut of the solar panel motor drive. Furthermore we implemented safeguards around the Solar Panel motor drive signal to prevent it from reaching 0 or exceeding 100% in the case of incorrect user input which was initially a source of the signal degradation.

Using the power source apply a 5V signal to each LED individually to ensure LED functionality. Additionally we applied a variable signal to ADC CH0 range[0-3.3V] to ensure measurement of the signal using a sequence 3 and only 8 sample points as measurement. Finally the external trigger for the battery connection was tested via grounding the appropriate GPIO pin which is set up as a PULL down input interrupt pin.

# 

# 8.0 SUMMARY

The lab outlines all major subsystems required to build an exploratory mining satellite. This phase focused on the design and development of the Veh Distance, Pirate Sensor, Pirate Mange, Image Capture, and Communication VIA Internet. The overall flow of control within the system as well a developed drivers for managing the basic satellite operations, status display , and alarm and warning annunciation functions have been outlined in detail in addition to the documentation for the additional systems. The report described and detailed all the subsystems and above sections provide full specifications and design documentation for the systems under current development. Both a high-level system architecture and detailed system specifications necessary to perform all the subset of the necessary control functions required by the client.

# 9.0 CONCLUSION

The implementation of the design specifications to the overall system required attention to detail and a strong debugging mindset. The project built upon the existing framework of the previous skeletal base developed in previous projects. We have successfully modified the skeletal framework to accommodate for additional safety features and modified functionality to develop the final project. The system is now a fully functional Satellite system with hard and soft real time constraints and a fully functional communications and defense system.

# 

# 10.0 WORK DIVISION

Jitesh Bhogal

Warning Management, Display Comms, Debugging, Command, Pirate, Veh Distance, OLED

Thomas Wei

Power Management, Thruster Management, ImageCapture, SolarPanel, BatteryTemperature, Testing

Xinyu Sui

System Kernel, Satellite Communication, Vehicle Communication, Display Mode Switch, Webserver, Testing

# 11.0 APPENDIX

## 11.1 Source Code

/\*

Thomas Wei, Jitesh Bhogal, Xinyu Sui

EE 472 Final project main file

\*/

#define mainINCLUDE\_WEB\_SERVER 0

/\* Standard includes. \*/

#include <stdio.h>

#include <math.h>

/\* Scheduler includes. \*/

#include "FreeRTOS.h"

#include "task.h"

#include "queue.h"

#include "semphr.h"

// Project Files

#include "myHeader.h"

/\* Hardware library includes. \*/

#include "hw\_memmap.h"

#include "hw\_types.h"

#include "hw\_sysctl.h"

#include "debug.h"

#include "hw\_ints.h"

#include "hw\_nvic.h"

#include "gpio.h"

#include "interrupt.h"

#include "systick.h"

#include "timer.h"

#include "pwm.h"

#include "adc.h"

#include "uart.h"

#include "sysctl.h"

#include "gpio.h"

#include "grlib.h"

#include "rit128x96x4.h"

#include "osram128x64x4.h"

#include "formike128x128x16.h"

// Button interrupt FLAGS

Bool g\_ulFlags=TRUE, mode=TRUE;

/\* Demo app includes. \*/

#include "lcd\_message.h"

#include "bitmap.h"

#define mainCHECK\_DELAY ( ( portTickType ) 5000 / portTICK\_RATE\_MS )

// Size of the stack allocated to the uIP task.

#define mainBASIC\_WEB\_STACK\_SIZE ( configMINIMAL\_STACK\_SIZE \* 3 )

// The OLED task uses the sprintf function so requires a little more stack too.

#define mainOLED\_TASK\_STACK\_SIZE ( configMINIMAL\_STACK\_SIZE + 50 )

// Task priorities.

#define mainQUEUE\_POLL\_PRIORITY ( tskIDLE\_PRIORITY + 2 )

#define mainCHECK\_TASK\_PRIORITY ( tskIDLE\_PRIORITY + 3 )

#define mainSEM\_TEST\_PRIORITY ( tskIDLE\_PRIORITY + 1 )

#define mainBLOCK\_Q\_PRIORITY ( tskIDLE\_PRIORITY + 2 )

#define mainCREATOR\_TASK\_PRIORITY ( tskIDLE\_PRIORITY + 3 )

#define mainINTEGER\_TASK\_PRIORITY ( tskIDLE\_PRIORITY )

#define mainGEN\_QUEUE\_TASK\_PRIORITY ( tskIDLE\_PRIORITY )

// The maximum number of messages that can be waiting for display at any one time.

#define mainOLED\_QUEUE\_SIZE ( 10 )

// Dimensions the buffer into which the jitter time is written.

#define mainMAX\_MSG\_LEN 25

/\*

The period of the system clock in nano seconds. This is used to calculate

the jitter time in nano seconds.

\*/

#define mainNS\_PER\_CLOCK ( ( unsigned portLONG ) ( ( 1.0 / ( double ) configCPU\_CLOCK\_HZ ) \* 1000000000.0 ) )

// Constants used when writing strings to the display.

#define mainCHARACTER\_HEIGHT ( 9 )

#define mainMAX\_ROWS\_128 ( mainCHARACTER\_HEIGHT \* 14 )

#define mainMAX\_ROWS\_96 ( mainCHARACTER\_HEIGHT \* 10 )

#define mainMAX\_ROWS\_64 ( mainCHARACTER\_HEIGHT \* 7 )

#define mainFULL\_SCALE ( 15 )

#define ulSSI\_FREQUENCY ( 3500000UL )

#define ABS(x) (((x)<0)?(-(x)):(x))

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

/\*-----------------------------------------------------------\*/

/\*

\* The task that handles the uIP stack. All TCP/IP processing is performed in

\* this task.

\*/

extern void vuIP\_Task( void \*pvParameters );

/\*

\* The display is written two by more than one task so is controlled by a

\* 'gatekeeper' task. This is the only task that is actually permitted to

\* access the display directly. Other tasks wanting to display a message send

\* the message to the gatekeeper.

\*/

static void vOLEDTask( void \*pvParameters );

/\*

\* Configure the hardware .

\*/

static void prvSetupHardware( void );

/\*

\* Configures the high frequency timers - those used to measure the timing

\* jitter while the real time kernel is executing.

\*/

extern void vSetupHighFrequencyTimer( void );

/\*

\* Hook functions that can get called by the kernel.

\*/

void vApplicationStackOverflowHook( xTaskHandle \*pxTask, signed portCHAR \*pcTaskName );

void vApplicationTickHook( void );

/\*

three dummy tasks of different priorities that simply run, announce

themselves, then sleep

\*/

void DisplayManage(void \*vParameters);

void PowerManage(void \*vParameters);

void ThrusterManage(void \*vParameters);

void ComsManage(void \*vParameters);

void SolarPanel(void \*vParameters);

void WarningManage(void \*vParameters);

void VComsManage(void \*vParameters);

void keypadManage(void\*data);

void Schedule(void);

void Distance(void\* Data);

void Pirate(void);

void UARTSend(const unsigned char \*pucBuffer, unsigned long ulCount);

void ImageCapture(void \*vParameters);

signed int optfft(signed int x[256], signed int y[256]);

void PirateManage(void \* Pirates);

void delay(unsigned long aValue);

/\*-----------------------------------------------------------\*/

/\*

The queue used to send messages to the OLED task.

\*/

xQueueHandle xOLEDQueue;

/\*-----------------------------------------------------------\*/

//initialize data structs//

void InitData( void );

//Global Variables

int globalCounter=0;

unsigned short batteryLevel=32, fuelLevel=100, powerConsumption=1, powerGeneration=0, BattTempKelvin=200;

unsigned int batteryBuffer[16];

unsigned int thrusterCommand=65418, \*batteryLevelPtr=batteryBuffer;

Bool solarPanelState=FALSE, solarPanelDeploy=FALSE, solarPanelRetract=FALSE,

fuelLow=FALSE, batteryLow=FALSE, driveMotorSpeedInc=FALSE,

driveMotorSpeedDec=FALSE, Measure=FALSE, Clear=FALSE, resVar=FALSE;

Bool UpFlag=FALSE,DownFlag=FALSE,StartTimer=FALSE,StopTimer=FALSE, distFLAG=FALSE,

TFLAG=FALSE, BTempAck=FALSE, Pirates = FALSE, startCapture=FALSE, finishCapture=FALSE;

char command=NULL, response=NULL, Ctype=NULL;

volatile unsigned long g\_ulGPIOb , distance=100;

char payload[20],Response[30];

static Bool Trigger=TRUE;

power myPower;

solarPanel mySolarPanel;

keypad myKeypad;

thruster myThruster;

coms myComs;

vehComs myVehComs;

display myDisplay;

warning myWarning;

Command myCommand;

image myImage;

// Bool BattOTemp = TRUE

Bool battOverTemp = TRUE;

unsigned short battTemp = 300, secondTemp = 300;

signed int rawDataBuffer[256];

signed int imageBuffer[256];

signed int \*imageDataRawPtr = rawDataBuffer;

signed int \*imageDataPtr = imageBuffer;

signed int pendingBuffer[16];

signed int \*imageDataPending = pendingBuffer;

unsigned short peakFreq = 0;

unsigned int battTempBuffer[16];

unsigned int \*battTempPtr=battTempBuffer;

//init display data

xTaskHandle commHandle=NULL;

int main( void )

{

InitData();

prvSetupHardware();

/\*

Create the queue used by the OLED task. Messages for display on the OLED

are received via this queue.

\*/

xOLEDQueue = xQueueCreate( mainOLED\_QUEUE\_SIZE, sizeof( xOLEDMessage ) );

xTaskCreate(Schedule,"Schedule",100,NULL,10,NULL);

vSetupHighFrequencyTimer();

vTaskStartScheduler();

return 0;

}

void Schedule(void){

static Bool solarPanelInserted=FALSE;

xTaskHandle solarPanelHandle=NULL, keypadHandle=NULL, imageHandle=NULL;

xTaskCreate( vOLEDTask, ( signed portCHAR \* ) "OLED", mainOLED\_TASK\_STACK\_SIZE, NULL, tskIDLE\_PRIORITY, NULL );

xTaskCreate(DisplayManage, "DisplayManage", 75,&(myDisplay), 5,NULL);

xTaskCreate(PowerManage, "PowerManage", 75,&(myPower), 1,NULL);

xTaskCreate(ThrusterManage, "ThrusterManage", 75,&(myThruster), 2,NULL);

if( SysCtlPeripheralPresent( SYSCTL\_PERIPH\_ETH ) )

xTaskCreate(ComsManage, "ComsManage", 400,&(myComs), 10,NULL);

xTaskCreate(WarningManage, "WarningManage", 100,&(myWarning), 3,NULL);

xTaskCreate(VComsManage, "VComsManage", 50,&(myVehComs), 4,NULL);

xTaskCreate(Distance, "Distance", 50,&(myVehComs), 8,NULL);

xTaskCreate(CommandManage,"Commandmanage",75,&(myCommand),9,&commHandle);

xTaskCreate(ImageCapture, "ImageCapture", 600, &(myImage), 11, NULL);

int z=0;

while(1){

if(g\_ulGPIOb){

if(!solarPanelInserted){

if(solarPanelHandle==NULL){

xTaskCreate(SolarPanel, "SolarPanel", 75,&(mySolarPanel), 6,&solarPanelHandle);

xTaskCreate(keypadManage,"KeyPad",50,&(myKeypad),7,&keypadHandle);

} else{

vTaskResume(solarPanelHandle);

vTaskResume(keypadHandle);

}

solarPanelInserted=!solarPanelInserted;

} else{

vTaskSuspend(solarPanelHandle);

vTaskSuspend(keypadHandle);

}

g\_ulGPIOb=0;

}

if(startCapture){

if(imageHandle==NULL)

xTaskCreate(ImageCapture, "ImageCapture", 600, &(myImage), 11, &imageHandle);

else

vTaskResume(imageHandle);

} else if(imageHandle!=NULL)

vTaskSuspend(imageHandle);

startCapture=FALSE;

z=GPIOPinRead(GPIO\_PORTD\_BASE,GPIO\_PIN\_2);

if (z==4)

{

GPIOPinWrite(GPIO\_PORTD\_BASE, GPIO\_PIN\_5, 0xFF);

}else if (z==0){

GPIOPinWrite(GPIO\_PORTD\_BASE, GPIO\_PIN\_5, GPIO\_LOW\_LEVEL);//ENABLE MANAGE TASK

}

globalCounter++;

vTaskDelay(50);

}

}

void CommandManage(void\*mydata)

{

Command \*PCDat=(Command\*)mydata;

static Bool OLED=TRUE;

while(1)

{

PCDat=(Command\*)mydata;

switch (\*(PCDat->cType))

{

case 'P':

sprintf(Response,"A stop",\*PCDat->batteryLevel);

IntDisable(INT\_GPIOC|INT\_GPIOD|INT\_GPIOE|INT\_GPIOF|INT\_GPIOG);

break;

case 'S':

sprintf(Response,"A start",\*PCDat->batteryLevel);

IntEnable(INT\_GPIOC|INT\_GPIOD|INT\_GPIOE|INT\_GPIOF|INT\_GPIOG);

break;

case 'D':

sprintf(Response,"A OLED",\*PCDat->batteryLevel);

switch (OLED)

{

case TRUE:

RIT128x96x4Disable();

OLED=FALSE;

break;

case FALSE:

RIT128x96x4Init(ulSSI\_FREQUENCY) ;

OLED=TRUE;

break;

default:

break;

}

break;

case 'T':

sprintf(Response,"A thruster command",\*PCDat->batteryLevel);

\*(PCDat->thrusterCommand)=atoi(PCDat->Payload);

break;

case 'M':

\*(PCDat->TransmitFlag)=TRUE;

switch(\*(PCDat->Payload))

{

case 'B':

sprintf(Response,"BatteryLevel:.%u",\*PCDat->batteryLevel);

break;

case 'F':

sprintf(Response,"FuelLevel:.%u",\*PCDat->fuelLevel);

break;

default:

sprintf(Response,"B.%u.F.%u",\*PCDat->batteryLevel,\*PCDat->fuelLevel);

break;

}

break;

// case 'A':

// \*(PCDat->TransmitFlag)=TRUE;

// sprintf(Response,"%c.%u.ACK",\*PCDat->cType,\*PCDat->Payload);

// break;

default:

\*(PCDat->TransmitFlag)=TRUE;

(PCDat->Response)="ERROR";

break;

}

vTaskDelay(50);

}

}

void DisplayManage(void \*vParameters)

{

display \*mydisp = (display\*)vParameters;

//static portCHAR cMessage[ mainMAX\_MSG\_LEN ];

xOLEDMessage xMessage;

unsigned int x=0;

unsigned int y=0;

unsigned int xI=100;

unsigned int yI=0;

xMessage.X\_Locs=&x;

xMessage.Y\_Locs=&y;

xMessage.X\_LocI=&xI;

xMessage.Y\_LocI=&yI;

static int i=0;

const char \*T1Text = "DisplayManage is running\n\r";

//sprintf(cMessage,"%u",GPIOPinRead(GPIO\_PORTC\_BASE, GPIO\_PIN\_6));

// vOLEDStringDraw( cMessage, 0, 80, mainFULL\_SCALE );

//long button=0;

while(1)

{

//RIT128x96x4StringDraw("dsp ", 10, 80, 15);

// button =GPIOPinRead(GPIO\_PORTF\_BASE, GPIO\_PIN\_1);

if (!g\_ulFlags){i=0;vTaskDelay(50);g\_ulFlags=TRUE; Clear=TRUE;}

// goto exit;

switch (mode)

{

case TRUE:

xMessage.msg=1;

switch (i){

case 0:

xMessage.pIMessage=(mydisp->batteryLevel);

x=0; y=0; xI=105; yI=0;

xMessage.pcMessage="Battery ";

i++;

goto exit;

break;

case 1:

xMessage.pIMessage=(mydisp->fuelLevel);

x=0; y=12; xI=105; yI=12;

xMessage.pcMessage="Fuel ";

i++;

goto exit;

break;

case 2:

xMessage.pIMessage=(mydisp->powerConsumption);

x=0; y=24; xI=105; yI=24;

xMessage.pcMessage="Power Consumption";

i++;

goto exit;

break;

case 3:

xMessage.pIMessage=(mydisp->powerGeneration);

x=0; y=36; xI=105; yI=36;

xMessage.pcMessage="Power Generation";

i++;

goto exit;

break;

case 4:

xMessage.pIMessage=(mydisp->solarPanelState);

x=0; y=48; xI=105; yI=48;

xMessage.pcMessage="Solar Panel";

i=5;

goto exit;

break;

case 5:

xMessage.pIMessage=(mydisp->VehDist);

x=0; y=60; xI=105; yI=60;

xMessage.pcMessage="Veh Distance";

i=6;

goto exit;

break;

case 6:

xMessage.pIMessage=(mydisp->battTemp);

x=0; y=72; xI=105; yI=72;

xMessage.pcMessage="Battery Temp 1";

i=7;

goto exit;

// case 7:

// xMessage.pIMessage = (mydisp->secondTemp);

// x=0; y=84; xI=105; yI=84;

// xMessage.pcMessage = "Battery Temp 2 (K)";

// i=0;

// GPIOPinIntClear(GPIO\_PORTF\_BASE, GPIO\_PIN\_1);

// goto exit;

case 7:

xMessage.pIMessage = (mydisp->peakFreq);

x=0; y=84; xI=70; yI=84;

xMessage.pcMessage = "PeakFreq";

i=0;

GPIOPinIntClear(GPIO\_PORTF\_BASE, GPIO\_PIN\_1);

goto exit;

}

break;

case FALSE:

xMessage.msg=0;

switch(i){

case 0:

xMessage.pIMessage=(mydisp->batteryLevel);

x=0; y=0; xI=105; yI=0;

xMessage.pcMessage="Battery ";

i++;

goto exit;

break;

case 1:

xMessage.pIMessage=(mydisp->fuelLevel);

x=0; y=15; xI=105; yI=15;

xMessage.pcMessage="Fuel ";

i++;

goto exit;

break;

case 2:

xMessage.pIMessage=(mydisp->batteryLow);

x=0; y=30; xI=105; yI=30;

if(\*(mydisp->batteryLow)){

xMessage.pcMessage="Battery Low ";

}else{xMessage.pcMessage=" "; }

i++;

goto exit;

break;

case 3:

xMessage.pIMessage=(mydisp->fuelLow);

x=0; y=45; xI=105; yI=45;

if(\*(mydisp->fuelLow)){

xMessage.pcMessage="Fuel Low ";

}else{xMessage.pcMessage=" "; }

i=4;

goto exit;

break;

case 4:

xMessage.pIMessage=(mydisp->battTemp);

x=0; y=60; xI=105; yI=60;

xMessage.pcMessage="Battery Temp 1";

i=5;

goto exit;

case 5:

xMessage.pIMessage = (mydisp->secondTemp);

x = 0; y = 75; xI = 105; yI = 75;

xMessage.pcMessage = "Battery Temp 2";

i=0;

GPIOPinIntClear(GPIO\_PORTF\_BASE, GPIO\_PIN\_1);

goto exit;

}

exit:

xQueueSend( xOLEDQueue, &xMessage, 0 );

vTaskDelay(50);

}

}

}

void PowerManage(void \*vParameters)

{

power \* myPower = (power\*) vParameters; // creates a pointer to void

// // to the myPower data struct

static int funcCount = 0; // keeps track of the function calls

static Bool incMethod = FALSE; // determines how powerConsumption is

// incremented

unsigned long ulADC0\_Value[1];

unsigned long ulTemp\_ValueC;

int prevMax = 373;

int currentMax = 0;

int index = 0, k = 0;

unsigned long ulADC1\_Value[1];

unsigned long ulADC2\_Value[1];

while(1)

{

HWREG(NVIC\_SW\_TRIG)=INT\_GPIOG-16;

//char temp[20];

if (Measure)

{

// ADCSequenceDataGet(ADC\_BASE, 0, g\_ulIndex);

// \*(myPower -> batteryLevel) = g\_ulIndex;

delay(5);

if(index > 15) {index = 0;}

ADCProcessorTrigger(ADC\_BASE, 3);

while(!ADCIntStatus(ADC\_BASE, 3, false)) { }

ADCIntClear(ADC\_BASE, 3);

ADCSequenceDataGet(ADC\_BASE, 3, ulADC0\_Value);

\*myPower -> batteryLevel = 32\*ulADC0\_Value[0]/(0x3FF);

(myPower -> batteryLevelPtr)[index] = \*myPower -> batteryLevel;

Measure=FALSE;

// \*myPower -> batteryLevel = 0;

}

else

{

\*myPower -> batteryLevel= \*myPower -> batteryLevel;

(myPower -> batteryLevelPtr)[index] = \*myPower -> batteryLevel;

}

if (\*(myPower->solarPanelState))

{

delay(4);

if(k > 15) {k = 0;}

ADCProcessorTrigger(ADC\_BASE, 2);

while (!ADCIntStatus(ADC\_BASE, 2, false)) {}

ADCIntClear(ADC\_BASE, 2);

ADCSequenceDataGet(ADC\_BASE, 2, ulADC1\_Value);

\*(myPower->battTemp) = (13 \* ulADC1\_Value[0] / (0x3FF) / 4) \* 32 + 33 + 273;

(myPower->battTempPtr)[k] = 325\*ulADC1\_Value[0] / (0x3FF);

k++;

if (\*(myPower->battTemp) > currentMax)

{

currentMax = (myPower->battTemp)[0];

}

if ((\*(myPower->battTemp) \* 6 / 5) > prevMax)

{

\*(myPower->battOverTemp) = TRUE;

}

if(k > 15) {k = 0;}

ADCProcessorTrigger(ADC\_BASE, 1);

while (!ADCIntStatus(ADC\_BASE, 1, false)) {}

ADCIntClear(ADC\_BASE, 1);

ADCSequenceDataGet(ADC\_BASE, 1, ulADC2\_Value);

\*(myPower->secondTemp) = (13 \* ulADC2\_Value[0] / (0x3FF) / 4) \* 32 + 33 + 273;

(myPower->battTempPtr)[k] = 325\*ulADC2\_Value[0] / (0x3FF);

k++;

if (\*(myPower->secondTemp) > currentMax)

{

currentMax = \*(myPower->secondTemp);

}

if ((\*(myPower->secondTemp) \* 6 / 5) > prevMax)

{

\*(myPower->battOverTemp) = TRUE;

}

prevMax = currentMax;

currentMax = 0;

}

//RIT128x96x4StringDraw("Pwr ", 10, 80, 15);

if(\*(myPower -> solarPanelState)) // increments powerGeneration based on the

// current

{

if(\*(myPower->batteryLevel) > 31) // switches the solarPanelState if battery above 95

{ // resets the powerGeneration

if(!(\*myPower->solarPanelRetract)){

g\_ulGPIOb=1;

}

(\*(myPower->solarPanelRetract)) = TRUE;

(\*(myPower->powerGeneration)) = 0;

}

else

{

(\*(myPower->powerGeneration)) += 2;// increments powerGeneration based on current

if(funcCount % 2) // function count and batteryLevel

{

(\*(myPower->powerGeneration))--;

if(\*(myPower->batteryLevel) > 18)

{

(\*(myPower->powerGeneration))--;

}

}

}

}

else

{

if((\*(myPower->batteryLevel)) <= 4) // switches solarPanelState if battery below 11

{

if(!\*(myPower -> solarPanelState))

{

if(!(\*myPower->solarPanelDeploy))

{

g\_ulGPIOb=1;

}

(\*(myPower->solarPanelDeploy)) = TRUE;

}

(\*(myPower -> powerGeneration)) = 13;// when solar panel is deployed powerGeneration

// goes to 13 to ensure battery does not hit 0

}

}

if(\*(myPower->powerConsumption) < 5)

{

incMethod = FALSE; // switches the method of incrementation when

// powerConsumption goes below 5

}

else if(\*(myPower->powerConsumption) > 10)

{

incMethod = TRUE; // switches the method of incrementation when

// powerConsumption goes above 10

}

if(incMethod) // increments powerConsumption based on the

// current increment method and funcCount

{

if(funcCount % 2)

{

(\*(myPower->powerConsumption))++;

}

else

{

(\*(myPower->powerConsumption)) -= 2;

}

}

else

{

if(funcCount % 2)

{

(\*(myPower->powerConsumption))--;

}

else

{

(\*(myPower->powerConsumption)) += 2;

}

}

funcCount++;

if ((\*(myPower->batteryLevel))<10)

{

\*myPower->batteryLow = TRUE;

} else {\*myPower->batteryLow = FALSE;}

//DEBUG

//sprintf(temp,"%u",\*myPower -> batteryLevel );

//RIT128x96x4StringDraw(" ", 10,80, 15);

//RIT128x96x4StringDraw(temp, 10,80, 15);

//DEBUG

vTaskDelay(100);

}

}

void ThrusterManage(void \*vParameters)

{

unsigned long period = 125000;

static unsigned long dutyCycle = 62500;

thruster \* myThruster = (thruster\*) vParameters; // creates a pointer to void

// to the myThruster data struct

int duration;

int magnitude;

while(1)

{

duration = \*(myThruster -> thrusterCommand) & 0xff00;

magnitude = \*(myThruster -> thrusterCommand) & 240;

dutyCycle = magnitude \* period / 240;

if(dutyCycle >= 0.99 \* period)

{

dutyCycle = 0.99 \* period;

}

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_PWM);

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOG);

GPIOPinTypeGPIOInput(GPIO\_PORTG\_BASE, GPIO\_PIN\_1);

GPIOPinWrite(GPIO\_PORTG\_BASE, GPIO\_PIN\_1, 0);

GPIOPinTypePWM(GPIO\_PORTG\_BASE, GPIO\_PIN\_1);

GPIOPadConfigSet(GPIO\_PORTG\_BASE, GPIO\_PIN\_1, GPIO\_STRENGTH\_8MA, GPIO\_PIN\_TYPE\_STD);

PWMGenConfigure(PWM0\_BASE, PWM\_GEN\_0, PWM\_GEN\_MODE\_UP\_DOWN | PWM\_GEN\_MODE\_NO\_SYNC);

PWMGenPeriodSet(PWM0\_BASE, PWM\_GEN\_0, period);

PWMPulseWidthSet(PWM0\_BASE, PWM\_OUT\_1, dutyCycle);

PWMGenEnable(PWM0\_BASE, PWM\_GEN\_0);

PWMOutputState(PWM0\_BASE, PWM\_OUT\_1\_BIT, 1);

// RIT128x96x4StringDraw("thrust ", 10, 80, 15);

if(globalCounter % 50 == 0)

{

if ((magnitude != 0) && (duration != 0) && (\*(myThruster -> fuelLevel) != 0))

{

if (duration >= 1280)

{

int limit = \*(myThruster -> fuelLevel) - (100 / (21600 / magnitude) \* 5);

if (limit <= 0) // if fuel level will drop below 0 in the next 5 seconds

{

\*(myThruster -> fuelLevel) = 0;

\*(myThruster -> thrusterCommand) = 0x0000; // thrusterCommand set to zeros to avoid unncesscary iteration of function

}

else

{

//(\*fuelLevel) -= 100 / (37324800 / magnitude) \* 5; // document specified fuelLevel decrease rate

\*(myThruster -> fuelLevel) -= 100 / (21600 / magnitude) \* 5; // fuelLevel decrease rate depends purely on magnitude

// 21600 = (5% \* 1800 seconds) / 240 \* 100

// fuel lasts for 1800 seconds at 5% (for demo purposes),

// is normalized by (/240) and the percentage magnitude is multiplied by 100

// similar calculations are used to derive the document

// specific fuelLevel consumption

\*(myThruster -> thrusterCommand) -= (256 \* 5); //subtracts 5 seconds of remaining runtime for major cycle looping

}

}

else

{

int limit = \*(myThruster -> fuelLevel) - (100 / (2160 / magnitude) \* (duration / 256));

if (limit <= 0) // if the fuel level will drop below 0 in the remaining duration

{

\*(myThruster -> fuelLevel) = 0;

\*(myThruster -> thrusterCommand) = 0x0000; // thrusterCommand set to zeros to avoid unncesscary iteration of function

}

else

{

//(\*fuelLevel) -= 100 / ((37324800 / magnitude) \* (duration / 256));

\*(myThruster -> fuelLevel) -= 100 / ((2160 / magnitude) \* (duration / 256)); // decreases fuelLevel by magnitude and remaining duration

\*(myThruster -> thrusterCommand) -= duration; // subtracts the remaining duration from thrusterCommand

}

}

if (\*(myThruster -> fuelLevel) <= 10) // sets fuelLow to TRUE when fuel is below or equal to 10

{

\*(myThruster -> fuelLow) = TRUE;

}

else // sets fuelLow to FALSE when fuel is greater than 10

{

\*(myThruster -> fuelLow) = FALSE;

}

}

else

{

PWMGenDisable(PWM0\_BASE, PWM\_GEN\_0);

PWMOutputState(PWM0\_BASE, PWM\_OUT\_1\_BIT, 0);

}

}

vTaskDelay(1000);

}

}

void vOLEDTask( void \*pvParameters )

{

xOLEDMessage xMessage;

unsigned portLONG ulY, ulMaxY;

static portCHAR cMessage[ mainMAX\_MSG\_LEN ];

extern volatile unsigned portLONG ulMaxJitter;

unsigned portBASE\_TYPE uxUnusedStackOnEntry;

const unsigned portCHAR \*pucImage;

static int zz=0;

// Functions to access the OLED.

void ( \*vOLEDInit )( unsigned portLONG ) = NULL;

void ( \*vOLEDStringDraw )( const portCHAR \*, unsigned portLONG, unsigned portLONG, unsigned portCHAR ) = NULL;

void ( \*vOLEDImageDraw )( const unsigned portCHAR \*, unsigned portLONG, unsigned portLONG, unsigned portLONG, unsigned portLONG ) = NULL;

void ( \*vOLEDClear )( void ) = NULL;

vOLEDInit = RIT128x96x4Init;

vOLEDStringDraw = RIT128x96x4StringDraw;

vOLEDImageDraw = RIT128x96x4ImageDraw;

vOLEDClear = RIT128x96x4Clear;

ulMaxY = mainMAX\_ROWS\_96;

pucImage = pucBasicBitmap;

// Just for demo purposes.

uxUnusedStackOnEntry = uxTaskGetStackHighWaterMark( NULL );

ulY = ulMaxY;

/\* Initialise the OLED \*/

vOLEDInit( ulSSI\_FREQUENCY );

long button=0;

xMessage.msg=1;

while( 1 )

{

// RIT128x96x4StringDraw("Oled ", 10, 80, 15);

if (Clear){

vOLEDClear();Clear=FALSE;}

// Wait for a message to arrive that requires displaying.

xQueueReceive( xOLEDQueue, &xMessage, portMAX\_DELAY );

// Write the message on the next available row.

// ulY += mainCHARACTER\_HEIGHT;

// button=GPIOPinRead(GPIO\_PORTF\_BASE, GPIO\_PIN\_1);

// if( button ==0)

// {

// if (zz==0){ //ulY = mainCHARACTER\_HEIGHT;

// vOLEDClear();zz++;

// }else if (zz==50){zz=0;} else {zz++;}

// }

// sprintf(cMessage,"%u",GPIOPinRead(GPIO\_PORTF\_BASE, GPIO\_PIN\_1));

//vOLEDStringDraw( cMessage, 0, 80, mainFULL\_SCALE );

// Display the message

// vOLEDClear();

//if (xMessage.msg==1)

//{

sprintf( cMessage, "%u", \*(xMessage.pIMessage));

vOLEDStringDraw( " ", \*xMessage.X\_LocI, \*xMessage.Y\_LocI, mainFULL\_SCALE );

vOLEDStringDraw( cMessage, \*xMessage.X\_LocI, \*xMessage.Y\_LocI, mainFULL\_SCALE );

sprintf( cMessage, "%s", xMessage.pcMessage);

vOLEDStringDraw( cMessage, \*xMessage.X\_Locs, \*xMessage.Y\_Locs, mainFULL\_SCALE );

//}

//else {

sprintf( cMessage, "%u", GPIOPinRead(GPIO\_PORTD\_BASE, GPIO\_PIN\_2));

vOLEDStringDraw( cMessage, 50, 0, mainFULL\_SCALE );

}

}

/\*-----------------------------------------------------------\*/

unsigned int receive(){

return randomInteger(0, COMMAX);

}

//void ComsManage(void \*vParameters)

//{

// coms \*myComs=(coms\*)vParameters;

// char temp;

// static Bool capture=FALSE;

// while(1)

// {

//// //send();

//// // receive message per major cycle, frequency 5/100

// if (globalCounter%1000==0&&randomInteger(0, 1000)<3) // 750

// {

// \*(myComs->thrusterCommand)=receive();

// switch(receive()%7){

// case 0:

// temp='F';

// break;

// case 1:

// temp='B';

// break;

// case 2:

// temp='L';

// break;

// case 3:

// temp='R';

// break;

// case 4:

// temp='D';

// break;

// case 5:

// temp='H';

// break;

// default:

// if(capture)

// {

// temp = 'I';

// }

// else

// {

// temp = 'S';

// }

// capture = (!capture);

// break;

// }

// if(\*(myComs->response)!=temp){

// resVar=!resVar;

// \*(myComs->response)=temp;

// }

// }

// vTaskDelay(1000);

// }

//}

void VComsManage(void \*vParameters)

{

vehComs \*myVehComs=(vehComs\*)vParameters;

while(1)

{

if(finishCapture){

UARTCharPutNonBlocking(UART0\_BASE, 'W');

finishCapture=FALSE;

}

// if(resVar){

// switch(\*myVehComs->response){

// case 'F':

// UARTSend("A Forward", 9);

// break;

// case 'B':

// UARTSend("A Back", 6);

// break;

// case 'L':

// UARTSend("A Left", 6);

// break;

// case 'R':

// UARTSend("A Right", 7);

// break;

// case 'D':

// UARTSend("A Start", 7);

// break;

// case 'H':

// UARTSend("A Stop", 6);

// case 'S':

// UARTSend("A Start capture", 15);

// ADCSequenceEnable(ADC\_BASE, 0);

// break;

// case 'I':

// UARTSend("A Image", 7);

// ADCSequenceDisable(ADC\_BASE, 0);

// break;

// }

// resVar=!resVar;

// }

vTaskDelay(1000);

}

}

void SolarPanel(void \*vParameters)

{

static unsigned short motorDrive = 0;

unsigned long period = 125000;

static Bool Flag = FALSE;

static unsigned long dutyCycle = 62500;

static int remainingDeployment = 1000;

while(1)

{

solarPanel \* myPanel = (solarPanel\*) vParameters;

// DEREREFWEJFIOEAWN DEREFERENCE POINTERS

//IntEnable(INT\_GPIOB);

// SysCtlPWMClockSet(SYSCTL\_PWMDIV\_32);

// if (\*(myPanel->solarPanelRetract))

// {

// remainingDeployment = 0;

// }

// else{

// remainingDeployment = 1000;

// 100% motor drive \* 10 seconds

//

// char temp[20];

// sprintf(temp,"%u",\*(myPanel->solarPanelDeploy));

// RIT128x96x4StringDraw(temp, 40, 80, 15);

// sprintf(temp,"%u",\*myPanel->solarPanelRetract);

// RIT128x96x4StringDraw(temp, 60, 80, 15);

// sprintf(temp,"%u",remainingDeployment);

// RIT128x96x4StringDraw(temp, 80, 80, 15);

if(!((\*(myPanel -> solarPanelDeploy) && 0 == remainingDeployment) &&

(\*(myPanel -> solarPanelRetract) && 1000 == remainingDeployment)))

{

if(\*myPanel->solarPanelDeploy || \*myPanel->solarPanelRetract)

{

// HWREG(NVIC\_SW\_TRIG)=INT\_GPIOE-16;

IntDisable(INT\_GPIOG);

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_PWM);

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

GPIOPinTypeGPIOInput(GPIO\_PORTF\_BASE, GPIO\_PIN\_0);

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_0, 0);

GPIOPinTypePWM(GPIO\_PORTF\_BASE, GPIO\_PIN\_0);

GPIOPadConfigSet(GPIO\_PORTF\_BASE, GPIO\_PIN\_0, GPIO\_STRENGTH\_8MA, GPIO\_PIN\_TYPE\_STD);

PWMGenConfigure(PWM0\_BASE, PWM\_GEN\_0, PWM\_GEN\_MODE\_UP\_DOWN | PWM\_GEN\_MODE\_NO\_SYNC);

PWMGenPeriodSet(PWM0\_BASE, PWM\_GEN\_0, period);

PWMPulseWidthSet(PWM0\_BASE, PWM\_OUT\_0, dutyCycle);

PWMGenEnable(PWM0\_BASE, PWM\_GEN\_0);

PWMOutputState(PWM0\_BASE, PWM\_OUT\_0\_BIT, 1);

}

if(\*(myPanel -> driveMotorSpeedInc))

{

if((dutyCycle + (0.05 \* period)) < period) // ensures duty cycle does not exceed 100%

{

dutyCycle += (period \* 0.05); // Increments motor speed by 5%

}

(\*(myPanel -> driveMotorSpeedInc)) = FALSE;

}

else if(\*(myPanel -> driveMotorSpeedDec))

{

int limit = dutyCycle - (0.05 \* period);

if(limit > 0) // ensures duty cycle does not drop below 0%

{

dutyCycle -= (period \* 0.05); // Decrements motor speed by 5%

}

(\*(myPanel -> driveMotorSpeedDec)) = FALSE;

}

motorDrive = dutyCycle \* 100 / period; // (dutyCycle / period) \* 100% / 10

// iterations per second

// solar panels are set up such that

// 10 seconds of 100% motorDrive will

// fully retract or detract a solar panel

// from a full ON or OFF state respectively

if(\*(myPanel -> solarPanelDeploy))

{

if((remainingDeployment - motorDrive) < 0)

{

remainingDeployment = 0;

}

else

{

remainingDeployment -= motorDrive;

}

}

else if(\*(myPanel->solarPanelRetract))

{

if((remainingDeployment + motorDrive) > 1000)

{

remainingDeployment = 1000;

}

else

{

remainingDeployment += motorDrive;

}

}

if(remainingDeployment == 0 && \*(myPanel -> solarPanelDeploy))

{

\*(myPanel -> solarPanelDeploy) = FALSE;

\*(myPanel -> solarPanelState) = TRUE;

g\_ulGPIOb=1;

PWMGenDisable(PWM0\_BASE, PWM\_GEN\_0);

PWMOutputState(PWM0\_BASE, PWM\_OUT\_0\_BIT, 0);

}

else if(remainingDeployment == 1000)

{

if(\*(myPanel -> solarPanelRetract))

{

\*(myPanel -> solarPanelRetract) = FALSE;

\*(myPanel -> solarPanelState) = FALSE;

g\_ulGPIOb=1;

PWMGenDisable(PWM0\_BASE, PWM\_GEN\_0);

PWMOutputState(PWM0\_BASE, PWM\_OUT\_0\_BIT, 0);

}

}

}

// char xyz[20];

// sprintf(xyz,"%u",dutyCycle);

// RIT128x96x4StringDraw(xyz, 20, 80, 15);

SysCtlDelay(600);

IntEnable(INT\_GPIOG);//wait 600 ms before turning on measure interrupts

vTaskDelay(500);

}

}

void keypadManage(void \* data)

{

///Keypad interrupts up

// SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOE);

//

// GPIOPinTypeGPIOInput(GPIO\_PORTE\_BASE, GPIO\_PIN\_2|GPIO\_PIN\_3);

// GPIODirModeSet(GPIO\_PORTE\_BASE, GPIO\_PIN\_2|GPIO\_PIN\_3,GPIO\_DIR\_MODE\_IN );

// GPIOPadConfigSet(GPIO\_PORTE\_BASE, GPIO\_PIN\_2|GPIO\_PIN\_3, GPIO\_STRENGTH\_2MA, GPIO\_PIN\_TYPE\_STD\_WPU);

// GPIOIntTypeSet(GPIO\_PORTE\_BASE, GPIO\_PIN\_2|GPIO\_PIN\_3 , GPIO\_FALLING\_EDGE);

// IntEnable(INT\_GPIOE);

while (1)

{

// RIT128x96x4StringDraw(" KKEYPAD ", 20, 80, 15);

keypad \*myKey = (keypad\*) data;

if (UpFlag){

\*myKey->driveMotorSpeedInc=TRUE; UpFlag=FALSE;

// RIT128x96x4StringDraw(" ", 20, 80, 15);

}

if(DownFlag){

\*myKey->driveMotorSpeedDec=TRUE;DownFlag=FALSE;

// RIT128x96x4StringDraw(" ", 80, 80, 15);

}

// GPIOPinIntClear(GPIO\_PORTE\_BASE, GPIO\_PIN\_3);

IntEnable(INT\_GPIOE);

vTaskDelay(200);

}

}

void WarningManage(void \*vParameters)

{

static short timer =0; // minor cycle is 1/6 a second

static short timer2= 0; // timer 1 for 1 second int timer 2 for 2 sec inv

static short timer3= 0;

static Bool t1 = FALSE;

static Bool t2 = FALSE;

static Bool t3= FALSE;

static Bool t4 = FALSE;

static short Btimer=0,timer4=0;

while(1)

{

//RIT128x96x4StringDraw("warn ", 10, 80, 15);

warning \*mywarning= (warning\*)vParameters;

//PORTC - 4 (PhA0) red, 5 (PC5) yellow, 6 (PhB0), 7(PC7)green

// 0xFF turn on 0x00 turn off

//GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_7, 0xFF);

if (Btimer>500)//if battery temp runs past 500 battery warning lights take precedence

{

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_7, 0x00);// turn off green

if(timer4==0 && t4){

if (t3 && 0==timer3) // if Red on turn it off at timer == 0

{

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_4, 0x00); // red

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_5, 0xFF);// turn ON yellow

t3=!t3;

}

else if ( !t3 && 0 == timer3) { // if its off turn it onn at timer 0

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_4, 0xFF); // red

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_5, 0x00);// turn OFF yellow

t3=!t3;

}

} else if ( !t4 && timer4==0){

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_5, 0x00);// turn OFF yellow

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_4, 0x00); // red

}

goto exit;

}

if ((\*(mywarning->fuelLevel) > 50) && (\*(mywarning->batteryLevel) > 18))

{ //RIT128x96x4StringDraw("Case 0", 00, 75, 15);

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_7, 0xFF); // turn on green

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_5, 0x00); // turn off yellow

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_4, 0x00); // turn off red

goto exit ;

} // green

if ((\*(mywarning->fuelLevel) >50)&&(\*(mywarning->batteryLevel) > 4))

{

// RIT128x96x4StringDraw("Case 1", 00, 75, 15);

//fuel > 50 > battery > 10

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_7, 0x00); // turn off green

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_4, 0x00); // turn off red

if (t1 && 0==timer) // if its on turn it off at timer == 0

{

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_5, 0x00);

t1=!t1;

} else if ( !t1 && 0 == timer) { // if its off turn it onn at timer 0

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_5, 0xFF);

t1=!t1;

}

goto exit;

} // yellow @ 1 sec

//fuel > 50 battery <10

if ((\*(mywarning->fuelLevel) >50)&&(\*(mywarning->batteryLevel) <5))

{

// RIT128x96x4StringDraw("Case 2", 00, 75, 15);

// fuel > 50 || 10 > battery > 0

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_7, 0x00); // turn off green

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_5, 0x00); // turn off yellow

if (t1 && 0==timer) // if its on turn it off at timer == 0

{

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_4, 0x00); // red

t1=!t1;

} else if ( !t1 && 0 == timer) { // if its off turn it onn at timer 0

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_4, 0xFF); // red

t1=!t1;

}

goto exit;

} // red @ 1 sec

// 50>fuel | battery > 50

if ((\*(mywarning->fuelLevel) > 10) && (\*(mywarning->batteryLevel) > 18))

{

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_7, 0x00); // turn off green

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_4, 0x00); // turn off Red

// RIT128x96x4StringDraw("Case 3", 00, 75, 15);

if (t2 && 0==timer2) // if its on turn it off at timer == 0

{

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_5, 0x00); // yellow

t2=!t2;

}

else if ( !t2 && 0 == timer2)

{ // if its off turn it onn at timer 0

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_5, 0xFF); // yellow

t2=!t2;

}

if (t1 && 0==timer) // if its on turn it off at timer == 0

{

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_5, 0x00);

t1=!t1;

}

else if ( !t1 && 0 == timer) { // if its off turn it onn at timer 0

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_5, 0xFF);

t1=!t1;

}

goto exit;

} // yellow @ 2 sec

// 50>fuel |50 > battery > 10

if ((\*(mywarning->fuelLevel) >10)&&(\*(mywarning->batteryLevel) > 4))

{

//RIT128x96x4StringDraw("Case 4", 00, 75, 15);

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_7, 0x00); // turn off green

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_4, 0x00); // turn off red

if (t2 && 0==timer2) // if its on turn it off at timer == 0

{

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_5, 0x00); // yellow

t2=!t2;

}

else if ( !t2 && 0 == timer2)

{ // if its off turn it onn at timer 0

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_5, 0xFF); // yellow

t2=!t2;

}

if (t1 && 0==timer) // if its on turn it off at timer == 0

{

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_5, 0x00);

t1=!t1;

}

else if ( !t1 && 0 == timer) { // if its off turn it onn at timer 0

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_5, 0xFF);

t1=!t1;

}

goto exit;

} // yellow @ 2 sec && yellow @ 1 sec

// 50 > fuel | 10 > battery

if ((\*(mywarning->fuelLevel) >10)&&(\*(mywarning->batteryLevel) <5))

{

//RIT128x96x4StringDraw("Case 5", 00, 75, 15);

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_7, 0x00); // turn off green

if (t1 && 0==timer) // if Red on turn it off at timer == 0

{

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_4, 0x00); // red

t1=!t1;

}

else if ( !t1 && 0 == timer) { // if its off turn it onn at timer 0

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_4, 0xFF); // red

t1=!t1;

}

if (t2 && 0==timer2) // if yellow on turn it off at timer == 0

{

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_5, 0x00); // yellow

t2=!t2;

}

else if ( !t2 && 0 == timer2)

{ // if its off turn it onn at timer 0

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_5, 0xFF); // yellow

t2=!t2;

}

goto exit;

} // yellow @ 2 sec && red @ 1 sec

// 10> fuel | battery>50

if ((\*(mywarning->fuelLevel) < 11) && (\*(mywarning->batteryLevel) > 18))

{

// RIT128x96x4StringDraw("Case 6", 00, 75, 15);

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_7, 0x00); // turn off green

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_5, 0x00); // turn off yellow

if (t2 && 0==timer2) // if Red on turn it off at timer == 0

{

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_4, 0x00); // red

t2=!t2;

}

else if ( !t2 && 0 == timer2) { // if its off turn it onn at timer 0

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_4, 0xFF); // red

t2=!t2;

}

goto exit;

} // red @ 1 sec

// 10>fuel | 50 > battery

if ((\*(mywarning->fuelLevel) <11)&&(\*(mywarning->batteryLevel) > 4))

{

//RIT128x96x4StringDraw("Case 7", 00, 75, 15);

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_7, 0x00);// turn off green

if (t2 && 0==timer2) // if Red on turn it off at timer == 0

{

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_4, 0x00); // red

t2=!t2;

}

else if ( !t2 && 0 == timer2) { // if its off turn it onn at timer 0

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_4, 0xFF); // red

t2=!t2;

}

if (t1 && 0==timer) // if yellow on turn it off at timer == 0

{

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_5, 0x00);

t1=!t1;

}

else if ( !t1 && 0 == timer) { // if yellow off turn it onn at timer 0

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_5, 0xFF);

t1=!t1;

}

goto exit;

} // yellow @ 2 sec && red @ 1 sec

// 10>fuel | 10 > battery

if ((\*(mywarning->fuelLevel) <11)&&(\*(mywarning->batteryLevel) <5))

{

//RIT128x96x4StringDraw("Case 8", 00, 75, 15);

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_7, 0x00);// turn off green

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_5, 0x00);// turn off yellow

if (t2 && 0==timer2) // if Red on turn it off at timer == 0

{

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_4, 0x00); // red

t2=!t2;

}

else if ( !t2 && 0 == timer2) { // if its off turn it onn at timer 0

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_4, 0xFF); // red

t2=!t2;

}

if (t1 && 0==timer) // if Red on turn it off at timer == 0

{

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_4, 0x00); // red

t1=!t1;

}

else if ( !t1 && 0 == timer) { // if its off turn it onn at timer 0

GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_4, 0xFF); // redt

t1=!t1;

}

goto exit;

} // red @ 2 sec && red @ 1 sec

exit :

if (5 == timer)//1sec

{ timer = 0; }

else { timer ++;}

if ( 10 == timer2)//2 sec

{timer2=0;}

else { timer2++;}

if (1==timer3)//10Hz

{

timer3=0;

}else {timer3++;}

if (25==timer4)//5sec

{

timer4=0;

}else{timer4++;}

//Battery

if (\*(mywarning->battOverTemp))

{

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_PWM);

//

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOE);

GPIOPinTypeGPIOInput(GPIO\_PORTE\_BASE, GPIO\_PIN\_1);

//

GPIOPinWrite(GPIO\_PORTE\_BASE, GPIO\_PIN\_1, 0);

//

GPIOPinTypePWM(GPIO\_PORTE\_BASE, GPIO\_PIN\_1);

GPIOPadConfigSet(GPIO\_PORTE\_BASE, GPIO\_PIN\_1, GPIO\_STRENGTH\_8MA, GPIO\_PIN\_TYPE\_STD);

//

PWMGenConfigure(PWM0\_BASE, PWM\_GEN\_2, PWM\_GEN\_MODE\_UP\_DOWN | PWM\_GEN\_MODE\_NO\_SYNC);

//

PWMGenPeriodSet(PWM0\_BASE, PWM\_GEN\_2, 156);

//

PWMPulseWidthSet(PWM0\_BASE, PWM\_OUT\_5, 78);

//

PWMGenEnable(PWM0\_BASE, PWM\_GEN\_2);

PWMOutputState(PWM0\_BASE, PWM\_OUT\_5\_BIT, 1);

Btimer++;

// GPIOPinWrite(GPIO\_PORTC\_BASE, GPIO\_PIN\_2, 0xFF);//

}

if(BTempAck)

{

PWMGenDisable(PWM0\_BASE, PWM\_GEN\_2); Btimer=0;

Btimer=0;

battOverTemp=FALSE;

BTempAck=FALSE;

}

else

{

Btimer++;

}

vTaskDelay(100);

}

}

void ImageCapture(void \*vParameters)

{

unsigned long ulADC\_Value[1];

image \* myImage = (image\*)vParameters;

int index = 0, peakFreqIndex, period = 125000, dutyCycle = 62500;

int m\_index = 1;

volatile double temp;

unsigned long samplingRate;

static int buffIndex = 0;

volatile int i, samplFreq, maxAmp=0;

// SysCtlPeripheralEnable(SYSCTL\_PERIPH\_PWM);

//

// SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOE);

//

// GPIOPinTypeGPIOInput(GPIO\_PORTE\_BASE, GPIO\_PIN\_0);

//

// GPIOPinWrite(GPIO\_PORTE\_BASE, GPIO\_PIN\_0, 0);

//

// GPIOPinTypePWM(GPIO\_PORTE\_BASE, GPIO\_PIN\_0);

//

// GPIOPadConfigSet(GPIO\_PORTE\_BASE, GPIO\_PIN\_0, GPIO\_STRENGTH\_8MA, GPIO\_PIN\_TYPE\_STD);

//

// PWMGenConfigure(PWM0\_BASE, PWM\_GEN\_2, PWM\_GEN\_MODE\_UP\_DOWN | PWM\_GEN\_MODE\_NO\_SYNC);

//

// PWMGenPeriodSet(PWM0\_BASE, PWM\_GEN\_2, period);

//

// PWMPulseWidthSet(PWM0\_BASE, PWM\_OUT\_4, dutyCycle);

//

// PWMGenEnable(PWM0\_BASE, PWM\_GEN\_2);

//

// PWMOutputState(PWM0\_BASE, PWM\_OUT\_4\_BIT, 1);

while (1)

{

while (index < 256)

{

// SysCtlADCSpeedSet(SYSCTL\_ADCSPEED\_500KSPS);

ADCProcessorTrigger(ADC\_BASE, 0);

while (!ADCIntStatus(ADC\_BASE, 0, false)) {}

ADCIntClear(ADC\_BASE, 0);

ADCSequenceDataGet(ADC\_BASE, 0, ulADC\_Value);

// samplingRate = SysCtlADCSpeedGet();

(myImage->imageDataRawPtr)[index] = ulADC\_Value[0] - 0.5; // /(0x3FF)

(myImage->imageDataPtr)[index] = 0;

index++;

//SysCtlDelay(430);

delay(5); // delay 5, freq 1k, index 30, samplFreq 8533, incr 33

// delay 4, index 25, samplFreq 10240, incr 40

// delay 1, index 9, samplFreq 28444, incr 111

// delay 2, index 14, samplFreq 18286, incr 71

// delay 3, index 19, samplFreq 13474, incr 53

// delay = 50, freq = 1k, index = 15, samplFreq = 17066.66, N = 256

// delay = 98, freq = 1k, index = 16, samplFreq = 16000

// delay = 100, freq = 1k, index = 26, sampl freq = 9846

// delay = 200, freq = 1k, index = 6, sampl freq = 42666.66

//delay = 30, freq = 1k, index = 92,

//delay = 30, freq = 2k, index = 72

}

index = 0;

maxAmp = 0;

peakFreqIndex = optfft(myImage->imageDataRawPtr, myImage->imageDataPtr);

// for (i = 0; i < 129; i++)

// {

// temp = sqrt((((myImage->imageDataRawPtr)[i])\*((myImage->imageDataRawPtr)[i]))

// + (((myImage->imageDataPtr)[i])\*((myImage->imageDataPtr)[i])));

// if (temp > maxAmp)

// {

// maxAmp = temp;

// m\_index = i;

// }

// }

// samplFreq = 500000;

// samplFreq = (myImage->imageDataRawPtr)[peakFreqIndex];

samplFreq = 8533; //8533

\*(myImage -> peakFreq) = samplFreq \* peakFreqIndex / 256;

// \*(myImage -> peakFreq) = samplFreq \* m\_index / 256;

// \*(myImage->peakFreq) = samplingRate;

if(buffIndex > 15)

{

buffIndex = 0;

}

finishCapture = TRUE;

vTaskDelay(500);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 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 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//#include "optfft.h"

#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;

/\* 256 bytes \*/

const signed char u1[] = {32, 32, 0, 32, 22, 0, -22, 32, 29, 22, 12, 0, -12, -22, -29, 32, 31, 29, 26, 22, 17, 12, 6, 0, -6, -12, -17, -22, -26, -29, -31, 32, 31, 31, 30, 29, 28, 26, 24, 22, 20, 17, 15, 12, 9, 6, 3, 0, -3, -6, -9, -12, -15, -17, -20, -22, -24, -26, -28, -29, -30, -31, -31, 32, 31, 31, 31, 31, 31, 30, 30, 29, 28, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 17, 16, 15, 13, 12, 10, 9, 7, 6, 4, 3, 1, 0, -1, -3, -4, -6, -7, -9, -10, -12, -13, -15, -16, -17, -19, -20, -21, -22, -23, -24, -25, -26, -27, -28, -28, -29, -30, -30, -31, -31, -31, -31, -31, 32, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 30, 30, 30, 30, 29, 29, 29, 28, 28, 28, 27, 27, 27, 26, 26, 25, 25, 24, 24, 23, 23, 22, 22, 21, 20, 20, 19, 19, 18, 17, 17, 16, 15, 15, 14, 13, 12, 12, 11, 10, 10, 9, 8, 7, 7, 6, 5, 4, 3, 3, 2, 1, 0, 0, 0, -1, -2, -3, -3, -4, -5, -6, -7, -7, -8, -9, -10, -10, -11, -12, -12, -13, -14, -15, -15, -16, -17, -17, -18, -19, -19, -20, -20, -21, -22, -22, -23, -23, -24, -24, -25, -25, -26, -26, -27, -27, -27, -28, -28, -28, -29, -29, -29, -30, -30, -30, -30, -31, -31, -31, -31, -31, -31, -31, -31, -31, -31};

/\* 256 bytes \*/

const signed char u2[] = { 0, 0, -32, 0, -22, -31, -22, 0, -12, -22, -29, -31, -29, -22, -12, 0, -6, -12, -17, -22, -26, -29, -31, -31, -31, -29, -26, -22, -17, -12, -6, 0, -3, -6, -9, -12, -15, -17, -20, -22, -24, -26, -28, -29, -30, -31, -31, -31, -31, -31, -30, -29, -28, -26, -24, -22, -20, -17, -15, -12, -9, -6, -3, 0, -1, -3, -4, -6, -7, -9, -10, -12, -13, -15, -16, -17, -19, -20, -21, -22, -23, -24, -25, -26, -27, -28, -28, -29, -30, -30, -31, -31, -31, -31, -31, -31, -31, -31, -31, -31, -31, -30, -30, -29, -28, -28, -27, -26, -25, -24, -23, -22, -21, -20, -19, -17, -16, -15, -13, -12, -10, -9, -7, -6, -4, -3, -1, 0, 0, -1, -2, -3, -3, -4, -5, -6, -7, -7, -8, -9, -10, -10, -11, -12, -12, -13, -14, -15, -15, -16, -17, -17, -18, -19, -19, -20, -20, -21, -22, -22, -23, -23, -24, -24, -25, -25, -26, -26, -27, -27, -27, -28, -28, -28, -29, -29, -29, -30, -30, -30, -30, -31, -31, -31, -31, -31, -31, -31, -31, -31, -31, -31, -31, -31, -31, -31, -31, -31, -31, -31, -31, -31, -30, -30, -30, -30, -29, -29, -29, -28, -28, -28, -27, -27, -27, -26, -26, -25, -25, -24, -24, -23, -23, -22, -22, -21, -20, -20, -19, -19, -18, -17, -17, -16, -15, -15, -14, -13, -12, -12, -11, -10, -10, -9, -8, -7, -7, -6, -5, -4, -3, -3, -2, -1, 0 };

/\* 512 bytes \*/

const unsigned char square[] = {

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0,

1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 2, 2,

2, 2, 2, 2, 2, 2, 2, 2,

3, 3, 3, 3, 3, 3, 3, 3,

4, 4, 4, 4, 4, 4, 4, 4,

5, 5, 5, 5, 5, 5, 5, 6,

6, 6, 6, 6, 6, 7, 7, 7,

7, 7, 7, 8, 8, 8, 8, 8,

9, 9, 9, 9, 9, 9, 10, 10,

10, 10, 10, 11, 11, 11, 11, 12,

12, 12, 12, 12, 13, 13, 13, 13,

14, 14, 14, 14, 15, 15, 15, 15,

16, 16, 16, 16, 17, 17, 17, 17,

18, 18, 18, 18, 19, 19, 19, 19,

20, 20, 20, 21, 21, 21, 21, 22,

22, 22, 23, 23, 23, 24, 24, 24,

25, 25, 25, 25, 26, 26, 26, 27,

27, 27, 28, 28, 28, 29, 29, 29,

30, 30, 30, 31, 31, 31, 32, 32,

33, 33, 33, 34, 34, 34, 35, 35,

36, 36, 36, 37, 37, 37, 38, 38,

39, 39, 39, 40, 40, 41, 41, 41,

42, 42, 43, 43, 43, 44, 44, 45,

45, 45, 46, 46, 47, 47, 48, 48,

49, 49, 49, 50, 50, 51, 51, 52,

52, 53, 53, 53, 54, 54, 55, 55,

56, 56, 57, 57, 58, 58, 59, 59,

60, 60, 61, 61, 62, 62, 63, 63,

64, 64, 65, 65, 66, 66, 67, 67,

68, 68, 69, 69, 70, 70, 71, 71,

72, 72, 73, 73, 74, 74, 75, 76,

76, 77, 77, 78, 78, 79, 79, 80,

81, 81, 82, 82, 83, 83, 84, 84,

85, 86, 86, 87, 87, 88, 89, 89,

90, 90, 91, 92, 92, 93, 93, 94,

95, 95, 96, 96, 97, 98, 98, 99,

100, 100, 101, 101, 102, 103, 103, 104,

105, 105, 106, 106, 107, 108, 108, 109,

110, 110, 111, 112, 112, 113, 114, 114,

115, 116, 116, 117, 118, 118, 119, 120,

121, 121, 122, 123, 123, 124, 125, 125,

126, 127, 127, 128, 129, 130, 130, 131,

132, 132, 133, 134, 135, 135, 136, 137,

138, 138, 139, 140, 141, 141, 142, 143,

144, 144, 145, 146, 147, 147, 148, 149,

150, 150, 151, 152, 153, 153, 154, 155,

156, 157, 157, 158, 159, 160, 160, 161,

162, 163, 164, 164, 165, 166, 167, 168,

169, 169, 170, 171, 172, 173, 173, 174,

175, 176, 177, 178, 178, 179, 180, 181,

182, 183, 183, 184, 185, 186, 187, 188,

189, 189, 190, 191, 192, 193, 194, 195,

196, 196, 197, 198, 199, 200, 201, 202,

203, 203, 204, 205, 206, 207, 208, 209,

210, 211, 212, 212, 213, 214, 215, 216,

217, 218, 219, 220, 221, 222, 223, 224,

225, 225, 226, 227, 228, 229, 230, 231,

232, 233, 234, 235, 236, 237, 238, 239,

240, 241, 242, 243, 244, 245, 246, 247,

248, 249, 250, 251, 252, 253, 254, 255

};

//#include "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++){

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]))];

l1 = (ABS(real[i]))\*(ABS(real[i])) + (ABS(imag[i]))\*(ABS(imag[i]));

if (l1 > l) {

j = i;

l = l1;

}

}

return (j);

}

void vApplicationStackOverflowHook( xTaskHandle \*pxTask, signed portCHAR \*pcTaskName )

{

( void ) pxTask;

( void ) pcTaskName;

while( 1 );

}

/\*-----------------------------------------------------------\*/

void prvSetupHardware( void )

{

/\*

If running on Rev A2 silicon, turn the LDO voltage up to 2.75V. This is

a workaround to allow the PLL to operate reliably.

\*/

if( DEVICE\_IS\_REVA2 )

{

SysCtlLDOSet( SYSCTL\_LDO\_2\_75V );

}

// Set the clocking to run from the PLL at 50 MHz

SysCtlClockSet( SYSCTL\_SYSDIV\_4 | SYSCTL\_USE\_PLL | SYSCTL\_OSC\_MAIN | SYSCTL\_XTAL\_8MHZ );

// Button interrupt hardware setup

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);//PUSHBUTTON

GPIOPinTypeGPIOInput(GPIO\_PORTF\_BASE, GPIO\_PIN\_1);

GPIODirModeSet(GPIO\_PORTF\_BASE, GPIO\_PIN\_1,GPIO\_DIR\_MODE\_IN );

GPIOPadConfigSet(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, GPIO\_STRENGTH\_2MA, GPIO\_PIN\_TYPE\_STD\_WPU);

GPIOIntTypeSet(GPIO\_PORTF\_BASE, GPIO\_PIN\_1 , GPIO\_FALLING\_EDGE);

GPIOPinIntEnable(GPIO\_PORTF\_BASE, GPIO\_PIN\_1);

IntEnable(INT\_GPIOF);

/// start with the int enabled

SysCtlPWMClockSet(SYSCTL\_PWMDIV\_32);

//Warning LED

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOC); // enable PORTC - 4 (PhA0), 5 (PC5), 6 (PhB0), 7(PC7)

GPIOPinTypeGPIOOutput(GPIO\_PORTC\_BASE, (GPIO\_PIN\_4 | GPIO\_PIN\_5 | GPIO\_PIN\_7));

//Warning Speaker

// GPIOPinTypeGPIOOutput(GPIO\_PORTC\_BASE, GPIO\_PIN\_2);

// SysCtlPeripheralEnable(SYSCTL\_PERIPH\_PWM);

////

// SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOE);

//

// GPIOPinTypeGPIOInput(GPIO\_PORTE\_BASE, GPIO\_PIN\_1);

////

// GPIOPinWrite(GPIO\_PORTE\_BASE, GPIO\_PIN\_1, 0);

////

// GPIOPinTypePWM(GPIO\_PORTE\_BASE, GPIO\_PIN\_1);

//

// GPIOPadConfigSet(GPIO\_PORTE\_BASE, GPIO\_PIN\_1, GPIO\_STRENGTH\_8MA, GPIO\_PIN\_TYPE\_STD);

////

// PWMGenConfigure(PWM0\_BASE, PWM\_GEN\_2, PWM\_GEN\_MODE\_UP\_DOWN | PWM\_GEN\_MODE\_NO\_SYNC);

////

// PWMGenPeriodSet(PWM0\_BASE, PWM\_GEN\_2, 50);

////

// PWMPulseWidthSet(PWM0\_BASE, PWM\_OUT\_2, 25);

////

//Battery Connect

//solar panel triggers interrupt before moving

// GPIOPinTypeGPIOInput(GPIO\_PORTC\_BASE, GPIO\_PIN\_6);

// GPIODirModeSet(GPIO\_PORTC\_BASE, GPIO\_PIN\_6,GPIO\_DIR\_MODE\_IN );

// GPIOPadConfigSet(GPIO\_PORTC\_BASE, GPIO\_PIN\_6, GPIO\_STRENGTH\_2MA, GPIO\_PIN\_TYPE\_STD\_WPU);

// GPIOIntTypeSet(GPIO\_PORTC\_BASE, GPIO\_PIN\_6 , GPIO\_FALLING\_EDGE);

// GPIOPinIntEnable(GPIO\_PORTC\_BASE, GPIO\_PIN\_6);

IntEnable(INT\_GPIOG);

//Pirate

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOD);

GPIOPinTypeGPIOInput(GPIO\_PORTD\_BASE,GPIO\_PIN\_2);

GPIOPinTypeGPIOOutput(GPIO\_PORTD\_BASE,GPIO\_PIN\_5);

GPIODirModeSet(GPIO\_PORTD\_BASE, GPIO\_PIN\_2 ,GPIO\_DIR\_MODE\_IN );

GPIODirModeSet(GPIO\_PORTD\_BASE, GPIO\_PIN\_5 ,GPIO\_DIR\_MODE\_OUT );

GPIOPadConfigSet(GPIO\_PORTD\_BASE, GPIO\_PIN\_2 |GPIO\_PIN\_5, GPIO\_STRENGTH\_8MA, GPIO\_PIN\_TYPE\_STD\_WPU);

//GPIOIntTypeSet(GPIO\_PORTD\_BASE, GPIO\_PIN\_2, GPIO\_HIGH\_LEVEL);

// GPIOPinIntEnable(GPIO\_PORTD\_BASE, GPIO\_PIN\_2 );

// IntEnable(INT\_GPIOD);

//distance sensor

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOC);

GPIOPinTypeGPIOInput(GPIO\_PORTC\_BASE, GPIO\_PIN\_6);

GPIODirModeSet(GPIO\_PORTC\_BASE, GPIO\_PIN\_6,GPIO\_DIR\_MODE\_IN );

GPIOPadConfigSet(GPIO\_PORTC\_BASE, GPIO\_PIN\_6, GPIO\_STRENGTH\_2MA, GPIO\_PIN\_TYPE\_STD\_WPU);

GPIOIntTypeSet(GPIO\_PORTC\_BASE, GPIO\_PIN\_6 , GPIO\_RISING\_EDGE);

GPIOPinIntEnable(GPIO\_PORTC\_BASE, GPIO\_PIN\_6);

IntEnable(INT\_GPIOC);

//Timer

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_TIMER1);

IntMasterEnable();

TimerConfigure(TIMER1\_BASE, TIMER\_CFG\_PERIODIC);

TimerLoadSet(TIMER1\_BASE, TIMER\_A, SysCtlClockGet());

IntEnable(INT\_TIMER1A);

TimerIntEnable(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT/100);

TimerEnable(TIMER1\_BASE, TIMER\_A);

//ADC STUFF

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_ADC);

ADCSequenceStepConfigure(ADC\_BASE, 3, 0, ADC\_CTL\_CH0 | ADC\_CTL\_IE |

ADC\_CTL\_END);

ADCSequenceEnable(ADC\_BASE, 3);

ADCIntClear(ADC\_BASE, 3);

ADCIntEnable(ADC\_BASE, 0);

ADCSequenceStepConfigure(ADC\_BASE, 2, 0, ADC\_CTL\_CH1 | ADC\_CTL\_IE |

ADC\_CTL\_END);

ADCSequenceEnable(ADC\_BASE, 2);

ADCIntClear(ADC\_BASE, 2);

ADCIntEnable(ADC\_BASE, 2);

ADCSequenceStepConfigure(ADC\_BASE, 1, 0, ADC\_CTL\_CH2 | ADC\_CTL\_IE |

ADC\_CTL\_END);

ADCSequenceEnable(ADC\_BASE, 1);

ADCIntClear(ADC\_BASE, 1);

ADCIntEnable(ADC\_BASE, 1);

ADCSequenceStepConfigure(ADC\_BASE, 0, 0, ADC\_CTL\_CH3 | ADC\_CTL\_IE |

ADC\_CTL\_END);

// SysCtlADCSpeedSet(SYSCTL\_ADCSPEED\_500KSPS);

ADCSequenceEnable(ADC\_BASE, 0);

ADCIntClear(ADC\_BASE, 0);

ADCIntEnable(ADC\_BASE, 0);

// ///Keypad interrupts up

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOE);

GPIOPinTypeGPIOInput(GPIO\_PORTE\_BASE, GPIO\_PIN\_2|GPIO\_PIN\_3);

GPIODirModeSet(GPIO\_PORTE\_BASE, GPIO\_PIN\_2|GPIO\_PIN\_3,GPIO\_DIR\_MODE\_IN );

GPIOPadConfigSet(GPIO\_PORTE\_BASE, GPIO\_PIN\_2|GPIO\_PIN\_3, GPIO\_STRENGTH\_2MA, GPIO\_PIN\_TYPE\_STD\_WPU);

GPIOIntTypeSet(GPIO\_PORTE\_BASE, GPIO\_PIN\_2|GPIO\_PIN\_3 , GPIO\_LOW\_LEVEL);

GPIOPinIntEnable(GPIO\_PORTE\_BASE, GPIO\_PIN\_2|GPIO\_PIN\_3);

IntEnable(INT\_GPIOE);

//// UART

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_UART0);

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOA);

/// Set GPIO A0 and A1 as UART pins.

GPIOPinTypeUART(GPIO\_PORTA\_BASE, GPIO\_PIN\_0 | GPIO\_PIN\_1);

/// Configure the UART for 115,200, 8-N-1 operation.

UARTConfigSetExpClk(UART0\_BASE, SysCtlClockGet(), 115200,

(UART\_CONFIG\_WLEN\_8 | UART\_CONFIG\_STOP\_ONE |

UART\_CONFIG\_PAR\_NONE));

/// Enable the UART interrupt.

IntEnable(INT\_UART0);

UARTIntEnable(UART0\_BASE, UART\_INT\_RX | UART\_INT\_RT);

}

void PirateManage(void \* Pirates)

{

Bool \* Pir=(Bool\*)Pirates;

if (GPIOPinRead(GPIO\_PORTD\_BASE,GPIO\_PIN\_2)!=0XFF)

{

GPIOIntTypeSet(GPIO\_PORTD\_BASE, GPIO\_PIN\_2 , GPIO\_LOW\_LEVEL);

GPIOPinWrite(GPIO\_PORTD\_BASE, GPIO\_PIN\_3, 0xFF);//ENABLE MANAGE TASK

GPIOPinIntEnable(GPIO\_PORTD\_BASE, GPIO\_PIN\_2);

IntEnable(INT\_GPIOD);

}else if (GPIOPinRead(GPIO\_PORTD\_BASE,GPIO\_PIN\_2)==0X00){

GPIOPinWrite(GPIO\_PORTD\_BASE, GPIO\_PIN\_3, 0xFF);//ENABLE MANAGE TASK

GPIOIntTypeSet(GPIO\_PORTD\_BASE, GPIO\_PIN\_2 , GPIO\_HIGH\_LEVEL);

GPIOPinIntEnable(GPIO\_PORTD\_BASE, GPIO\_PIN\_2);

IntEnable(INT\_GPIOD);

}

}

void Pirate(void)

{

GPIOPinIntClear(GPIO\_PORTD\_BASE, GPIO\_PIN\_2);

Pirates=!Pirates;

IntDisable(INT\_GPIOD);

}

/\*-----------------------------------------------------------\*/

void INTDistance(void)

{

GPIOPinIntClear(GPIO\_PORTC\_BASE, GPIO\_PIN\_6);

if(!distFLAG){

distFLAG=TRUE;

}

}

void Distance(void\* Data)

{

vehComs \*data=(vehComs\*)Data;

static int s=0,f=0;

StopTimer=FALSE;

StartTimer=TRUE;

static unsigned long distbuff[]={0,0,0,0,0,0,0,0};

const int MaxPeriod=100;

while(1)

{

if (distFLAG){

if(StartTimer && !StopTimer){

StopTimer=FALSE;

StartTimer=FALSE;

}else if(StopTimer && !StartTimer)

{

StopTimer=FALSE;

StartTimer=FALSE;

}

distFLAG=FALSE;

}

if(!StartTimer && s==0 && !StopTimer)

{

s=globalCounter;

StopTimer=TRUE;

StartTimer=FALSE;

}else if(!StartTimer && !StopTimer && s!=0)

{

if ((globalCounter-s)>MaxPeriod)

{

distbuff[f]=2000;

}else if((globalCounter-s)==6)

{

distbuff[f]=0;

}else{

distbuff[f]=((globalCounter-s))\*20;}

// if((distance[f]-distance[f%7])>10)//if current dist is greater than 10% of last distance

// {

\*(data->dist)=distbuff[f];

// }

f=0;

StopTimer=FALSE;

StartTimer=TRUE;

s=0;

f++;

}

if (7==f){f=0;}

vTaskDelay(50);

}

}

void vApplicationTickHook( void )

{

static xOLEDMessage xMessage = { "PASS" };

static unsigned portLONG ulTicksSinceLastDisplay = 0;

portBASE\_TYPE xHigherPriorityTaskWoken = pdFALSE;

/\*

Called from every tick interrupt. Have enough ticks passed to make it

time to perform our health status check again?

\*/

ulTicksSinceLastDisplay++;

if( ulTicksSinceLastDisplay >= mainCHECK\_DELAY )

{

ulTicksSinceLastDisplay = 0;

}

}

void InitData( void )

{

// initialize each structure

myPower.solarPanelState=&solarPanelState;

myPower.solarPanelDeploy=&solarPanelDeploy;

myPower.solarPanelRetract=&solarPanelRetract;

myPower.batteryLevelPtr=batteryLevelPtr;

myPower.fuelLevel=&fuelLevel;

myPower.powerConsumption=&powerConsumption;

myPower.powerGeneration=&powerGeneration;

myPower.batteryLow=&batteryLow;

myPower.batteryLevel=&batteryLevel;

myPower.battTemp = &battTemp;

myPower.secondTemp = &secondTemp;

myPower.battOverTemp = &battOverTemp;

myPower.battTempPtr=battTempPtr;

myCommand.cType=&Ctype;

myCommand.Payload=payload;

myCommand.thrusterCommand=&thrusterCommand;

myCommand.TransmitFlag=&TFLAG;

myCommand.batteryLevel=&batteryLevel;

myCommand.fuelLevel=&fuelLevel;

myCommand.Response=Response;

mySolarPanel.driveMotorSpeedDec=&driveMotorSpeedDec;

mySolarPanel.driveMotorSpeedInc=&driveMotorSpeedInc;

mySolarPanel.solarPanelDeploy=&solarPanelDeploy;

mySolarPanel.solarPanelRetract=&solarPanelRetract;

mySolarPanel.solarPanelState=&solarPanelState;

myKeypad.driveMotorSpeedDec=&driveMotorSpeedDec;

myKeypad.driveMotorSpeedInc=&driveMotorSpeedInc;

myThruster.fuelLevel=&fuelLevel;

myThruster.thrusterCommand=&thrusterCommand;

myThruster.fuelLow=&fuelLow;

myComs.batteryLevel=&batteryLevel;

myComs.batteryLow=&batteryLow;

myComs.fuelLevel=&fuelLevel;

myComs.fuelLow=&fuelLow;

myComs.powerConsumption=&powerConsumption;

myComs.powerGeneration=&powerGeneration;

myComs.solarPanelState=&solarPanelState;

myComs.thrusterCommand=&thrusterCommand;

myComs.response=&response;

myComs.battTemp=&battTemp;

myComs.secondTemp=&secondTemp;

myComs.dist=&distance;

myComs.peakFreq=&peakFreq;

myComs.cType=&Ctype;

myComs.Payload=payload;

myComs.Response=Response;

myComs.TransmitFlag=&TFLAG;

myVehComs.command=&command;

myVehComs.response=&response;

myVehComs.dist=&distance;

myDisplay.batteryLevel=&batteryLevel;

myDisplay.batteryLow=&batteryLow;

myDisplay.fuelLevel=&fuelLevel;

myDisplay.fuelLow=&fuelLow;

myDisplay.powerConsumption=&powerConsumption;

myDisplay.powerGeneration=&powerGeneration;

myDisplay.solarPanelState=&solarPanelState;

myDisplay.VehDist=&distance;

//myDisplay.TempKelvin=&BattTempKelvin;

myDisplay.battTemp = &battTemp;

myDisplay.secondTemp = &secondTemp;

myDisplay.battOverTemp = &battOverTemp;

myDisplay.peakFreq=&peakFreq;

myWarning.batteryLevel=&batteryLevel;

myWarning.batteryLow=&batteryLow;

myWarning.fuelLevel=&fuelLevel;

myWarning.fuelLow=&fuelLow;

// myWarning.BattOverTemp=&BattOTemp;

myWarning.battOverTemp = &battOverTemp;

myImage.imageDataRawPtr = imageDataRawPtr;

myImage.imageDataPtr = imageDataPtr;

myImage.peakFreq = &peakFreq;

myImage.imageDataPending=imageDataPending;

//GPIOPortIntRegister(GPIO\_PORTF\_BASE, &IntGPIOa);

}

void measureINT(void)

{

if (!Measure){

// RIT128x96x4Clear();

Measure=TRUE;}

GPIOPinIntClear(GPIO\_PORTC\_BASE, GPIO\_PIN\_6);

//RIT128x96x4StringDraw("measure int ", 10, 80, 15);

}

void Timer1IntHandler(void)

{

//

// Clear the timer interrupt.

//

TimerIntClear(TIMER1\_BASE, TIMER\_TIMA\_TIMEOUT);

//RIT128x96x4StringDraw("TIMER ", 10, 80, 15);

globalCounter++;

//

// Update the interrupt status on the display.

//

//IntMasterDisable();

//IntMasterEnable();

}

void ThrustUpKey(void)

{

// char x[20];

// sprintf(x,"%u",GPIOPinRead(GPIO\_PORTE\_BASE, GPIO\_PIN\_2));

// RIT128x96x4StringDraw(x, 40, 80, 15);

if(0==GPIOPinRead(GPIO\_PORTE\_BASE, GPIO\_PIN\_2))

{

// RIT128x96x4StringDraw(" UP ", 20, 80, 15);

if(!UpFlag){UpFlag=TRUE;}

GPIOPinIntClear(GPIO\_PORTE\_BASE, GPIO\_PIN\_2);

}

// GPIOPinIntClear(GPIO\_PORTE\_BASE, GPIO\_PIN\_2);

//delay(4000);

if (0==GPIOPinRead(GPIO\_PORTE\_BASE, GPIO\_PIN\_3))

{

// RIT128x96x4StringDraw(" DOWN ", 80, 80, 15);

if(!DownFlag){ DownFlag=TRUE; }

GPIOPinIntClear(GPIO\_PORTE\_BASE, GPIO\_PIN\_3);

}

IntDisable(INT\_GPIOE);

}

void IntGPIOa(void)

{

if (g\_ulFlags)

{mode=!mode; g\_ulFlags=FALSE; BTempAck=TRUE; }

GPIOPinIntClear(GPIO\_PORTF\_BASE, GPIO\_PIN\_1);

}

void UARTSend(const unsigned char \*pucBuffer, unsigned long ulCount)

{

// Loop while there are more characters to send.

while(ulCount--)

{

// Write the next character to the UART.

UARTCharPutNonBlocking(UART0\_BASE, \*pucBuffer++);

}

}

void UARTIntHandler(void){

// Clear the asserted interrupts.

UARTIntClear(UART0\_BASE, UARTIntStatus(UART0\_BASE, true));

//\*(myVehComs.response)=\*(myVehComs.command);

if(UARTCharsAvail(UART0\_BASE))

{

\*(myVehComs.command)=UARTCharGetNonBlocking(UART0\_BASE);

//\*(myVehComs.response)=\*(myVehComs.command);

switch(\*(myVehComs.command)){

case 'T':

UARTCharPutNonBlocking(UART0\_BASE, 'K');

break;

case 'D':

UARTCharPutNonBlocking(UART0\_BASE, 'C');

break;

default:

\*(myVehComs.response)='N';

UARTCharPutNonBlocking(UART0\_BASE, 'N');

break;

}

}

}

void delay(unsigned long aValue)

{

volatile unsigned long i = 0;

volatile unsigned int j = 0;

for (i = aValue; i > 0; i--)

{

for (j = 0; j < 100; j++);

}

return;

}

/\*

Thomas Wei, Jitesh Bhogal, Xinyu Sui

startup\_ewarm.c

\*/

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// startup\_ewarm.c - Boot code for Stellaris.

//

// Copyright (c) 2006-2007 Luminary Micro, Inc. All rights reserved.

//

// This is part of revision 100 of the Stellaris Ethernet

// Applications Library.

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// Enable the IAR extensions for this source file.

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#pragma language=extended

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// Forward declaration of the default fault handlers.

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

static void NmiSR(void);

static void FaultISR(void);

static void IntDefaultHandler(void);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// External declaration for the interrupt handler used by the application.

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// The entry point for the application.

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

extern void \_\_iar\_program\_start(void);

extern void xPortPendSVHandler(void);

extern void xPortSysTickHandler(void);

extern void vPortSVCHandler(void);

extern void vT2InterruptHandler( void );

extern void vT3InterruptHandler( void );

extern void vEMAC\_ISR( void );

extern void Timer0IntHandler( void );

extern void UARTIntHandler( void );

extern void IntGPIOa(void);

extern void measureINT(void);

extern void Timer1IntHandler(void);

extern void ThrustDWNKey(void);

extern void ThrustUpKey(void);

extern void INTDistance(void);

extern void Pirate(void);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// Reserve space for the system stack.

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#ifndef STACK\_SIZE

#define STACK\_SIZE 256

#endif

static unsigned long pulStack[STACK\_SIZE] @ ".noinit";

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// A union that describes the entries of the vector table. The union is needed

// since the first entry is the stack pointer and the remainder are function

// pointers.

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

typedef union

{

void (\*pfnHandler)(void);

unsigned long ulPtr;

}

uVectorEntry;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// The minimal vector table for a Cortex-M3. Note that the proper constructs

// must be placed on this to ensure that it ends up at physical address

// 0x0000.0000.

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\_\_root const uVectorEntry \_\_vector\_table[] @ ".intvec" =

{

{ .ulPtr = (unsigned long)pulStack + sizeof(pulStack) },

// The initial stack pointer

\_\_iar\_program\_start, // The reset handler

NmiSR, // The NMI handler

FaultISR, // The hard fault handler

IntDefaultHandler, // The MPU fault handler

IntDefaultHandler, // The bus fault handler

IntDefaultHandler, // The usage fault handler

0, // Reserved

0, // Reserved

0, // Reserved

0, // Reserved

vPortSVCHandler, // SVCall handler

IntDefaultHandler, // Debug monitor handler

0, // Reserved

xPortPendSVHandler, // The PendSV handler

xPortSysTickHandler, // The SysTick handler

IntDefaultHandler, // GPIO Port A

IntDefaultHandler, // GPIO Port B

INTDistance, // GPIO Port C

Pirate, // GPIO Port D

ThrustUpKey, // GPIO Port E

UARTIntHandler, // UART0 Rx and Tx

IntDefaultHandler, // UART1 Rx and Tx

IntDefaultHandler, // SSI Rx and Tx

IntDefaultHandler, // I2C Master and Slave

IntDefaultHandler, // PWM Fault

IntDefaultHandler, // PWM Generator 0

IntDefaultHandler, // PWM Generator 1

IntDefaultHandler, // PWM Generator 2

IntDefaultHandler, // Quadrature Encoder

IntDefaultHandler, // ADC Sequence 0

IntDefaultHandler, // ADC Sequence 1

IntDefaultHandler, // ADC Sequence 2

IntDefaultHandler, // ADC Sequence 3

IntDefaultHandler, // Watchdog timer

Timer0IntHandler, // Timer 0 subtimer A

IntDefaultHandler, // Timer 0 subtimer B

Timer1IntHandler, // Timer 1 subtimer A

IntDefaultHandler, // Timer 1 subtimer B

vT2InterruptHandler, // Timer 2 subtimer A

IntDefaultHandler, // Timer 2 subtimer B

IntDefaultHandler, // Analog Comparator 0

IntDefaultHandler, // Analog Comparator 1

IntDefaultHandler, // Analog Comparator 2

IntDefaultHandler, // System Control (PLL, OSC, BO)

IntDefaultHandler, // FLASH Control

IntGPIOa, // GPIO Port F

measureINT, // GPIO Port G

IntDefaultHandler, // GPIO Port H

IntDefaultHandler, // UART2 Rx and Tx

IntDefaultHandler, // SSI1 Rx and Tx

vT3InterruptHandler, // Timer 3 subtimer A

IntDefaultHandler, // Timer 3 subtimer B

IntDefaultHandler, // I2C1 Master and Slave

IntDefaultHandler, // Quadrature Encoder 1

IntDefaultHandler, // CAN0

IntDefaultHandler, // CAN1

IntDefaultHandler, // CAN2

vEMAC\_ISR, // Ethernet

IntDefaultHandler, // Hibernate

IntDefaultHandler, // USB0

IntDefaultHandler, // PWM Generator 3

IntDefaultHandler, // uDMA Software Transfer

IntDefaultHandler // uDMA Error

};

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// This is the code that gets called when the processor receives a NMI. This

// simply enters an infinite loop, preserving the system state for examination

// by a debugger.

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

static void

NmiSR(void)

{

//

// Enter an infinite loop.

//

while(1)

{

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// This is the code that gets called when the processor receives a fault

// interrupt. This simply enters an infinite loop, preserving the system state

// for examination by a debugger.

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

static void

FaultISR(void)

{

//

// Enter an infinite loop.

//

while(1)

{

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// This is the code that gets called when the processor receives an unexpected

// interrupt. This simply enters an infinite loop, preserving the system state

// for examination by a debugger.

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

static void

IntDefaultHandler(void)

{

//

// Go into an infinite loop.

//

while(1)

{

}

}

/\*

Xinyu Sui

EE 472 Final project header

\*/

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// The error routine that is called if the driver library encounters an error.

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#ifdef DEBUG

void

\_\_error\_\_(char \*pcFilename, unsigned long ulLine)

{

}

#endif

#define LEFT 0

#define RIGHT 1

#define UP 2

#define DOWN 3

#define COMMAX 65535

#define NULL 0

#include "C:\StellarisWare\inc\hw\_types.h"

#include "C:\StellarisWare\driverlib\debug.h"

#include "C:\StellarisWare\driverlib\sysctl.h"

#include "C:\StellarisWare\boards\ek-lm3s8962\drivers\rit128x96x4.h"

#include "C:\StellarisWare\inc\hw\_ints.h"

#include "C:\StellarisWare\inc\hw\_memmap.h"

#include "C:\StellarisWare\inc\hw\_nvic.h"

#include "C:\StellarisWare\driverlib\gpio.h"

#include "C:\StellarisWare\driverlib\interrupt.h"

#include "C:\StellarisWare\driverlib\systick.h"

#include "C:\StellarisWare\driverlib\timer.h"

#include "C:\StellarisWare\driverlib\pwm.h"

#include "C:\StellarisWare\driverlib\adc.h"

#include "C:\StellarisWare\driverlib\uart.h"

#include "FreeRTOS.h"

#include "task.h"

#include "queue.h"

#include "semphr.h"

extern volatile unsigned long g\_ulGPIOb;

typedef enum{FALSE, TRUE} Bool;

extern int globalCounter;

extern Bool mode, addSolarPanel,Measure, startCapture, finishCapture;

typedef struct{

int \*real;

int \*complex;

} complex;

typedef struct mystructTCB{

void (\*myTask)(void\*);

void \*taskDataPtr;

struct mystructTCB \*next;

struct mystructTCB \*prev;

} TCB;

typedef struct{

Bool \*solarPanelState;

Bool \*solarPanelDeploy;

Bool \*solarPanelRetract;

Bool \*batteryLow;

unsigned short \*batteryLevel;

unsigned int \*batteryLevelPtr;

unsigned short \*powerConsumption;

unsigned short \*powerGeneration;

unsigned short \*fuelLevel;

unsigned short \*battTemp;

unsigned short \*secondTemp;

Bool \*battOverTemp;

unsigned int \*battTempPtr;

} power;

typedef struct{

Bool \*solarPanelState;

Bool \*solarPanelDeploy;

Bool \*solarPanelRetract;

Bool \*driveMotorSpeedInc;

Bool \*driveMotorSpeedDec;

} solarPanel;

typedef struct{

Bool \*driveMotorSpeedInc;

Bool \*driveMotorSpeedDec;

} keypad;

typedef struct{

unsigned int \*thrusterCommand;

unsigned short \*fuelLevel;

Bool \*fuelLow;

} thruster;

typedef struct{

Bool \*fuelLow;

Bool \*batteryLow;

Bool \*solarPanelState;

Bool \*TransmitFlag;

unsigned short \*batteryLevel;

unsigned short \*fuelLevel;

unsigned short \*powerConsumption;

unsigned short \*powerGeneration;

unsigned int \*thrusterCommand;

unsigned short \*battTemp;

unsigned short \*secondTemp;

char \*response;

volatile unsigned long \*dist;

unsigned short \*peakFreq;

char \*cType;

char \*Payload;

char \*Response;

} coms;

typedef struct{

char \*command;

char \*response;

volatile unsigned long \*dist;

} vehComs;

typedef struct{

Bool \*fuelLow;

Bool \*batteryLow;

Bool \*solarPanelState;

unsigned short \*batteryLevel;

unsigned short \*fuelLevel;

unsigned short \*powerConsumption;

unsigned short \*powerGeneration;

volatile unsigned long \*VehDist;

unsigned short \*TempKelvin;

unsigned short \*battTemp;

unsigned short \*secondTemp;

Bool \*battOverTemp;

unsigned short \*peakFreq;

} display;

typedef struct{

Bool \*fuelLow;

Bool \*batteryLow;

unsigned short \*batteryLevel;

unsigned short \*fuelLevel;

Bool \*battOverTemp;

} warning;

typedef struct{

signed int \*imageDataRawPtr;

signed int \*imageDataPtr;

unsigned short \*peakFreq;

signed int \*imageDataPending;

} image;

typedef struct{

char \*cType;

char \*Payload;

char \*Response;

unsigned int \*thrusterCommand;

Bool \*TransmitFlag;

unsigned short \*batteryLevel;

unsigned short \*fuelLevel;

}Command;

extern xTaskHandle commHandle;

extern coms myComs;

extern char \*chav, \*pcText;

void delay(unsigned long aValue);

void schedule(TCB\*\* head, TCB\*\* tail, TCB\* solarPanelSys, TCB\* keypadSys);

void thrusterManage(void\* myThruster);

void powerManage(void\* myPower);

void displayManage(void\* myDisplay);

void comsManage(void\* myComs);

void warningManage(void\* myWarning);

void vehComsManage(void\* myVehComsManage);

void keypadManage(void\* myKeypad);

void solarPanelManage(void\* mySolarPanel);

int randomInteger(int low, int high);

// interrupt routine for push button

void IntGPIOa(void);

void ThrustUpKey(void);

// TIMER

void Timer1IntHandler(void);

void UARTIntHandler(void);

void solarpanelINT(void);

void measureINT(void);

void Distance(void\* Data);

void CommandManage(void\*mydata);

extern void ENET(void);

/\*

Xinyu Sui

ulP\_Task.c

\*/

/\*

FreeRTOS V7.0.1 - Copyright (C) 2011 Real Time Engineers Ltd.

\*/

/\* Standard includes. \*/

#include <string.h>

/\* Scheduler includes. \*/

#include "FreeRTOS.h"

#include "task.h"

#include "semphr.h"

/\* uip includes. \*/

#include "hw\_types.h"

#include "uip.h"

#include "uip\_arp.h"

#include "httpd.h"

#include "timer.h"

#include "clock-arch.h"

#include "hw\_ethernet.h"

#include "ethernet.h"

#include "hw\_memmap.h"

#include "lmi\_flash.h"

#include "sysctl.h"

/\* Demo includes. \*/

#include "emac.h"

#include "partest.h"

#include "lcd\_message.h"

#include "myHeader.h"

struct timer {

clock\_time\_t start;

clock\_time\_t interval;

};

/\*-----------------------------------------------------------\*/

/\* IP address configuration. \*/

/\*

#define uipIP\_ADDR0 172

#define uipIP\_ADDR1 25

#define uipIP\_ADDR2 218

#define uipIP\_ADDR3 19

\*/

#define uipIP\_ADDR0 128

#define uipIP\_ADDR1 95

#define uipIP\_ADDR2 141

#define uipIP\_ADDR3 195

/\* How long to wait before attempting to connect the MAC again. \*/

#define uipINIT\_WAIT 100

/\* Shortcut to the header within the Rx buffer. \*/

#define xHeader ((struct uip\_eth\_hdr \*) &uip\_buf[ 0 ])

/\* Standard constant. \*/

#define uipTOTAL\_FRAME\_HEADER\_SIZE 54

/\*-----------------------------------------------------------\*/

/\*

\* Send the uIP buffer to the MAC.

\*/

static void prvENET\_Send(void);

/\*

\* Setup the MAC address in the MAC itself, and in the uIP stack.

\*/

static void prvSetMACAddress( void );

/\*

\* Port functions required by the uIP stack.

\*/

void clock\_init( void );

clock\_time\_t clock\_time( void );

/\*-----------------------------------------------------------\*/

/\* The semaphore used by the ISR to wake the uIP task. \*/

extern xSemaphoreHandle xEMACSemaphore;

/\*-----------------------------------------------------------\*/

void clock\_init(void)

{

/\* This is done when the scheduler starts. \*/

}

/\*-----------------------------------------------------------\*/

/\* Define clock functions here to avoid header file name clash between uIP

and the Luminary Micro driver library. \*/

clock\_time\_t clock\_time( void )

{

return xTaskGetTickCount();

}

extern void timer\_set(struct timer \*t, clock\_time\_t interval);

extern int timer\_expired(struct timer \*t);

extern void timer\_reset(struct timer \*t);

void ComsManage( void \*pvParameters )

{

portBASE\_TYPE i;

uip\_ipaddr\_t xIPAddr;

struct timer periodic\_timer, arp\_timer;

extern void ( vEMAC\_ISR )( void );

/\* Enable/Reset the Ethernet Controller \*/

SysCtlPeripheralEnable( SYSCTL\_PERIPH\_ETH );

SysCtlPeripheralReset( SYSCTL\_PERIPH\_ETH );

/\* Create the semaphore used by the ISR to wake this task. \*/

vSemaphoreCreateBinary( xEMACSemaphore );

/\* Initialise the uIP stack. \*/

timer\_set( &periodic\_timer, configTICK\_RATE\_HZ / 2 );

timer\_set( &arp\_timer, configTICK\_RATE\_HZ \* 10 );

uip\_init();

uip\_ipaddr( xIPAddr, uipIP\_ADDR0, uipIP\_ADDR1, uipIP\_ADDR2, uipIP\_ADDR3 );

uip\_sethostaddr( xIPAddr );

httpd\_init();

while( vInitEMAC() != pdPASS )

{

vTaskDelay( uipINIT\_WAIT );

}

prvSetMACAddress();

for( ;; )

{

/\* Is there received data ready to be processed? \*/

uip\_len = uiGetEMACRxData( uip\_buf );

if( uip\_len > 0 )

{

/\* Standard uIP loop taken from the uIP manual. \*/

if( xHeader->type == htons( UIP\_ETHTYPE\_IP ) )

{

uip\_arp\_ipin();

uip\_input();

/\* If the above function invocation resulted in data that

should be sent out on the network, the global variable

uip\_len is set to a value > 0. \*/

if( uip\_len > 0 )

{

uip\_arp\_out();

prvENET\_Send();

}

}

else if( xHeader->type == htons( UIP\_ETHTYPE\_ARP ) )

{

uip\_arp\_arpin();

/\* If the above function invocation resulted in data that

should be sent out on the network, the global variable

uip\_len is set to a value > 0. \*/

if( uip\_len > 0 )

{

prvENET\_Send();

}

}

}

else

{

if( timer\_expired( &periodic\_timer ) )

{

timer\_reset( &periodic\_timer );

for( i = 0; i < UIP\_CONNS; i++ )

{

uip\_periodic( i );

/\* If the above function invocation resulted in data that

should be sent out on the network, the global variable

uip\_len is set to a value > 0. \*/

if( uip\_len > 0 )

{

uip\_arp\_out();

prvENET\_Send();

}

}

/\* Call the ARP timer function every 10 seconds. \*/

if( timer\_expired( &arp\_timer ) )

{

timer\_reset( &arp\_timer );

uip\_arp\_timer();

}

}

else

{

/\* We did not receive a packet, and there was no periodic

processing to perform. Block for a fixed period. If a packet

is received during this period we will be woken by the ISR

giving us the Semaphore. \*/

xSemaphoreTake( xEMACSemaphore, configTICK\_RATE\_HZ / 2 );

}

}

}

}

/\*-----------------------------------------------------------\*/

static void prvENET\_Send(void)

{

vInitialiseSend();

vIncrementTxLength( uip\_len );

vSendBufferToMAC();

}

/\*-----------------------------------------------------------\*/

static void prvSetMACAddress( void )

{

unsigned portLONG ulUser0, ulUser1;

unsigned char pucMACArray[8];

struct uip\_eth\_addr xAddr;

/\* Get the device MAC address from flash \*/

FlashUserGet(&ulUser0, &ulUser1);

/\* Convert the MAC address from flash into sequence of bytes. \*/

pucMACArray[0] = ((ulUser0 >> 0) & 0xff);

pucMACArray[1] = ((ulUser0 >> 8) & 0xff);

pucMACArray[2] = ((ulUser0 >> 16) & 0xff);

pucMACArray[3] = ((ulUser1 >> 0) & 0xff);

pucMACArray[4] = ((ulUser1 >> 8) & 0xff);

pucMACArray[5] = ((ulUser1 >> 16) & 0xff);

/\* Program the MAC address. \*/

EthernetMACAddrSet(ETH\_BASE, pucMACArray);

xAddr.addr[ 0 ] = pucMACArray[0];

xAddr.addr[ 1 ] = pucMACArray[1];

xAddr.addr[ 2 ] = pucMACArray[2];

xAddr.addr[ 3 ] = pucMACArray[3];

xAddr.addr[ 4 ] = pucMACArray[4];

xAddr.addr[ 5 ] = pucMACArray[5];

uip\_setethaddr( xAddr );

}

/\*-----------------------------------------------------------\*/

char \*chav, \*pcText;

void vApplicationProcessFormInput( portCHAR \*pcInputString, portBASE\_TYPE xInputLength )

{

static portCHAR cMessageForDisplay[ 32 ];

//extern xQueueHandle xOLEDQueue;

//xOLEDMessage xOLEDMessage;

//xOLEDMessage xMessage;

//unsigned int x=0;

//unsigned int y=0;

//unsigned int xI=100;

//unsigned int yI=0;

//xMessage.X\_Locs=&x;

//xMessage.Y\_Locs=&y;

//xMessage.X\_LocI=&xI;

//xMessage.Y\_LocI=&yI;

/\* Process the form input sent by the IO page of the served HTML. \*/

chav = strstr( pcInputString, "?" );

if( chav )

{

// /\* Turn LED's on or off in accordance with the check box status. \*/

// if( strstr( c, "LED0=1" ) != NULL )

// {

// vParTestSetLED( 0, 1 );

// }

// else

// {

// vParTestSetLED( 0, 0 );

// }

//

/\* Find the start of the text to be displayed on the LCD. \*/

pcText = strstr( chav, "LCD=" );

pcText += strlen( "LCD=" );

/\* Terminate the file name for further processing within uIP. \*/

\*chav = 0x00;

/\* Terminate the LCD string. \*/

chav = strstr( pcText, " " );

if( chav != NULL )

{

\*chav = 0x00;

}

/\* Add required spaces. \*/

while( ( chav = strstr( pcText, "+" ) ) != NULL )

{

\*chav = ' ';

}

\*(myComs.cType)=\*(pcText);

if (\*(pcText)=='T')

sprintf(myComs.Payload,"%d", atoi(pcText+1));

/\* Write the message to the LCD. \*/

// strcpy( cMessageForDisplay, pcText );

// xMessage.pcMessage = ( signed portCHAR \* ) cMessageForDisplay;

// xQueueSend( xOLEDQueue, &xMessage, portMAX\_DELAY );

}

}

/\*

Xinyu Sui

fsdata.c

\*/

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// File System Image.

//

// This file was automatically generated using the makefsfile utility.

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

static const unsigned char data\_404\_html[] =

{

/\* /404.html \*/

0x2f, 0x34, 0x30, 0x34, 0x2e, 0x68, 0x74, 0x6d,

0x6c, 0x00,

0x3c, 0x68, 0x74, 0x6d, 0x6c, 0x3e, 0x0d, 0x0a,

0x20, 0x20, 0x3c, 0x62, 0x6f, 0x64, 0x79, 0x20,

0x62, 0x67, 0x63, 0x6f, 0x6c, 0x6f, 0x72, 0x3d,

0x22, 0x77, 0x68, 0x69, 0x74, 0x65, 0x22, 0x3e,

0x0d, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x3c, 0x63,

0x65, 0x6e, 0x74, 0x65, 0x72, 0x3e, 0x0d, 0x0a,

0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x3c, 0x68,

0x31, 0x3e, 0x34, 0x30, 0x34, 0x20, 0x2d, 0x20,

0x66, 0x69, 0x6c, 0x65, 0x20, 0x6e, 0x6f, 0x74,

0x20, 0x66, 0x6f, 0x75, 0x6e, 0x64, 0x3c, 0x2f,

0x68, 0x31, 0x3e, 0x0d, 0x0a, 0x20, 0x20, 0x20,

0x20, 0x20, 0x20, 0x3c, 0x68, 0x33, 0x3e, 0x47,

0x6f, 0x20, 0x3c, 0x61, 0x20, 0x68, 0x72, 0x65,

0x66, 0x3d, 0x22, 0x2f, 0x22, 0x3e, 0x68, 0x65,

0x72, 0x65, 0x3c, 0x2f, 0x61, 0x3e, 0x20, 0x69,

0x6e, 0x73, 0x74, 0x65, 0x61, 0x64, 0x2e, 0x3c,

0x2f, 0x68, 0x33, 0x3e, 0x0d, 0x0a, 0x20, 0x20,

0x20, 0x20, 0x3c, 0x2f, 0x63, 0x65, 0x6e, 0x74,

0x65, 0x72, 0x3e, 0x0d, 0x0a, 0x20, 0x20, 0x3c,

0x2f, 0x62, 0x6f, 0x64, 0x79, 0x3e, 0x0d, 0x0a,

0x3c, 0x2f, 0x68, 0x74, 0x6d, 0x6c, 0x3e,

};

static const unsigned char data\_command\_shtml[] =

{

/\* /command.shtml \*/

0x2f, 0x63, 0x6f, 0x6d, 0x6d, 0x61, 0x6e, 0x64,

0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c, 0x00,

0x3c, 0x21, 0x44, 0x4f, 0x43, 0x54, 0x59, 0x50,

0x45, 0x20, 0x48, 0x54, 0x4d, 0x4c, 0x20, 0x50,

0x55, 0x42, 0x4c, 0x49, 0x43, 0x20, 0x22, 0x2d,

0x2f, 0x2f, 0x57, 0x33, 0x43, 0x2f, 0x2f, 0x44,

0x54, 0x44, 0x20, 0x48, 0x54, 0x4d, 0x4c, 0x20,

0x34, 0x2e, 0x30, 0x31, 0x20, 0x54, 0x72, 0x61,

0x6e, 0x73, 0x69, 0x74, 0x69, 0x6f, 0x6e, 0x61,

0x6c, 0x2f, 0x2f, 0x45, 0x4e, 0x22, 0x20, 0x22,

0x68, 0x74, 0x74, 0x70, 0x3a, 0x2f, 0x2f, 0x77,

0x77, 0x77, 0x2e, 0x77, 0x33, 0x2e, 0x6f, 0x72,

0x67, 0x2f, 0x54, 0x52, 0x2f, 0x68, 0x74, 0x6d,

0x6c, 0x34, 0x2f, 0x6c, 0x6f, 0x6f, 0x73, 0x65,

0x2e, 0x64, 0x74, 0x64, 0x22, 0x3e, 0x0d, 0x0a,

0x3c, 0x68, 0x74, 0x6d, 0x6c, 0x3e, 0x0d, 0x0a,

0x20, 0x20, 0x3c, 0x68, 0x65, 0x61, 0x64, 0x3e,

0x0d, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x3c, 0x74,

0x69, 0x74, 0x6c, 0x65, 0x3e, 0x46, 0x72, 0x65,

0x65, 0x52, 0x54, 0x4f, 0x53, 0x2e, 0x6f, 0x72,

0x67, 0x20, 0x75, 0x49, 0x50, 0x20, 0x57, 0x45,

0x42, 0x20, 0x73, 0x65, 0x72, 0x76, 0x65, 0x72,

0x20, 0x64, 0x65, 0x6d, 0x6f, 0x3c, 0x2f, 0x74,

0x69, 0x74, 0x6c, 0x65, 0x3e, 0x0d, 0x0a, 0x20,

0x20, 0x3c, 0x2f, 0x68, 0x65, 0x61, 0x64, 0x3e,

0x0d, 0x0a, 0x20, 0x20, 0x3c, 0x42, 0x4f, 0x44,

0x59, 0x20, 0x62, 0x67, 0x63, 0x6f, 0x6c, 0x6f,

0x72, 0x3d, 0x22, 0x23, 0x43, 0x43, 0x43, 0x43,

0x66, 0x66, 0x22, 0x3e, 0x0d, 0x0a, 0x3c, 0x66,

0x6f, 0x6e, 0x74, 0x20, 0x66, 0x61, 0x63, 0x65,

0x3d, 0x22, 0x61, 0x72, 0x69, 0x61, 0x6c, 0x22,

0x3e, 0x0d, 0x0a, 0x3c, 0x61, 0x20, 0x68, 0x72,

0x65, 0x66, 0x3d, 0x22, 0x69, 0x6e, 0x64, 0x65,

0x78, 0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c, 0x22,

0x3e, 0x52, 0x54, 0x4f, 0x53, 0x20, 0x53, 0x74,

0x61, 0x74, 0x73, 0x3c, 0x2f, 0x61, 0x3e, 0x20,

0x3c, 0x62, 0x3e, 0x7c, 0x3c, 0x2f, 0x62, 0x3e,

0x20, 0x3c, 0x61, 0x20, 0x68, 0x72, 0x65, 0x66,

0x3d, 0x22, 0x73, 0x74, 0x61, 0x74, 0x73, 0x2e,

0x73, 0x68, 0x74, 0x6d, 0x6c, 0x22, 0x3e, 0x54,

0x43, 0x50, 0x20, 0x53, 0x74, 0x61, 0x74, 0x73,

0x3c, 0x2f, 0x61, 0x3e, 0x20, 0x3c, 0x62, 0x3e,

0x7c, 0x3c, 0x2f, 0x62, 0x3e, 0x20, 0x3c, 0x61,

0x20, 0x68, 0x72, 0x65, 0x66, 0x3d, 0x22, 0x74,

0x63, 0x70, 0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c,

0x22, 0x3e, 0x43, 0x6f, 0x6e, 0x6e, 0x65, 0x63,

0x74, 0x69, 0x6f, 0x6e, 0x73, 0x3c, 0x2f, 0x61,

0x3e, 0x20, 0x3c, 0x62, 0x3e, 0x7c, 0x3c, 0x2f,

0x62, 0x3e, 0x20, 0x3c, 0x61, 0x20, 0x68, 0x72,

0x65, 0x66, 0x3d, 0x22, 0x64, 0x69, 0x73, 0x70,

0x6c, 0x61, 0x79, 0x2e, 0x73, 0x68, 0x74, 0x6d,

0x6c, 0x22, 0x3e, 0x44, 0x69, 0x73, 0x70, 0x6c,

0x61, 0x79, 0x3c, 0x2f, 0x61, 0x3e, 0x20, 0x3c,

0x62, 0x3e, 0x7c, 0x3c, 0x2f, 0x62, 0x3e, 0x20,

0x3c, 0x61, 0x20, 0x68, 0x72, 0x65, 0x66, 0x3d,

0x22, 0x63, 0x6f, 0x6d, 0x6d, 0x61, 0x6e, 0x64,

0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c, 0x22, 0x3e,

0x43, 0x6f, 0x6d, 0x6d, 0x61, 0x6e, 0x64, 0x3c,

0x2f, 0x61, 0x3e, 0x20, 0x3c, 0x62, 0x3e, 0x7c,

0x3c, 0x2f, 0x62, 0x3e, 0x20, 0x3c, 0x61, 0x20,

0x68, 0x72, 0x65, 0x66, 0x3d, 0x22, 0x69, 0x6f,

0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c, 0x22, 0x3e,

0x49, 0x4f, 0x3c, 0x2f, 0x61, 0x3e, 0x0d, 0x0a,

0x3c, 0x62, 0x72, 0x3e, 0x3c, 0x70, 0x3e, 0x0d,

0x0a, 0x3c, 0x68, 0x72, 0x3e, 0x0d, 0x0a, 0x3c,

0x62, 0x3e, 0x43, 0x6f, 0x6d, 0x6d, 0x61, 0x6e,

0x64, 0x3c, 0x2f, 0x62, 0x3e, 0x3c, 0x62, 0x72,

0x3e, 0x0d, 0x0a, 0x0d, 0x0a, 0x3c, 0x70, 0x3e,

0x0d, 0x0a, 0x0d, 0x0a, 0x54, 0x6f, 0x20, 0x73,

0x65, 0x6e, 0x64, 0x20, 0x61, 0x20, 0x63, 0x6f,

0x6d, 0x6d, 0x61, 0x6e, 0x64, 0x2c, 0x20, 0x65,

0x6e, 0x74, 0x65, 0x72, 0x20, 0x74, 0x68, 0x65,

0x20, 0x63, 0x6f, 0x6d, 0x6d, 0x61, 0x6e, 0x64,

0x20, 0x69, 0x6e, 0x74, 0x6f, 0x20, 0x74, 0x68,

0x65, 0x20, 0x74, 0x65, 0x78, 0x74, 0x20, 0x62,

0x6f, 0x78, 0x2c, 0x20, 0x74, 0x68, 0x65, 0x6e,

0x20, 0x63, 0x6c, 0x69, 0x63, 0x6b, 0x20, 0x22,

0x55, 0x70, 0x64, 0x61, 0x74, 0x65, 0x20, 0x49,

0x4f, 0x22, 0x2e, 0x0d, 0x0a, 0x0d, 0x0a, 0x0d,

0x0a, 0x3c, 0x70, 0x3e, 0x0d, 0x0a, 0x3c, 0x66,

0x6f, 0x72, 0x6d, 0x20, 0x6e, 0x61, 0x6d, 0x65,

0x3d, 0x22, 0x61, 0x46, 0x6f, 0x72, 0x6d, 0x22,

0x20, 0x61, 0x63, 0x74, 0x69, 0x6f, 0x6e, 0x3d,

0x22, 0x2f, 0x63, 0x6f, 0x6d, 0x6d, 0x61, 0x6e,

0x64, 0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c, 0x22,

0x20, 0x6d, 0x65, 0x74, 0x68, 0x6f, 0x64, 0x3d,

0x22, 0x67, 0x65, 0x74, 0x22, 0x3e, 0x0d, 0x0a,

0x25, 0x21, 0x20, 0x63, 0x6f, 0x6d, 0x6d, 0x61,

0x6e, 0x64, 0x2d, 0x69, 0x6e, 0x0d, 0x0a, 0x3c,

0x70, 0x3e, 0x0d, 0x0a, 0x3c, 0x69, 0x6e, 0x70,

0x75, 0x74, 0x20, 0x74, 0x79, 0x70, 0x65, 0x3d,

0x22, 0x73, 0x75, 0x62, 0x6d, 0x69, 0x74, 0x22,

0x20, 0x76, 0x61, 0x6c, 0x75, 0x65, 0x3d, 0x22,

0x55, 0x70, 0x64, 0x61, 0x74, 0x65, 0x20, 0x49,

0x4f, 0x22, 0x3e, 0x0d, 0x0a, 0x3c, 0x2f, 0x66,

0x6f, 0x72, 0x6d, 0x3e, 0x0d, 0x0a, 0x3c, 0x62,

0x72, 0x3e, 0x3c, 0x70, 0x3e, 0x0d, 0x0a, 0x3c,

0x2f, 0x66, 0x6f, 0x6e, 0x74, 0x3e, 0x0d, 0x0a,

0x3c, 0x2f, 0x62, 0x6f, 0x64, 0x79, 0x3e, 0x0d,

0x0a, 0x3c, 0x2f, 0x68, 0x74, 0x6d, 0x6c, 0x3e,

0x0d, 0x0a,

};

static const unsigned char data\_display\_shtml[] =

{

/\* /display.shtml \*/

0x2f, 0x64, 0x69, 0x73, 0x70, 0x6c, 0x61, 0x79,

0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c, 0x00,

0x3c, 0x21, 0x44, 0x4f, 0x43, 0x54, 0x59, 0x50,

0x45, 0x20, 0x48, 0x54, 0x4d, 0x4c, 0x20, 0x50,

0x55, 0x42, 0x4c, 0x49, 0x43, 0x20, 0x22, 0x2d,

0x2f, 0x2f, 0x57, 0x33, 0x43, 0x2f, 0x2f, 0x44,

0x54, 0x44, 0x20, 0x48, 0x54, 0x4d, 0x4c, 0x20,

0x34, 0x2e, 0x30, 0x31, 0x20, 0x54, 0x72, 0x61,

0x6e, 0x73, 0x69, 0x74, 0x69, 0x6f, 0x6e, 0x61,

0x6c, 0x2f, 0x2f, 0x45, 0x4e, 0x22, 0x20, 0x22,

0x68, 0x74, 0x74, 0x70, 0x3a, 0x2f, 0x2f, 0x77,

0x77, 0x77, 0x2e, 0x77, 0x33, 0x2e, 0x6f, 0x72,

0x67, 0x2f, 0x54, 0x52, 0x2f, 0x68, 0x74, 0x6d,

0x6c, 0x34, 0x2f, 0x6c, 0x6f, 0x6f, 0x73, 0x65,

0x2e, 0x64, 0x74, 0x64, 0x22, 0x3e, 0x0d, 0x0a,

0x3c, 0x68, 0x74, 0x6d, 0x6c, 0x3e, 0x0d, 0x0a,

0x20, 0x20, 0x3c, 0x68, 0x65, 0x61, 0x64, 0x3e,

0x0d, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x3c, 0x74,

0x69, 0x74, 0x6c, 0x65, 0x3e, 0x46, 0x72, 0x65,

0x65, 0x52, 0x54, 0x4f, 0x53, 0x2e, 0x6f, 0x72,

0x67, 0x20, 0x75, 0x49, 0x50, 0x20, 0x57, 0x45,

0x42, 0x20, 0x73, 0x65, 0x72, 0x76, 0x65, 0x72,

0x20, 0x64, 0x65, 0x6d, 0x6f, 0x3c, 0x2f, 0x74,

0x69, 0x74, 0x6c, 0x65, 0x3e, 0x0d, 0x0a, 0x20,

0x20, 0x3c, 0x2f, 0x68, 0x65, 0x61, 0x64, 0x3e,

0x0d, 0x0a, 0x20, 0x20, 0x3c, 0x42, 0x4f, 0x44,

0x59, 0x20, 0x6f, 0x6e, 0x4c, 0x6f, 0x61, 0x64,

0x3d, 0x22, 0x77, 0x69, 0x6e, 0x64, 0x6f, 0x77,

0x2e, 0x73, 0x65, 0x74, 0x54, 0x69, 0x6d, 0x65,

0x6f, 0x75, 0x74, 0x28, 0x26, 0x71, 0x75, 0x6f,

0x74, 0x3b, 0x6c, 0x6f, 0x63, 0x61, 0x74, 0x69,

0x6f, 0x6e, 0x2e, 0x68, 0x72, 0x65, 0x66, 0x3d,

0x27, 0x64, 0x69, 0x73, 0x70, 0x6c, 0x61, 0x79,

0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c, 0x27, 0x26,

0x71, 0x75, 0x6f, 0x74, 0x3b, 0x2c, 0x35, 0x30,

0x30, 0x30, 0x29, 0x22, 0x62, 0x67, 0x63, 0x6f,

0x6c, 0x6f, 0x72, 0x3d, 0x22, 0x23, 0x43, 0x43,

0x43, 0x43, 0x66, 0x66, 0x22, 0x3e, 0x0d, 0x0a,

0x3c, 0x66, 0x6f, 0x6e, 0x74, 0x20, 0x66, 0x61,

0x63, 0x65, 0x3d, 0x22, 0x61, 0x72, 0x69, 0x61,

0x6c, 0x22, 0x3e, 0x0d, 0x0a, 0x3c, 0x61, 0x20,

0x68, 0x72, 0x65, 0x66, 0x3d, 0x22, 0x69, 0x6e,

0x64, 0x65, 0x78, 0x2e, 0x73, 0x68, 0x74, 0x6d,

0x6c, 0x22, 0x3e, 0x52, 0x54, 0x4f, 0x53, 0x20,

0x53, 0x74, 0x61, 0x74, 0x73, 0x3c, 0x2f, 0x61,

0x3e, 0x20, 0x3c, 0x62, 0x3e, 0x7c, 0x3c, 0x2f,

0x62, 0x3e, 0x20, 0x3c, 0x61, 0x20, 0x68, 0x72,

0x65, 0x66, 0x3d, 0x22, 0x73, 0x74, 0x61, 0x74,

0x73, 0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c, 0x22,

0x3e, 0x54, 0x43, 0x50, 0x20, 0x53, 0x74, 0x61,

0x74, 0x73, 0x3c, 0x2f, 0x61, 0x3e, 0x20, 0x3c,

0x62, 0x3e, 0x7c, 0x3c, 0x2f, 0x62, 0x3e, 0x20,

0x3c, 0x61, 0x20, 0x68, 0x72, 0x65, 0x66, 0x3d,

0x22, 0x74, 0x63, 0x70, 0x2e, 0x73, 0x68, 0x74,

0x6d, 0x6c, 0x22, 0x3e, 0x43, 0x6f, 0x6e, 0x6e,

0x65, 0x63, 0x74, 0x69, 0x6f, 0x6e, 0x73, 0x3c,

0x2f, 0x61, 0x3e, 0x20, 0x3c, 0x62, 0x3e, 0x7c,

0x3c, 0x2f, 0x62, 0x3e, 0x20, 0x3c, 0x61, 0x20,

0x68, 0x72, 0x65, 0x66, 0x3d, 0x22, 0x64, 0x69,

0x73, 0x70, 0x6c, 0x61, 0x79, 0x2e, 0x73, 0x68,

0x74, 0x6d, 0x6c, 0x22, 0x3e, 0x44, 0x69, 0x73,

0x70, 0x6c, 0x61, 0x79, 0x3c, 0x2f, 0x61, 0x3e,

0x20, 0x3c, 0x62, 0x3e, 0x7c, 0x3c, 0x2f, 0x62,

0x3e, 0x20, 0x3c, 0x61, 0x20, 0x68, 0x72, 0x65,

0x66, 0x3d, 0x22, 0x63, 0x6f, 0x6d, 0x6d, 0x61,

0x6e, 0x64, 0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c,

0x22, 0x3e, 0x43, 0x6f, 0x6d, 0x6d, 0x61, 0x6e,

0x64, 0x3c, 0x2f, 0x61, 0x3e, 0x20, 0x3c, 0x62,

0x3e, 0x7c, 0x3c, 0x2f, 0x62, 0x3e, 0x20, 0x3c,

0x61, 0x20, 0x68, 0x72, 0x65, 0x66, 0x3d, 0x22,

0x69, 0x6f, 0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c,

0x22, 0x3e, 0x49, 0x4f, 0x3c, 0x2f, 0x61, 0x3e,

0x0d, 0x0a, 0x3c, 0x62, 0x72, 0x3e, 0x3c, 0x70,

0x3e, 0x0d, 0x0a, 0x3c, 0x68, 0x72, 0x3e, 0x0d,

0x0a, 0x3c, 0x62, 0x72, 0x3e, 0x3c, 0x70, 0x3e,

0x0d, 0x0a, 0x3c, 0x68, 0x32, 0x3e, 0x0d, 0x0a,

0x53, 0x61, 0x74, 0x65, 0x6c, 0x6c, 0x69, 0x74,

0x65, 0x3a, 0x26, 0x6e, 0x62, 0x73, 0x70, 0x3b,

0x26, 0x6e, 0x62, 0x73, 0x70, 0x3b, 0x26, 0x6e,

0x62, 0x73, 0x70, 0x3b, 0x26, 0x6e, 0x62, 0x73,

0x70, 0x3b, 0x26, 0x6e, 0x62, 0x73, 0x70, 0x3b,

0x26, 0x6e, 0x62, 0x73, 0x70, 0x3b, 0x26, 0x6e,

0x62, 0x73, 0x70, 0x3b, 0x4e, 0x41, 0x4d, 0x45,

0x3c, 0x62, 0x72, 0x3e, 0x0d, 0x0a, 0x44, 0x61,

0x74, 0x65, 0x3a, 0x26, 0x6e, 0x62, 0x73, 0x70,

0x3b, 0x26, 0x6e, 0x62, 0x73, 0x70, 0x3b, 0x26,

0x6e, 0x62, 0x73, 0x70, 0x3b, 0x26, 0x6e, 0x62,

0x73, 0x70, 0x3b, 0x26, 0x6e, 0x62, 0x73, 0x70,

0x3b, 0x26, 0x6e, 0x62, 0x73, 0x70, 0x3b, 0x26,

0x6e, 0x62, 0x73, 0x70, 0x3b, 0x26, 0x6e, 0x62,

0x73, 0x70, 0x3b, 0x26, 0x6e, 0x62, 0x73, 0x70,

0x3b, 0x26, 0x6e, 0x62, 0x73, 0x70, 0x3b, 0x26,

0x6e, 0x62, 0x73, 0x70, 0x3b, 0x26, 0x6e, 0x62,

0x73, 0x70, 0x3b, 0x26, 0x6e, 0x62, 0x73, 0x70,

0x3b, 0x3c, 0x73, 0x70, 0x61, 0x6e, 0x20, 0x69,

0x64, 0x3d, 0x22, 0x74, 0x69, 0x6d, 0x65, 0x22,

0x3e, 0x3c, 0x2f, 0x73, 0x70, 0x61, 0x6e, 0x3e,

0x3c, 0x62, 0x72, 0x3e, 0x0d, 0x0a, 0x4f, 0x70,

0x65, 0x72, 0x61, 0x74, 0x6f, 0x72, 0x3a, 0x09,

0x09, 0x09, 0x3c, 0x62, 0x72, 0x3e, 0x0d, 0x0a,

0x3c, 0x2f, 0x68, 0x32, 0x3e, 0x0d, 0x0a, 0x0d,

0x0a, 0x3c, 0x73, 0x63, 0x72, 0x69, 0x70, 0x74,

0x20, 0x6c, 0x61, 0x6e, 0x67, 0x75, 0x61, 0x67,

0x65, 0x3d, 0x22, 0x6a, 0x61, 0x76, 0x61, 0x73,

0x63, 0x72, 0x69, 0x70, 0x74, 0x22, 0x3e, 0x0d,

0x0a, 0x76, 0x61, 0x72, 0x20, 0x74, 0x6f, 0x64,

0x61, 0x79, 0x20, 0x3d, 0x20, 0x6e, 0x65, 0x77,

0x20, 0x44, 0x61, 0x74, 0x65, 0x28, 0x29, 0x3b,

0x0d, 0x0a, 0x64, 0x6f, 0x63, 0x75, 0x6d, 0x65,

0x6e, 0x74, 0x2e, 0x67, 0x65, 0x74, 0x45, 0x6c,

0x65, 0x6d, 0x65, 0x6e, 0x74, 0x42, 0x79, 0x49,

0x64, 0x28, 0x27, 0x74, 0x69, 0x6d, 0x65, 0x27,

0x29, 0x2e, 0x69, 0x6e, 0x6e, 0x65, 0x72, 0x48,

0x54, 0x4d, 0x4c, 0x3d, 0x74, 0x6f, 0x64, 0x61,

0x79, 0x3b, 0x0d, 0x0a, 0x3c, 0x2f, 0x73, 0x63,

0x72, 0x69, 0x70, 0x74, 0x3e, 0x0d, 0x0a, 0x0d,

0x0a, 0x3c, 0x74, 0x61, 0x62, 0x6c, 0x65, 0x20,

0x77, 0x69, 0x64, 0x74, 0x68, 0x3d, 0x22, 0x33,

0x30, 0x30, 0x22, 0x20, 0x62, 0x6f, 0x72, 0x64,

0x65, 0x72, 0x3d, 0x22, 0x30, 0x22, 0x3e, 0x0d,

0x0a, 0x3c, 0x74, 0x72, 0x3e, 0x3c, 0x74, 0x64,

0x20, 0x61, 0x6c, 0x69, 0x67, 0x6e, 0x3d, 0x22,

0x6c, 0x65, 0x66, 0x74, 0x22, 0x3e, 0x3c, 0x66,

0x6f, 0x6e, 0x74, 0x20, 0x66, 0x61, 0x63, 0x65,

0x3d, 0x22, 0x63, 0x6f, 0x75, 0x72, 0x69, 0x65,

0x72, 0x22, 0x3e, 0x3c, 0x70, 0x72, 0x65, 0x3e,

0x0d, 0x0a, 0x46, 0x75, 0x65, 0x6c, 0x20, 0x4c,

0x6f, 0x77, 0x3a, 0x0d, 0x0a, 0x42, 0x61, 0x74,

0x74, 0x65, 0x72, 0x79, 0x20, 0x4c, 0x6f, 0x77,

0x3a, 0x0d, 0x0a, 0x53, 0x6f, 0x6c, 0x61, 0x72,

0x20, 0x50, 0x61, 0x6e, 0x65, 0x6c, 0x20, 0x53,

0x74, 0x61, 0x74, 0x65, 0x3a, 0x0d, 0x0a, 0x42,

0x61, 0x74, 0x74, 0x65, 0x72, 0x79, 0x20, 0x4c,

0x65, 0x76, 0x65, 0x6c, 0x3a, 0x0d, 0x0a, 0x46,

0x75, 0x65, 0x6c, 0x20, 0x4c, 0x65, 0x76, 0x65,

0x6c, 0x3a, 0x0d, 0x0a, 0x50, 0x6f, 0x77, 0x65,

0x72, 0x20, 0x43, 0x6f, 0x6e, 0x73, 0x75, 0x6d,

0x70, 0x74, 0x69, 0x6f, 0x6e, 0x3a, 0x0d, 0x0a,

0x42, 0x61, 0x74, 0x74, 0x65, 0x72, 0x79, 0x20,

0x54, 0x65, 0x6d, 0x70, 0x65, 0x72, 0x61, 0x74,

0x75, 0x72, 0x65, 0x20, 0x31, 0x3a, 0x0d, 0x0a,

0x42, 0x61, 0x74, 0x74, 0x65, 0x72, 0x79, 0x20,

0x54, 0x65, 0x6d, 0x70, 0x65, 0x72, 0x61, 0x74,

0x75, 0x72, 0x65, 0x20, 0x32, 0x3a, 0x0d, 0x0a,

0x54, 0x72, 0x61, 0x6e, 0x73, 0x70, 0x6f, 0x72,

0x74, 0x20, 0x44, 0x69, 0x73, 0x74, 0x61, 0x6e,

0x63, 0x65, 0x3a, 0x0d, 0x0a, 0x49, 0x6d, 0x61,

0x67, 0x65, 0x20, 0x44, 0x61, 0x74, 0x61, 0x3a,

0x0d, 0x0a, 0x3c, 0x2f, 0x70, 0x72, 0x65, 0x3e,

0x3c, 0x2f, 0x66, 0x6f, 0x6e, 0x74, 0x3e, 0x3c,

0x2f, 0x74, 0x64, 0x3e, 0x3c, 0x74, 0x64, 0x3e,

0x3c, 0x70, 0x72, 0x65, 0x3e, 0x25, 0x21, 0x20,

0x63, 0x6f, 0x6d, 0x6d, 0x75, 0x6e, 0x69, 0x63,

0x2d, 0x6f, 0x75, 0x74, 0x0d, 0x0a, 0x3c, 0x2f,

0x70, 0x72, 0x65, 0x3e, 0x3c, 0x2f, 0x74, 0x61,

0x62, 0x6c, 0x65, 0x3e, 0x0d, 0x0a, 0x3c, 0x62,

0x72, 0x3e, 0x3c, 0x70, 0x3e, 0x0d, 0x0a, 0x3c,

0x2f, 0x66, 0x6f, 0x6e, 0x74, 0x3e, 0x0d, 0x0a,

0x3c, 0x2f, 0x62, 0x6f, 0x64, 0x79, 0x3e, 0x0d,

0x0a, 0x3c, 0x2f, 0x68, 0x74, 0x6d, 0x6c, 0x3e,

0x0d, 0x0a,

};

static const unsigned char data\_index\_html[] =

{

/\* /index.html \*/

0x2f, 0x69, 0x6e, 0x64, 0x65, 0x78, 0x2e, 0x68,

0x74, 0x6d, 0x6c, 0x00,

0x3c, 0x21, 0x44, 0x4f, 0x43, 0x54, 0x59, 0x50,

0x45, 0x20, 0x48, 0x54, 0x4d, 0x4c, 0x20, 0x50,

0x55, 0x42, 0x4c, 0x49, 0x43, 0x20, 0x22, 0x2d,

0x2f, 0x2f, 0x57, 0x33, 0x43, 0x2f, 0x2f, 0x44,

0x54, 0x44, 0x20, 0x48, 0x54, 0x4d, 0x4c, 0x20,

0x34, 0x2e, 0x30, 0x31, 0x20, 0x54, 0x72, 0x61,

0x6e, 0x73, 0x69, 0x74, 0x69, 0x6f, 0x6e, 0x61,

0x6c, 0x2f, 0x2f, 0x45, 0x4e, 0x22, 0x20, 0x22,

0x68, 0x74, 0x74, 0x70, 0x3a, 0x2f, 0x2f, 0x77,

0x77, 0x77, 0x2e, 0x77, 0x33, 0x2e, 0x6f, 0x72,

0x67, 0x2f, 0x54, 0x52, 0x2f, 0x68, 0x74, 0x6d,

0x6c, 0x34, 0x2f, 0x6c, 0x6f, 0x6f, 0x73, 0x65,

0x2e, 0x64, 0x74, 0x64, 0x22, 0x3e, 0x0d, 0x0a,

0x3c, 0x68, 0x74, 0x6d, 0x6c, 0x3e, 0x0d, 0x0a,

0x20, 0x20, 0x3c, 0x68, 0x65, 0x61, 0x64, 0x3e,

0x0d, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x3c, 0x74,

0x69, 0x74, 0x6c, 0x65, 0x3e, 0x46, 0x72, 0x65,

0x65, 0x52, 0x54, 0x4f, 0x53, 0x2e, 0x6f, 0x72,

0x67, 0x20, 0x75, 0x49, 0x50, 0x20, 0x57, 0x45,

0x42, 0x20, 0x73, 0x65, 0x72, 0x76, 0x65, 0x72,

0x20, 0x64, 0x65, 0x6d, 0x6f, 0x3c, 0x2f, 0x74,

0x69, 0x74, 0x6c, 0x65, 0x3e, 0x0d, 0x0a, 0x20,

0x20, 0x3c, 0x2f, 0x68, 0x65, 0x61, 0x64, 0x3e,

0x0d, 0x0a, 0x20, 0x20, 0x3c, 0x42, 0x4f, 0x44,

0x59, 0x20, 0x6f, 0x6e, 0x4c, 0x6f, 0x61, 0x64,

0x3d, 0x22, 0x77, 0x69, 0x6e, 0x64, 0x6f, 0x77,

0x2e, 0x73, 0x65, 0x74, 0x54, 0x69, 0x6d, 0x65,

0x6f, 0x75, 0x74, 0x28, 0x26, 0x71, 0x75, 0x6f,

0x74, 0x3b, 0x6c, 0x6f, 0x63, 0x61, 0x74, 0x69,

0x6f, 0x6e, 0x2e, 0x68, 0x72, 0x65, 0x66, 0x3d,

0x27, 0x69, 0x6e, 0x64, 0x65, 0x78, 0x2e, 0x73,

0x68, 0x74, 0x6d, 0x6c, 0x27, 0x26, 0x71, 0x75,

0x6f, 0x74, 0x3b, 0x2c, 0x35, 0x30, 0x30, 0x30,

0x29, 0x22, 0x62, 0x67, 0x63, 0x6f, 0x6c, 0x6f,

0x72, 0x3d, 0x22, 0x23, 0x43, 0x43, 0x43, 0x43,

0x66, 0x66, 0x22, 0x3e, 0x0d, 0x0a, 0x3c, 0x66,

0x6f, 0x6e, 0x74, 0x20, 0x66, 0x61, 0x63, 0x65,

0x3d, 0x22, 0x61, 0x72, 0x69, 0x61, 0x6c, 0x22,

0x3e, 0x0d, 0x0a, 0x3c, 0x61, 0x20, 0x68, 0x72,

0x65, 0x66, 0x3d, 0x22, 0x69, 0x6e, 0x64, 0x65,

0x78, 0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c, 0x22,

0x3e, 0x52, 0x54, 0x4f, 0x53, 0x20, 0x53, 0x74,

0x61, 0x74, 0x73, 0x3c, 0x2f, 0x61, 0x3e, 0x20,

0x3c, 0x62, 0x3e, 0x7c, 0x3c, 0x2f, 0x62, 0x3e,

0x20, 0x3c, 0x61, 0x20, 0x68, 0x72, 0x65, 0x66,

0x3d, 0x22, 0x73, 0x74, 0x61, 0x74, 0x73, 0x2e,

0x73, 0x68, 0x74, 0x6d, 0x6c, 0x22, 0x3e, 0x54,

0x43, 0x50, 0x20, 0x53, 0x74, 0x61, 0x74, 0x73,

0x3c, 0x2f, 0x61, 0x3e, 0x20, 0x3c, 0x62, 0x3e,

0x7c, 0x3c, 0x2f, 0x62, 0x3e, 0x20, 0x3c, 0x61,

0x20, 0x68, 0x72, 0x65, 0x66, 0x3d, 0x22, 0x74,

0x63, 0x70, 0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c,

0x22, 0x3e, 0x43, 0x6f, 0x6e, 0x6e, 0x65, 0x63,

0x74, 0x69, 0x6f, 0x6e, 0x73, 0x3c, 0x2f, 0x61,

0x3e, 0x20, 0x3c, 0x62, 0x3e, 0x7c, 0x3c, 0x2f,

0x62, 0x3e, 0x20, 0x3c, 0x61, 0x20, 0x68, 0x72,

0x65, 0x66, 0x3d, 0x22, 0x64, 0x69, 0x73, 0x70,

0x6c, 0x61, 0x79, 0x2e, 0x73, 0x68, 0x74, 0x6d,

0x6c, 0x22, 0x3e, 0x44, 0x69, 0x73, 0x70, 0x6c,

0x61, 0x79, 0x3c, 0x2f, 0x61, 0x3e, 0x20, 0x3c,

0x62, 0x3e, 0x7c, 0x3c, 0x2f, 0x62, 0x3e, 0x20,

0x3c, 0x61, 0x20, 0x68, 0x72, 0x65, 0x66, 0x3d,

0x22, 0x63, 0x6f, 0x6d, 0x6d, 0x61, 0x6e, 0x64,

0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c, 0x22, 0x3e,

0x43, 0x6f, 0x6d, 0x6d, 0x61, 0x6e, 0x64, 0x3c,

0x2f, 0x61, 0x3e, 0x20, 0x3c, 0x62, 0x3e, 0x7c,

0x3c, 0x2f, 0x62, 0x3e, 0x20, 0x3c, 0x61, 0x20,

0x68, 0x72, 0x65, 0x66, 0x3d, 0x22, 0x69, 0x6f,

0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c, 0x22, 0x3e,

0x49, 0x4f, 0x3c, 0x2f, 0x61, 0x3e, 0x0d, 0x0a,

0x3c, 0x62, 0x72, 0x3e, 0x3c, 0x70, 0x3e, 0x0d,

0x0a, 0x3c, 0x68, 0x72, 0x3e, 0x0d, 0x0a, 0x3c,

0x62, 0x72, 0x3e, 0x3c, 0x70, 0x3e, 0x0d, 0x0a,

0x3c, 0x68, 0x32, 0x3e, 0x54, 0x61, 0x73, 0x6b,

0x20, 0x73, 0x74, 0x61, 0x74, 0x69, 0x73, 0x74,

0x69, 0x63, 0x73, 0x3c, 0x2f, 0x68, 0x32, 0x3e,

0x0d, 0x0a, 0x50, 0x61, 0x67, 0x65, 0x20, 0x77,

0x69, 0x6c, 0x6c, 0x20, 0x72, 0x65, 0x66, 0x72,

0x65, 0x73, 0x68, 0x20, 0x65, 0x76, 0x65, 0x72,

0x79, 0x20, 0x35, 0x20, 0x73, 0x65, 0x63, 0x6f,

0x6e, 0x64, 0x73, 0x2e, 0x3c, 0x70, 0x3e, 0x0d,

0x0a, 0x3c, 0x66, 0x6f, 0x6e, 0x74, 0x20, 0x66,

0x61, 0x63, 0x65, 0x3d, 0x22, 0x63, 0x6f, 0x75,

0x72, 0x69, 0x65, 0x72, 0x22, 0x3e, 0x3c, 0x70,

0x72, 0x65, 0x3e, 0x54, 0x61, 0x73, 0x6b, 0x20,

0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,

0x20, 0x53, 0x74, 0x61, 0x74, 0x65, 0x20, 0x20,

0x50, 0x72, 0x69, 0x6f, 0x72, 0x69, 0x74, 0x79,

0x20, 0x20, 0x53, 0x74, 0x61, 0x63, 0x6b, 0x09,

0x23, 0x3c, 0x62, 0x72, 0x3e, 0x2a, 0x2a, 0x2a,

0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a,

0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a,

0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a,

0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a,

0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a,

0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x3c, 0x62, 0x72,

0x3e, 0x0d, 0x0a, 0x25, 0x21, 0x20, 0x72, 0x74,

0x6f, 0x73, 0x2d, 0x73, 0x74, 0x61, 0x74, 0x73,

0x0d, 0x0a, 0x3c, 0x2f, 0x70, 0x72, 0x65, 0x3e,

0x3c, 0x2f, 0x66, 0x6f, 0x6e, 0x74, 0x3e, 0x0d,

0x0a, 0x3c, 0x2f, 0x66, 0x6f, 0x6e, 0x74, 0x3e,

0x0d, 0x0a, 0x3c, 0x2f, 0x62, 0x6f, 0x64, 0x79,

0x3e, 0x0d, 0x0a, 0x3c, 0x2f, 0x68, 0x74, 0x6d,

0x6c, 0x3e, 0x0d, 0x0a, 0x0d, 0x0a,

};

static const unsigned char data\_index\_shtml[] =

{

/\* /index.shtml \*/

0x2f, 0x69, 0x6e, 0x64, 0x65, 0x78, 0x2e, 0x73,

0x68, 0x74, 0x6d, 0x6c, 0x00,

0x3c, 0x21, 0x44, 0x4f, 0x43, 0x54, 0x59, 0x50,

0x45, 0x20, 0x48, 0x54, 0x4d, 0x4c, 0x20, 0x50,

0x55, 0x42, 0x4c, 0x49, 0x43, 0x20, 0x22, 0x2d,

0x2f, 0x2f, 0x57, 0x33, 0x43, 0x2f, 0x2f, 0x44,

0x54, 0x44, 0x20, 0x48, 0x54, 0x4d, 0x4c, 0x20,

0x34, 0x2e, 0x30, 0x31, 0x20, 0x54, 0x72, 0x61,

0x6e, 0x73, 0x69, 0x74, 0x69, 0x6f, 0x6e, 0x61,

0x6c, 0x2f, 0x2f, 0x45, 0x4e, 0x22, 0x20, 0x22,

0x68, 0x74, 0x74, 0x70, 0x3a, 0x2f, 0x2f, 0x77,

0x77, 0x77, 0x2e, 0x77, 0x33, 0x2e, 0x6f, 0x72,

0x67, 0x2f, 0x54, 0x52, 0x2f, 0x68, 0x74, 0x6d,

0x6c, 0x34, 0x2f, 0x6c, 0x6f, 0x6f, 0x73, 0x65,

0x2e, 0x64, 0x74, 0x64, 0x22, 0x3e, 0x0d, 0x0a,

0x3c, 0x68, 0x74, 0x6d, 0x6c, 0x3e, 0x0d, 0x0a,

0x20, 0x20, 0x3c, 0x68, 0x65, 0x61, 0x64, 0x3e,

0x0d, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x3c, 0x74,

0x69, 0x74, 0x6c, 0x65, 0x3e, 0x46, 0x72, 0x65,

0x65, 0x52, 0x54, 0x4f, 0x53, 0x2e, 0x6f, 0x72,

0x67, 0x20, 0x75, 0x49, 0x50, 0x20, 0x57, 0x45,

0x42, 0x20, 0x73, 0x65, 0x72, 0x76, 0x65, 0x72,

0x20, 0x64, 0x65, 0x6d, 0x6f, 0x3c, 0x2f, 0x74,

0x69, 0x74, 0x6c, 0x65, 0x3e, 0x0d, 0x0a, 0x20,

0x20, 0x3c, 0x2f, 0x68, 0x65, 0x61, 0x64, 0x3e,

0x0d, 0x0a, 0x20, 0x20, 0x3c, 0x42, 0x4f, 0x44,

0x59, 0x20, 0x6f, 0x6e, 0x4c, 0x6f, 0x61, 0x64,

0x3d, 0x22, 0x77, 0x69, 0x6e, 0x64, 0x6f, 0x77,

0x2e, 0x73, 0x65, 0x74, 0x54, 0x69, 0x6d, 0x65,

0x6f, 0x75, 0x74, 0x28, 0x26, 0x71, 0x75, 0x6f,

0x74, 0x3b, 0x6c, 0x6f, 0x63, 0x61, 0x74, 0x69,

0x6f, 0x6e, 0x2e, 0x68, 0x72, 0x65, 0x66, 0x3d,

0x27, 0x69, 0x6e, 0x64, 0x65, 0x78, 0x2e, 0x73,

0x68, 0x74, 0x6d, 0x6c, 0x27, 0x26, 0x71, 0x75,

0x6f, 0x74, 0x3b, 0x2c, 0x35, 0x30, 0x30, 0x30,

0x29, 0x22, 0x62, 0x67, 0x63, 0x6f, 0x6c, 0x6f,

0x72, 0x3d, 0x22, 0x23, 0x43, 0x43, 0x43, 0x43,

0x66, 0x66, 0x22, 0x3e, 0x0d, 0x0a, 0x3c, 0x66,

0x6f, 0x6e, 0x74, 0x20, 0x66, 0x61, 0x63, 0x65,

0x3d, 0x22, 0x61, 0x72, 0x69, 0x61, 0x6c, 0x22,

0x3e, 0x0d, 0x0a, 0x3c, 0x61, 0x20, 0x68, 0x72,

0x65, 0x66, 0x3d, 0x22, 0x69, 0x6e, 0x64, 0x65,

0x78, 0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c, 0x22,

0x3e, 0x52, 0x54, 0x4f, 0x53, 0x20, 0x53, 0x74,

0x61, 0x74, 0x73, 0x3c, 0x2f, 0x61, 0x3e, 0x20,

0x3c, 0x62, 0x3e, 0x7c, 0x3c, 0x2f, 0x62, 0x3e,

0x20, 0x3c, 0x61, 0x20, 0x68, 0x72, 0x65, 0x66,

0x3d, 0x22, 0x73, 0x74, 0x61, 0x74, 0x73, 0x2e,

0x73, 0x68, 0x74, 0x6d, 0x6c, 0x22, 0x3e, 0x54,

0x43, 0x50, 0x20, 0x53, 0x74, 0x61, 0x74, 0x73,

0x3c, 0x2f, 0x61, 0x3e, 0x20, 0x3c, 0x62, 0x3e,

0x7c, 0x3c, 0x2f, 0x62, 0x3e, 0x20, 0x3c, 0x61,

0x20, 0x68, 0x72, 0x65, 0x66, 0x3d, 0x22, 0x74,

0x63, 0x70, 0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c,

0x22, 0x3e, 0x43, 0x6f, 0x6e, 0x6e, 0x65, 0x63,

0x74, 0x69, 0x6f, 0x6e, 0x73, 0x3c, 0x2f, 0x61,

0x3e, 0x20, 0x3c, 0x62, 0x3e, 0x7c, 0x3c, 0x2f,

0x62, 0x3e, 0x20, 0x3c, 0x61, 0x20, 0x68, 0x72,

0x65, 0x66, 0x3d, 0x22, 0x64, 0x69, 0x73, 0x70,

0x6c, 0x61, 0x79, 0x2e, 0x73, 0x68, 0x74, 0x6d,

0x6c, 0x22, 0x3e, 0x44, 0x69, 0x73, 0x70, 0x6c,

0x61, 0x79, 0x3c, 0x2f, 0x61, 0x3e, 0x20, 0x3c,

0x62, 0x3e, 0x7c, 0x3c, 0x2f, 0x62, 0x3e, 0x20,

0x3c, 0x61, 0x20, 0x68, 0x72, 0x65, 0x66, 0x3d,

0x22, 0x63, 0x6f, 0x6d, 0x6d, 0x61, 0x6e, 0x64,

0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c, 0x22, 0x3e,

0x43, 0x6f, 0x6d, 0x6d, 0x61, 0x6e, 0x64, 0x3c,

0x2f, 0x61, 0x3e, 0x20, 0x3c, 0x62, 0x3e, 0x7c,

0x3c, 0x2f, 0x62, 0x3e, 0x20, 0x3c, 0x61, 0x20,

0x68, 0x72, 0x65, 0x66, 0x3d, 0x22, 0x69, 0x6f,

0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c, 0x22, 0x3e,

0x49, 0x4f, 0x3c, 0x2f, 0x61, 0x3e, 0x0d, 0x0a,

0x3c, 0x62, 0x72, 0x3e, 0x3c, 0x70, 0x3e, 0x0d,

0x0a, 0x3c, 0x68, 0x72, 0x3e, 0x0d, 0x0a, 0x3c,

0x62, 0x72, 0x3e, 0x3c, 0x70, 0x3e, 0x0d, 0x0a,

0x3c, 0x68, 0x32, 0x3e, 0x54, 0x61, 0x73, 0x6b,

0x20, 0x73, 0x74, 0x61, 0x74, 0x69, 0x73, 0x74,

0x69, 0x63, 0x73, 0x3c, 0x2f, 0x68, 0x32, 0x3e,

0x0d, 0x0a, 0x50, 0x61, 0x67, 0x65, 0x20, 0x77,

0x69, 0x6c, 0x6c, 0x20, 0x72, 0x65, 0x66, 0x72,

0x65, 0x73, 0x68, 0x20, 0x65, 0x76, 0x65, 0x72,

0x79, 0x20, 0x35, 0x20, 0x73, 0x65, 0x63, 0x6f,

0x6e, 0x64, 0x73, 0x2e, 0x3c, 0x70, 0x3e, 0x0d,

0x0a, 0x3c, 0x66, 0x6f, 0x6e, 0x74, 0x20, 0x66,

0x61, 0x63, 0x65, 0x3d, 0x22, 0x63, 0x6f, 0x75,

0x72, 0x69, 0x65, 0x72, 0x22, 0x3e, 0x3c, 0x70,

0x72, 0x65, 0x3e, 0x54, 0x61, 0x73, 0x6b, 0x20,

0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,

0x20, 0x53, 0x74, 0x61, 0x74, 0x65, 0x20, 0x20,

0x50, 0x72, 0x69, 0x6f, 0x72, 0x69, 0x74, 0x79,

0x20, 0x20, 0x53, 0x74, 0x61, 0x63, 0x6b, 0x09,

0x23, 0x3c, 0x62, 0x72, 0x3e, 0x2a, 0x2a, 0x2a,

0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a,

0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a,

0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a,

0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a,

0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a,

0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x3c, 0x62, 0x72,

0x3e, 0x0d, 0x0a, 0x25, 0x21, 0x20, 0x72, 0x74,

0x6f, 0x73, 0x2d, 0x73, 0x74, 0x61, 0x74, 0x73,

0x0d, 0x0a, 0x3c, 0x2f, 0x70, 0x72, 0x65, 0x3e,

0x3c, 0x2f, 0x66, 0x6f, 0x6e, 0x74, 0x3e, 0x0d,

0x0a, 0x3c, 0x2f, 0x66, 0x6f, 0x6e, 0x74, 0x3e,

0x0d, 0x0a, 0x3c, 0x2f, 0x62, 0x6f, 0x64, 0x79,

0x3e, 0x0d, 0x0a, 0x3c, 0x2f, 0x68, 0x74, 0x6d,

0x6c, 0x3e, 0x0d, 0x0a, 0x0d, 0x0a,

};

static const unsigned char data\_io\_shtml[] =

{

/\* /io.shtml \*/

0x2f, 0x69, 0x6f, 0x2e, 0x73, 0x68, 0x74, 0x6d,

0x6c, 0x00,

0x3c, 0x21, 0x44, 0x4f, 0x43, 0x54, 0x59, 0x50,

0x45, 0x20, 0x48, 0x54, 0x4d, 0x4c, 0x20, 0x50,

0x55, 0x42, 0x4c, 0x49, 0x43, 0x20, 0x22, 0x2d,

0x2f, 0x2f, 0x57, 0x33, 0x43, 0x2f, 0x2f, 0x44,

0x54, 0x44, 0x20, 0x48, 0x54, 0x4d, 0x4c, 0x20,

0x34, 0x2e, 0x30, 0x31, 0x20, 0x54, 0x72, 0x61,

0x6e, 0x73, 0x69, 0x74, 0x69, 0x6f, 0x6e, 0x61,

0x6c, 0x2f, 0x2f, 0x45, 0x4e, 0x22, 0x20, 0x22,

0x68, 0x74, 0x74, 0x70, 0x3a, 0x2f, 0x2f, 0x77,

0x77, 0x77, 0x2e, 0x77, 0x33, 0x2e, 0x6f, 0x72,

0x67, 0x2f, 0x54, 0x52, 0x2f, 0x68, 0x74, 0x6d,

0x6c, 0x34, 0x2f, 0x6c, 0x6f, 0x6f, 0x73, 0x65,

0x2e, 0x64, 0x74, 0x64, 0x22, 0x3e, 0x0d, 0x0a,

0x3c, 0x68, 0x74, 0x6d, 0x6c, 0x3e, 0x0d, 0x0a,

0x20, 0x20, 0x3c, 0x68, 0x65, 0x61, 0x64, 0x3e,

0x0d, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x3c, 0x74,

0x69, 0x74, 0x6c, 0x65, 0x3e, 0x46, 0x72, 0x65,

0x65, 0x52, 0x54, 0x4f, 0x53, 0x2e, 0x6f, 0x72,

0x67, 0x20, 0x75, 0x49, 0x50, 0x20, 0x57, 0x45,

0x42, 0x20, 0x73, 0x65, 0x72, 0x76, 0x65, 0x72,

0x20, 0x64, 0x65, 0x6d, 0x6f, 0x3c, 0x2f, 0x74,

0x69, 0x74, 0x6c, 0x65, 0x3e, 0x0d, 0x0a, 0x20,

0x20, 0x3c, 0x2f, 0x68, 0x65, 0x61, 0x64, 0x3e,

0x0d, 0x0a, 0x20, 0x20, 0x3c, 0x42, 0x4f, 0x44,

0x59, 0x20, 0x62, 0x67, 0x63, 0x6f, 0x6c, 0x6f,

0x72, 0x3d, 0x22, 0x23, 0x43, 0x43, 0x43, 0x43,

0x66, 0x66, 0x22, 0x3e, 0x0d, 0x0a, 0x3c, 0x66,

0x6f, 0x6e, 0x74, 0x20, 0x66, 0x61, 0x63, 0x65,

0x3d, 0x22, 0x61, 0x72, 0x69, 0x61, 0x6c, 0x22,

0x3e, 0x0d, 0x0a, 0x3c, 0x61, 0x20, 0x68, 0x72,

0x65, 0x66, 0x3d, 0x22, 0x69, 0x6e, 0x64, 0x65,

0x78, 0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c, 0x22,

0x3e, 0x52, 0x54, 0x4f, 0x53, 0x20, 0x53, 0x74,

0x61, 0x74, 0x73, 0x3c, 0x2f, 0x61, 0x3e, 0x20,

0x3c, 0x62, 0x3e, 0x7c, 0x3c, 0x2f, 0x62, 0x3e,

0x20, 0x3c, 0x61, 0x20, 0x68, 0x72, 0x65, 0x66,

0x3d, 0x22, 0x73, 0x74, 0x61, 0x74, 0x73, 0x2e,

0x73, 0x68, 0x74, 0x6d, 0x6c, 0x22, 0x3e, 0x54,

0x43, 0x50, 0x20, 0x53, 0x74, 0x61, 0x74, 0x73,

0x3c, 0x2f, 0x61, 0x3e, 0x20, 0x3c, 0x62, 0x3e,

0x7c, 0x3c, 0x2f, 0x62, 0x3e, 0x20, 0x3c, 0x61,

0x20, 0x68, 0x72, 0x65, 0x66, 0x3d, 0x22, 0x74,

0x63, 0x70, 0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c,

0x22, 0x3e, 0x43, 0x6f, 0x6e, 0x6e, 0x65, 0x63,

0x74, 0x69, 0x6f, 0x6e, 0x73, 0x3c, 0x2f, 0x61,

0x3e, 0x20, 0x3c, 0x62, 0x3e, 0x7c, 0x3c, 0x2f,

0x62, 0x3e, 0x20, 0x3c, 0x61, 0x20, 0x68, 0x72,

0x65, 0x66, 0x3d, 0x22, 0x64, 0x69, 0x73, 0x70,

0x6c, 0x61, 0x79, 0x2e, 0x73, 0x68, 0x74, 0x6d,

0x6c, 0x22, 0x3e, 0x44, 0x69, 0x73, 0x70, 0x6c,

0x61, 0x79, 0x3c, 0x2f, 0x61, 0x3e, 0x20, 0x3c,

0x62, 0x3e, 0x7c, 0x3c, 0x2f, 0x62, 0x3e, 0x20,

0x3c, 0x61, 0x20, 0x68, 0x72, 0x65, 0x66, 0x3d,

0x22, 0x63, 0x6f, 0x6d, 0x6d, 0x61, 0x6e, 0x64,

0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c, 0x22, 0x3e,

0x43, 0x6f, 0x6d, 0x6d, 0x61, 0x6e, 0x64, 0x3c,

0x2f, 0x61, 0x3e, 0x20, 0x3c, 0x62, 0x3e, 0x7c,

0x3c, 0x2f, 0x62, 0x3e, 0x20, 0x3c, 0x61, 0x20,

0x68, 0x72, 0x65, 0x66, 0x3d, 0x22, 0x69, 0x6f,

0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c, 0x22, 0x3e,

0x49, 0x4f, 0x3c, 0x2f, 0x61, 0x3e, 0x0d, 0x0a,

0x3c, 0x62, 0x72, 0x3e, 0x3c, 0x70, 0x3e, 0x0d,

0x0a, 0x3c, 0x68, 0x72, 0x3e, 0x0d, 0x0a, 0x3c,

0x62, 0x3e, 0x4c, 0x45, 0x44, 0x20, 0x61, 0x6e,

0x64, 0x20, 0x4c, 0x43, 0x44, 0x20, 0x49, 0x4f,

0x3c, 0x2f, 0x62, 0x3e, 0x3c, 0x62, 0x72, 0x3e,

0x0d, 0x0a, 0x0d, 0x0a, 0x3c, 0x70, 0x3e, 0x0d,

0x0a, 0x0d, 0x0a, 0x55, 0x73, 0x65, 0x20, 0x74,

0x68, 0x65, 0x20, 0x63, 0x68, 0x65, 0x63, 0x6b,

0x20, 0x62, 0x6f, 0x78, 0x20, 0x74, 0x6f, 0x20,

0x74, 0x75, 0x72, 0x6e, 0x20, 0x6f, 0x6e, 0x20,

0x6f, 0x72, 0x20, 0x6f, 0x66, 0x66, 0x20, 0x74,

0x68, 0x65, 0x20, 0x4c, 0x45, 0x44, 0x2c, 0x20,

0x65, 0x6e, 0x74, 0x65, 0x72, 0x20, 0x74, 0x65,

0x78, 0x74, 0x20, 0x74, 0x6f, 0x20, 0x64, 0x69,

0x73, 0x70, 0x6c, 0x61, 0x79, 0x20, 0x6f, 0x6e,

0x20, 0x74, 0x68, 0x65, 0x20, 0x4f, 0x4c, 0x45,

0x44, 0x20, 0x64, 0x69, 0x73, 0x70, 0x6c, 0x61,

0x79, 0x2c, 0x20, 0x74, 0x68, 0x65, 0x6e, 0x20,

0x63, 0x6c, 0x69, 0x63, 0x6b, 0x20, 0x22, 0x55,

0x70, 0x64, 0x61, 0x74, 0x65, 0x20, 0x49, 0x4f,

0x22, 0x2e, 0x0d, 0x0a, 0x0d, 0x0a, 0x0d, 0x0a,

0x3c, 0x70, 0x3e, 0x0d, 0x0a, 0x3c, 0x66, 0x6f,

0x72, 0x6d, 0x20, 0x6e, 0x61, 0x6d, 0x65, 0x3d,

0x22, 0x61, 0x46, 0x6f, 0x72, 0x6d, 0x22, 0x20,

0x61, 0x63, 0x74, 0x69, 0x6f, 0x6e, 0x3d, 0x22,

0x2f, 0x69, 0x6f, 0x2e, 0x73, 0x68, 0x74, 0x6d,

0x6c, 0x22, 0x20, 0x6d, 0x65, 0x74, 0x68, 0x6f,

0x64, 0x3d, 0x22, 0x67, 0x65, 0x74, 0x22, 0x3e,

0x0d, 0x0a, 0x25, 0x21, 0x20, 0x6c, 0x65, 0x64,

0x2d, 0x69, 0x6f, 0x0d, 0x0a, 0x3c, 0x70, 0x3e,

0x0d, 0x0a, 0x3c, 0x69, 0x6e, 0x70, 0x75, 0x74,

0x20, 0x74, 0x79, 0x70, 0x65, 0x3d, 0x22, 0x73,

0x75, 0x62, 0x6d, 0x69, 0x74, 0x22, 0x20, 0x76,

0x61, 0x6c, 0x75, 0x65, 0x3d, 0x22, 0x55, 0x70,

0x64, 0x61, 0x74, 0x65, 0x20, 0x49, 0x4f, 0x22,

0x3e, 0x0d, 0x0a, 0x3c, 0x2f, 0x66, 0x6f, 0x72,

0x6d, 0x3e, 0x0d, 0x0a, 0x3c, 0x62, 0x72, 0x3e,

0x3c, 0x70, 0x3e, 0x0d, 0x0a, 0x3c, 0x2f, 0x66,

0x6f, 0x6e, 0x74, 0x3e, 0x0d, 0x0a, 0x3c, 0x2f,

0x62, 0x6f, 0x64, 0x79, 0x3e, 0x0d, 0x0a, 0x3c,

0x2f, 0x68, 0x74, 0x6d, 0x6c, 0x3e, 0x0d, 0x0a,

0x0d, 0x0a,

};

static const unsigned char data\_stats\_shtml[] =

{

/\* /stats.shtml \*/

0x2f, 0x73, 0x74, 0x61, 0x74, 0x73, 0x2e, 0x73,

0x68, 0x74, 0x6d, 0x6c, 0x00,

0x3c, 0x21, 0x44, 0x4f, 0x43, 0x54, 0x59, 0x50,

0x45, 0x20, 0x48, 0x54, 0x4d, 0x4c, 0x20, 0x50,

0x55, 0x42, 0x4c, 0x49, 0x43, 0x20, 0x22, 0x2d,

0x2f, 0x2f, 0x57, 0x33, 0x43, 0x2f, 0x2f, 0x44,

0x54, 0x44, 0x20, 0x48, 0x54, 0x4d, 0x4c, 0x20,

0x34, 0x2e, 0x30, 0x31, 0x20, 0x54, 0x72, 0x61,

0x6e, 0x73, 0x69, 0x74, 0x69, 0x6f, 0x6e, 0x61,

0x6c, 0x2f, 0x2f, 0x45, 0x4e, 0x22, 0x20, 0x22,

0x68, 0x74, 0x74, 0x70, 0x3a, 0x2f, 0x2f, 0x77,

0x77, 0x77, 0x2e, 0x77, 0x33, 0x2e, 0x6f, 0x72,

0x67, 0x2f, 0x54, 0x52, 0x2f, 0x68, 0x74, 0x6d,

0x6c, 0x34, 0x2f, 0x6c, 0x6f, 0x6f, 0x73, 0x65,

0x2e, 0x64, 0x74, 0x64, 0x22, 0x3e, 0x0d, 0x0a,

0x3c, 0x68, 0x74, 0x6d, 0x6c, 0x3e, 0x0d, 0x0a,

0x20, 0x20, 0x3c, 0x68, 0x65, 0x61, 0x64, 0x3e,

0x0d, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x3c, 0x74,

0x69, 0x74, 0x6c, 0x65, 0x3e, 0x46, 0x72, 0x65,

0x65, 0x52, 0x54, 0x4f, 0x53, 0x2e, 0x6f, 0x72,

0x67, 0x20, 0x75, 0x49, 0x50, 0x20, 0x57, 0x45,

0x42, 0x20, 0x73, 0x65, 0x72, 0x76, 0x65, 0x72,

0x20, 0x64, 0x65, 0x6d, 0x6f, 0x3c, 0x2f, 0x74,

0x69, 0x74, 0x6c, 0x65, 0x3e, 0x0d, 0x0a, 0x20,

0x20, 0x3c, 0x2f, 0x68, 0x65, 0x61, 0x64, 0x3e,

0x0d, 0x0a, 0x20, 0x20, 0x3c, 0x42, 0x4f, 0x44,

0x59, 0x20, 0x62, 0x67, 0x63, 0x6f, 0x6c, 0x6f,

0x72, 0x3d, 0x22, 0x23, 0x43, 0x43, 0x43, 0x43,

0x66, 0x66, 0x22, 0x3e, 0x0d, 0x0a, 0x3c, 0x66,

0x6f, 0x6e, 0x74, 0x20, 0x66, 0x61, 0x63, 0x65,

0x3d, 0x22, 0x61, 0x72, 0x69, 0x61, 0x6c, 0x22,

0x3e, 0x0d, 0x0a, 0x3c, 0x61, 0x20, 0x68, 0x72,

0x65, 0x66, 0x3d, 0x22, 0x69, 0x6e, 0x64, 0x65,

0x78, 0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c, 0x22,

0x3e, 0x52, 0x54, 0x4f, 0x53, 0x20, 0x53, 0x74,

0x61, 0x74, 0x73, 0x3c, 0x2f, 0x61, 0x3e, 0x20,

0x3c, 0x62, 0x3e, 0x7c, 0x3c, 0x2f, 0x62, 0x3e,

0x20, 0x3c, 0x61, 0x20, 0x68, 0x72, 0x65, 0x66,

0x3d, 0x22, 0x73, 0x74, 0x61, 0x74, 0x73, 0x2e,

0x73, 0x68, 0x74, 0x6d, 0x6c, 0x22, 0x3e, 0x54,

0x43, 0x50, 0x20, 0x53, 0x74, 0x61, 0x74, 0x73,

0x3c, 0x2f, 0x61, 0x3e, 0x20, 0x3c, 0x62, 0x3e,

0x7c, 0x3c, 0x2f, 0x62, 0x3e, 0x20, 0x3c, 0x61,

0x20, 0x68, 0x72, 0x65, 0x66, 0x3d, 0x22, 0x74,

0x63, 0x70, 0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c,

0x22, 0x3e, 0x43, 0x6f, 0x6e, 0x6e, 0x65, 0x63,

0x74, 0x69, 0x6f, 0x6e, 0x73, 0x3c, 0x2f, 0x61,

0x3e, 0x20, 0x3c, 0x62, 0x3e, 0x7c, 0x3c, 0x2f,

0x62, 0x3e, 0x20, 0x3c, 0x61, 0x20, 0x68, 0x72,

0x65, 0x66, 0x3d, 0x22, 0x64, 0x69, 0x73, 0x70,

0x6c, 0x61, 0x79, 0x2e, 0x73, 0x68, 0x74, 0x6d,

0x6c, 0x22, 0x3e, 0x44, 0x69, 0x73, 0x70, 0x6c,

0x61, 0x79, 0x3c, 0x2f, 0x61, 0x3e, 0x20, 0x3c,

0x62, 0x3e, 0x7c, 0x3c, 0x2f, 0x62, 0x3e, 0x20,

0x3c, 0x61, 0x20, 0x68, 0x72, 0x65, 0x66, 0x3d,

0x22, 0x63, 0x6f, 0x6d, 0x6d, 0x61, 0x6e, 0x64,

0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c, 0x22, 0x3e,

0x43, 0x6f, 0x6d, 0x6d, 0x61, 0x6e, 0x64, 0x3c,

0x2f, 0x61, 0x3e, 0x20, 0x3c, 0x62, 0x3e, 0x7c,

0x3c, 0x2f, 0x62, 0x3e, 0x20, 0x3c, 0x61, 0x20,

0x68, 0x72, 0x65, 0x66, 0x3d, 0x22, 0x69, 0x6f,

0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c, 0x22, 0x3e,

0x49, 0x4f, 0x3c, 0x2f, 0x61, 0x3e, 0x0d, 0x0a,

0x3c, 0x62, 0x72, 0x3e, 0x3c, 0x70, 0x3e, 0x0d,

0x0a, 0x3c, 0x68, 0x72, 0x3e, 0x0d, 0x0a, 0x3c,

0x62, 0x72, 0x3e, 0x3c, 0x70, 0x3e, 0x0d, 0x0a,

0x3c, 0x68, 0x32, 0x3e, 0x4e, 0x65, 0x74, 0x77,

0x6f, 0x72, 0x6b, 0x20, 0x73, 0x74, 0x61, 0x74,

0x69, 0x73, 0x74, 0x69, 0x63, 0x73, 0x3c, 0x2f,

0x68, 0x32, 0x3e, 0x0d, 0x0a, 0x3c, 0x74, 0x61,

0x62, 0x6c, 0x65, 0x20, 0x77, 0x69, 0x64, 0x74,

0x68, 0x3d, 0x22, 0x33, 0x30, 0x30, 0x22, 0x20,

0x62, 0x6f, 0x72, 0x64, 0x65, 0x72, 0x3d, 0x22,

0x30, 0x22, 0x3e, 0x0d, 0x0a, 0x3c, 0x74, 0x72,

0x3e, 0x3c, 0x74, 0x64, 0x20, 0x61, 0x6c, 0x69,

0x67, 0x6e, 0x3d, 0x22, 0x6c, 0x65, 0x66, 0x74,

0x22, 0x3e, 0x3c, 0x66, 0x6f, 0x6e, 0x74, 0x20,

0x66, 0x61, 0x63, 0x65, 0x3d, 0x22, 0x63, 0x6f,

0x75, 0x72, 0x69, 0x65, 0x72, 0x22, 0x3e, 0x3c,

0x70, 0x72, 0x65, 0x3e, 0x0d, 0x0a, 0x49, 0x50,

0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,

0x20, 0x20, 0x20, 0x50, 0x61, 0x63, 0x6b, 0x65,

0x74, 0x73, 0x20, 0x64, 0x72, 0x6f, 0x70, 0x70,

0x65, 0x64, 0x0d, 0x0a, 0x20, 0x20, 0x20, 0x20,

0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,

0x20, 0x50, 0x61, 0x63, 0x6b, 0x65, 0x74, 0x73,

0x20, 0x72, 0x65, 0x63, 0x65, 0x69, 0x76, 0x65,

0x64, 0x0d, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x20,

0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,

0x50, 0x61, 0x63, 0x6b, 0x65, 0x74, 0x73, 0x20,

0x73, 0x65, 0x6e, 0x74, 0x0d, 0x0a, 0x49, 0x50,

0x20, 0x65, 0x72, 0x72, 0x6f, 0x72, 0x73, 0x20,

0x20, 0x20, 0x20, 0x49, 0x50, 0x20, 0x76, 0x65,

0x72, 0x73, 0x69, 0x6f, 0x6e, 0x2f, 0x68, 0x65,

0x61, 0x64, 0x65, 0x72, 0x20, 0x6c, 0x65, 0x6e,

0x67, 0x74, 0x68, 0x0d, 0x0a, 0x20, 0x20, 0x20,

0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,

0x20, 0x20, 0x49, 0x50, 0x20, 0x6c, 0x65, 0x6e,

0x67, 0x74, 0x68, 0x2c, 0x20, 0x68, 0x69, 0x67,

0x68, 0x20, 0x62, 0x79, 0x74, 0x65, 0x0d, 0x0a,

0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,

0x20, 0x20, 0x20, 0x20, 0x20, 0x49, 0x50, 0x20,

0x6c, 0x65, 0x6e, 0x67, 0x74, 0x68, 0x2c, 0x20,

0x6c, 0x6f, 0x77, 0x20, 0x62, 0x79, 0x74, 0x65,

0x0d, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,

0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x49,

0x50, 0x20, 0x66, 0x72, 0x61, 0x67, 0x6d, 0x65,

0x6e, 0x74, 0x73, 0x0d, 0x0a, 0x20, 0x20, 0x20,

0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,

0x20, 0x20, 0x48, 0x65, 0x61, 0x64, 0x65, 0x72,

0x20, 0x63, 0x68, 0x65, 0x63, 0x6b, 0x73, 0x75,

0x6d, 0x0d, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x20,

0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,

0x57, 0x72, 0x6f, 0x6e, 0x67, 0x20, 0x70, 0x72,

0x6f, 0x74, 0x6f, 0x63, 0x6f, 0x6c, 0x0d, 0x0a,

0x49, 0x43, 0x4d, 0x50, 0x09, 0x20, 0x20, 0x20,

0x20, 0x20, 0x50, 0x61, 0x63, 0x6b, 0x65, 0x74,

0x73, 0x20, 0x64, 0x72, 0x6f, 0x70, 0x70, 0x65,

0x64, 0x0d, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x20,

0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,

0x50, 0x61, 0x63, 0x6b, 0x65, 0x74, 0x73, 0x20,

0x72, 0x65, 0x63, 0x65, 0x69, 0x76, 0x65, 0x64,

0x0d, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,

0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x50,

0x61, 0x63, 0x6b, 0x65, 0x74, 0x73, 0x20, 0x73,

0x65, 0x6e, 0x74, 0x0d, 0x0a, 0x20, 0x20, 0x20,

0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,

0x20, 0x20, 0x54, 0x79, 0x70, 0x65, 0x20, 0x65,

0x72, 0x72, 0x6f, 0x72, 0x73, 0x0d, 0x0a, 0x54,

0x43, 0x50, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,

0x20, 0x20, 0x20, 0x20, 0x50, 0x61, 0x63, 0x6b,

0x65, 0x74, 0x73, 0x20, 0x64, 0x72, 0x6f, 0x70,

0x70, 0x65, 0x64, 0x0d, 0x0a, 0x20, 0x20, 0x20,

0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,

0x20, 0x20, 0x50, 0x61, 0x63, 0x6b, 0x65, 0x74,

0x73, 0x20, 0x72, 0x65, 0x63, 0x65, 0x69, 0x76,

0x65, 0x64, 0x0d, 0x0a, 0x20, 0x20, 0x20, 0x20,

0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,

0x20, 0x50, 0x61, 0x63, 0x6b, 0x65, 0x74, 0x73,

0x20, 0x73, 0x65, 0x6e, 0x74, 0x0d, 0x0a, 0x20,

0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,

0x20, 0x20, 0x20, 0x20, 0x43, 0x68, 0x65, 0x63,

0x6b, 0x73, 0x75, 0x6d, 0x20, 0x65, 0x72, 0x72,

0x6f, 0x72, 0x73, 0x0d, 0x0a, 0x20, 0x20, 0x20,

0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,

0x20, 0x20, 0x44, 0x61, 0x74, 0x61, 0x20, 0x70,

0x61, 0x63, 0x6b, 0x65, 0x74, 0x73, 0x20, 0x77,

0x69, 0x74, 0x68, 0x6f, 0x75, 0x74, 0x20, 0x41,

0x43, 0x4b, 0x73, 0x0d, 0x0a, 0x20, 0x20, 0x20,

0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,

0x20, 0x20, 0x52, 0x65, 0x73, 0x65, 0x74, 0x73,

0x0d, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,

0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x52,

0x65, 0x74, 0x72, 0x61, 0x6e, 0x73, 0x6d, 0x69,

0x73, 0x73, 0x69, 0x6f, 0x6e, 0x73, 0x0d, 0x0a,

0x09, 0x20, 0x20, 0x20, 0x20, 0x20, 0x4e, 0x6f,

0x20, 0x63, 0x6f, 0x6e, 0x6e, 0x65, 0x63, 0x74,

0x69, 0x6f, 0x6e, 0x20, 0x61, 0x76, 0x61, 0x6c,

0x69, 0x61, 0x62, 0x6c, 0x65, 0x0d, 0x0a, 0x09,

0x20, 0x20, 0x20, 0x20, 0x20, 0x43, 0x6f, 0x6e,

0x6e, 0x65, 0x63, 0x74, 0x69, 0x6f, 0x6e, 0x20,

0x61, 0x74, 0x74, 0x65, 0x6d, 0x70, 0x74, 0x73,

0x20, 0x74, 0x6f, 0x20, 0x63, 0x6c, 0x6f, 0x73,

0x65, 0x64, 0x20, 0x70, 0x6f, 0x72, 0x74, 0x73,

0x0d, 0x0a, 0x3c, 0x2f, 0x70, 0x72, 0x65, 0x3e,

0x3c, 0x2f, 0x66, 0x6f, 0x6e, 0x74, 0x3e, 0x3c,

0x2f, 0x74, 0x64, 0x3e, 0x3c, 0x74, 0x64, 0x3e,

0x3c, 0x70, 0x72, 0x65, 0x3e, 0x25, 0x21, 0x20,

0x6e, 0x65, 0x74, 0x2d, 0x73, 0x74, 0x61, 0x74,

0x73, 0x0d, 0x0a, 0x3c, 0x2f, 0x70, 0x72, 0x65,

0x3e, 0x3c, 0x2f, 0x74, 0x61, 0x62, 0x6c, 0x65,

0x3e, 0x0d, 0x0a, 0x3c, 0x2f, 0x66, 0x6f, 0x6e,

0x74, 0x3e, 0x0d, 0x0a, 0x3c, 0x2f, 0x62, 0x6f,

0x64, 0x79, 0x3e, 0x0d, 0x0a, 0x3c, 0x2f, 0x68,

0x74, 0x6d, 0x6c, 0x3e, 0x0d, 0x0a,

};

static const unsigned char data\_tcp\_shtml[] =

{

/\* /tcp.shtml \*/

0x2f, 0x74, 0x63, 0x70, 0x2e, 0x73, 0x68, 0x74,

0x6d, 0x6c, 0x00,

0x3c, 0x21, 0x44, 0x4f, 0x43, 0x54, 0x59, 0x50,

0x45, 0x20, 0x48, 0x54, 0x4d, 0x4c, 0x20, 0x50,

0x55, 0x42, 0x4c, 0x49, 0x43, 0x20, 0x22, 0x2d,

0x2f, 0x2f, 0x57, 0x33, 0x43, 0x2f, 0x2f, 0x44,

0x54, 0x44, 0x20, 0x48, 0x54, 0x4d, 0x4c, 0x20,

0x34, 0x2e, 0x30, 0x31, 0x20, 0x54, 0x72, 0x61,

0x6e, 0x73, 0x69, 0x74, 0x69, 0x6f, 0x6e, 0x61,

0x6c, 0x2f, 0x2f, 0x45, 0x4e, 0x22, 0x20, 0x22,

0x68, 0x74, 0x74, 0x70, 0x3a, 0x2f, 0x2f, 0x77,

0x77, 0x77, 0x2e, 0x77, 0x33, 0x2e, 0x6f, 0x72,

0x67, 0x2f, 0x54, 0x52, 0x2f, 0x68, 0x74, 0x6d,

0x6c, 0x34, 0x2f, 0x6c, 0x6f, 0x6f, 0x73, 0x65,

0x2e, 0x64, 0x74, 0x64, 0x22, 0x3e, 0x0d, 0x0a,

0x3c, 0x68, 0x74, 0x6d, 0x6c, 0x3e, 0x0d, 0x0a,

0x20, 0x20, 0x3c, 0x68, 0x65, 0x61, 0x64, 0x3e,

0x0d, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x3c, 0x74,

0x69, 0x74, 0x6c, 0x65, 0x3e, 0x46, 0x72, 0x65,

0x65, 0x52, 0x54, 0x4f, 0x53, 0x2e, 0x6f, 0x72,

0x67, 0x20, 0x75, 0x49, 0x50, 0x20, 0x57, 0x45,

0x42, 0x20, 0x73, 0x65, 0x72, 0x76, 0x65, 0x72,

0x20, 0x64, 0x65, 0x6d, 0x6f, 0x3c, 0x2f, 0x74,

0x69, 0x74, 0x6c, 0x65, 0x3e, 0x0d, 0x0a, 0x20,

0x20, 0x3c, 0x2f, 0x68, 0x65, 0x61, 0x64, 0x3e,

0x0d, 0x0a, 0x20, 0x20, 0x3c, 0x42, 0x4f, 0x44,

0x59, 0x20, 0x62, 0x67, 0x63, 0x6f, 0x6c, 0x6f,

0x72, 0x3d, 0x22, 0x23, 0x43, 0x43, 0x43, 0x43,

0x66, 0x66, 0x22, 0x3e, 0x0d, 0x0a, 0x3c, 0x66,

0x6f, 0x6e, 0x74, 0x20, 0x66, 0x61, 0x63, 0x65,

0x3d, 0x22, 0x61, 0x72, 0x69, 0x61, 0x6c, 0x22,

0x3e, 0x0d, 0x0a, 0x3c, 0x61, 0x20, 0x68, 0x72,

0x65, 0x66, 0x3d, 0x22, 0x69, 0x6e, 0x64, 0x65,

0x78, 0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c, 0x22,

0x3e, 0x52, 0x54, 0x4f, 0x53, 0x20, 0x53, 0x74,

0x61, 0x74, 0x73, 0x3c, 0x2f, 0x61, 0x3e, 0x20,

0x3c, 0x62, 0x3e, 0x7c, 0x3c, 0x2f, 0x62, 0x3e,

0x20, 0x3c, 0x61, 0x20, 0x68, 0x72, 0x65, 0x66,

0x3d, 0x22, 0x73, 0x74, 0x61, 0x74, 0x73, 0x2e,

0x73, 0x68, 0x74, 0x6d, 0x6c, 0x22, 0x3e, 0x54,

0x43, 0x50, 0x20, 0x53, 0x74, 0x61, 0x74, 0x73,

0x3c, 0x2f, 0x61, 0x3e, 0x20, 0x3c, 0x62, 0x3e,

0x7c, 0x3c, 0x2f, 0x62, 0x3e, 0x20, 0x3c, 0x61,

0x20, 0x68, 0x72, 0x65, 0x66, 0x3d, 0x22, 0x74,

0x63, 0x70, 0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c,

0x22, 0x3e, 0x43, 0x6f, 0x6e, 0x6e, 0x65, 0x63,

0x74, 0x69, 0x6f, 0x6e, 0x73, 0x3c, 0x2f, 0x61,

0x3e, 0x20, 0x3c, 0x62, 0x3e, 0x7c, 0x3c, 0x2f,

0x62, 0x3e, 0x20, 0x3c, 0x61, 0x20, 0x68, 0x72,

0x65, 0x66, 0x3d, 0x22, 0x64, 0x69, 0x73, 0x70,

0x6c, 0x61, 0x79, 0x2e, 0x73, 0x68, 0x74, 0x6d,

0x6c, 0x22, 0x3e, 0x44, 0x69, 0x73, 0x70, 0x6c,

0x61, 0x79, 0x3c, 0x2f, 0x61, 0x3e, 0x20, 0x3c,

0x62, 0x3e, 0x7c, 0x3c, 0x2f, 0x62, 0x3e, 0x20,

0x3c, 0x61, 0x20, 0x68, 0x72, 0x65, 0x66, 0x3d,

0x22, 0x63, 0x6f, 0x6d, 0x6d, 0x61, 0x6e, 0x64,

0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c, 0x22, 0x3e,

0x43, 0x6f, 0x6d, 0x6d, 0x61, 0x6e, 0x64, 0x3c,

0x2f, 0x61, 0x3e, 0x20, 0x3c, 0x62, 0x3e, 0x7c,

0x3c, 0x2f, 0x62, 0x3e, 0x20, 0x3c, 0x61, 0x20,

0x68, 0x72, 0x65, 0x66, 0x3d, 0x22, 0x69, 0x6f,

0x2e, 0x73, 0x68, 0x74, 0x6d, 0x6c, 0x22, 0x3e,

0x49, 0x4f, 0x3c, 0x2f, 0x61, 0x3e, 0x0d, 0x0a,

0x3c, 0x62, 0x72, 0x3e, 0x3c, 0x70, 0x3e, 0x0d,

0x0a, 0x3c, 0x68, 0x72, 0x3e, 0x0d, 0x0a, 0x3c,

0x62, 0x72, 0x3e, 0x0d, 0x0a, 0x3c, 0x68, 0x32,

0x3e, 0x4e, 0x65, 0x74, 0x77, 0x6f, 0x72, 0x6b,

0x20, 0x63, 0x6f, 0x6e, 0x6e, 0x65, 0x63, 0x74,

0x69, 0x6f, 0x6e, 0x73, 0x3c, 0x2f, 0x68, 0x32,

0x3e, 0x0d, 0x0a, 0x3c, 0x70, 0x3e, 0x0d, 0x0a,

0x3c, 0x74, 0x61, 0x62, 0x6c, 0x65, 0x3e, 0x0d,

0x0a, 0x3c, 0x74, 0x72, 0x3e, 0x3c, 0x74, 0x68,

0x3e, 0x4c, 0x6f, 0x63, 0x61, 0x6c, 0x3c, 0x2f,

0x74, 0x68, 0x3e, 0x3c, 0x74, 0x68, 0x3e, 0x52,

0x65, 0x6d, 0x6f, 0x74, 0x65, 0x3c, 0x2f, 0x74,

0x68, 0x3e, 0x3c, 0x74, 0x68, 0x3e, 0x53, 0x74,

0x61, 0x74, 0x65, 0x3c, 0x2f, 0x74, 0x68, 0x3e,

0x3c, 0x74, 0x68, 0x3e, 0x52, 0x65, 0x74, 0x72,

0x61, 0x6e, 0x73, 0x6d, 0x69, 0x73, 0x73, 0x69,

0x6f, 0x6e, 0x73, 0x3c, 0x2f, 0x74, 0x68, 0x3e,

0x3c, 0x74, 0x68, 0x3e, 0x54, 0x69, 0x6d, 0x65,

0x72, 0x3c, 0x2f, 0x74, 0x68, 0x3e, 0x3c, 0x74,

0x68, 0x3e, 0x46, 0x6c, 0x61, 0x67, 0x73, 0x3c,

0x2f, 0x74, 0x68, 0x3e, 0x3c, 0x2f, 0x74, 0x72,

0x3e, 0x0d, 0x0a, 0x25, 0x21, 0x20, 0x74, 0x63,

0x70, 0x2d, 0x63, 0x6f, 0x6e, 0x6e, 0x65, 0x63,

0x74, 0x69, 0x6f, 0x6e, 0x73, 0x0d, 0x0a, 0x3c,

0x2f, 0x70, 0x72, 0x65, 0x3e, 0x3c, 0x2f, 0x66,

0x6f, 0x6e, 0x74, 0x3e, 0x0d, 0x0a, 0x3c, 0x2f,

0x66, 0x6f, 0x6e, 0x74, 0x3e, 0x0d, 0x0a, 0x3c,

0x2f, 0x62, 0x6f, 0x64, 0x79, 0x3e, 0x0d, 0x0a,

0x3c, 0x2f, 0x68, 0x74, 0x6d, 0x6c, 0x3e, 0x0d,

0x0a, 0x0d, 0x0a,

};

const struct httpd\_fsdata\_file file\_tcp\_shtml[] =

{

{

NULL,

data\_tcp\_shtml,

data\_tcp\_shtml + 11,

sizeof(data\_tcp\_shtml) - 11

}

};

const struct httpd\_fsdata\_file file\_stats\_shtml[] =

{

{

file\_tcp\_shtml,

data\_stats\_shtml,

data\_stats\_shtml + 13,

sizeof(data\_stats\_shtml) - 13

}

};

const struct httpd\_fsdata\_file file\_io\_shtml[] =

{

{

file\_stats\_shtml,

data\_io\_shtml,

data\_io\_shtml + 10,

sizeof(data\_io\_shtml) - 10

}

};

const struct httpd\_fsdata\_file file\_index\_shtml[] =

{

{

file\_io\_shtml,

data\_index\_shtml,

data\_index\_shtml + 13,

sizeof(data\_index\_shtml) - 13

}

};

const struct httpd\_fsdata\_file file\_index\_html[] =

{

{

file\_index\_shtml,

data\_index\_html,

data\_index\_html + 12,

sizeof(data\_index\_html) - 12

}

};

const struct httpd\_fsdata\_file file\_display\_shtml[] =

{

{

file\_index\_html,

data\_display\_shtml,

data\_display\_shtml + 15,

sizeof(data\_display\_shtml) - 15

}

};

const struct httpd\_fsdata\_file file\_command\_shtml[] =

{

{

file\_display\_shtml,

data\_command\_shtml,

data\_command\_shtml + 15,

sizeof(data\_command\_shtml) - 15

}

};

const struct httpd\_fsdata\_file file\_404\_html[] =

{

{

file\_command\_shtml,

data\_404\_html,

data\_404\_html + 10,

sizeof(data\_404\_html) - 10

}

};

#define HTTPD\_FS\_ROOT file\_404\_html

#define HTTPD\_FS\_NUMFILES 8

/\*

Xinyu Sui

httpd-cgi.c

\*/

/\*\*

\* \addtogroup httpd

\* @{

\*/

/\*\*

\* \file

\* Web server script interface

\* \author

\* Adam Dunkels <adam@sics.se>

\*

\*/

/\*

\* Copyright (c) 2001-2006, Adam Dunkels.

\* All rights reserved.

\* This file is part of the uIP TCP/IP stack.

\*

\* $Id: httpd-cgi.c,v 1.2 2006/06/11 21:46:37 adam Exp $

\*

\*/

#include "uip.h"

#include "psock.h"

#include "httpd.h"

#include "httpd-cgi.h"

#include "httpd-fs.h"

#include "myHeader.h"

#include <stdio.h>

#include <string.h>

HTTPD\_CGI\_CALL(file, "file-stats", file\_stats);

HTTPD\_CGI\_CALL(tcp, "tcp-connections", tcp\_stats);

HTTPD\_CGI\_CALL(net, "net-stats", net\_stats);

HTTPD\_CGI\_CALL(rtos, "rtos-stats", rtos\_stats );

HTTPD\_CGI\_CALL(io, "led-io", led\_io );

HTTPD\_CGI\_CALL(communic, "communic-out", communic\_out);

HTTPD\_CGI\_CALL(command, "command-in", command\_in);

static const struct httpd\_cgi\_call \*calls[] = { &file, &tcp, &net, &rtos, &io, &communic, &command, NULL };

/\*---------------------------------------------------------------------------\*/

static

PT\_THREAD(nullfunction(struct httpd\_state \*s, char \*ptr))

{

PSOCK\_BEGIN(&s->sout);

PSOCK\_END(&s->sout);

}

/\*---------------------------------------------------------------------------\*/

httpd\_cgifunction

httpd\_cgi(char \*name)

{

const struct httpd\_cgi\_call \*\*f;

/\* Find the matching name in the table, return the function. \*/

for(f = calls; \*f != NULL; ++f) {

if(strncmp((\*f)->name, name, strlen((\*f)->name)) == 0) {

return (\*f)->function;

}

}

return nullfunction;

}

/\*---------------------------------------------------------------------------\*/

static unsigned short

generate\_file\_stats(void \*arg)

{

char \*f = (char \*)arg;

return snprintf((char \*)uip\_appdata, UIP\_APPDATA\_SIZE, "%5u", httpd\_fs\_count(f));

}

/\*---------------------------------------------------------------------------\*/

static

PT\_THREAD(file\_stats(struct httpd\_state \*s, char \*ptr))

{

PSOCK\_BEGIN(&s->sout);

PSOCK\_GENERATOR\_SEND(&s->sout, generate\_file\_stats, strchr(ptr, ' ') + 1);

PSOCK\_END(&s->sout);

}

/\*---------------------------------------------------------------------------\*/

static const char closed[] = /\* "CLOSED",\*/

{0x43, 0x4c, 0x4f, 0x53, 0x45, 0x44, 0};

static const char syn\_rcvd[] = /\* "SYN-RCVD",\*/

{0x53, 0x59, 0x4e, 0x2d, 0x52, 0x43, 0x56,

0x44, 0};

static const char syn\_sent[] = /\* "SYN-SENT",\*/

{0x53, 0x59, 0x4e, 0x2d, 0x53, 0x45, 0x4e,

0x54, 0};

static const char established[] = /\* "ESTABLISHED",\*/

{0x45, 0x53, 0x54, 0x41, 0x42, 0x4c, 0x49, 0x53, 0x48,

0x45, 0x44, 0};

static const char fin\_wait\_1[] = /\* "FIN-WAIT-1",\*/

{0x46, 0x49, 0x4e, 0x2d, 0x57, 0x41, 0x49,

0x54, 0x2d, 0x31, 0};

static const char fin\_wait\_2[] = /\* "FIN-WAIT-2",\*/

{0x46, 0x49, 0x4e, 0x2d, 0x57, 0x41, 0x49,

0x54, 0x2d, 0x32, 0};

static const char closing[] = /\* "CLOSING",\*/

{0x43, 0x4c, 0x4f, 0x53, 0x49,

0x4e, 0x47, 0};

static const char time\_wait[] = /\* "TIME-WAIT,"\*/

{0x54, 0x49, 0x4d, 0x45, 0x2d, 0x57, 0x41,

0x49, 0x54, 0};

static const char last\_ack[] = /\* "LAST-ACK"\*/

{0x4c, 0x41, 0x53, 0x54, 0x2d, 0x41, 0x43,

0x4b, 0};

static const char \*states[] = {

closed,

syn\_rcvd,

syn\_sent,

established,

fin\_wait\_1,

fin\_wait\_2,

closing,

time\_wait,

last\_ack};

//////////////////////////////////////////////////////////////////////////////

static unsigned short

gen\_communic\_out(void \*arg)

{

struct httpd\_state \*s = (struct httpd\_state \*)arg;

return snprintf((char \*)uip\_appdata, UIP\_APPDATA\_SIZE,

"%u\n%u\n%u\n%u\n%u\n%u\n%u\n%u\n%u\n%u\n",

\*(myComs.fuelLow), \*(myComs.batteryLow), \*(myComs.solarPanelState),

\*(myComs.batteryLevel), \*(myComs.fuelLevel), \*(myComs.powerConsumption),

\*(myComs.battTemp), \*(myComs.secondTemp), \*(myComs.dist),

\*(myComs.peakFreq));

}

/\*---------------------------------------------------------------------------\*/

static

PT\_THREAD(communic\_out(struct httpd\_state \*s, char \*ptr))

{

PSOCK\_BEGIN(&s->sout);

PSOCK\_GENERATOR\_SEND(&s->sout, gen\_communic\_out, s);

PSOCK\_END(&s->sout);

}

/\*---------------------------------------------------------------------------\*/

static unsigned short

gen\_command\_in(void \*arg)

{

if(commHandle!=NULL)

vTaskResume(commHandle);

if(\*(myComs.TransmitFlag)){

sprintf( uip\_appdata,

"<input type=\"text\" name=\"LCD\" value=\"\" size=\"16\">"\

"<tr>Response: %s</tr>",

myComs.Response);

\*(myComs.TransmitFlag)=FALSE;

} else{

sprintf( uip\_appdata,

"<input type=\"text\" name=\"LCD\" value=\"\" size=\"16\">"\

"<tr>Response:</tr>",

NULL);

}

return strlen( uip\_appdata );

}

/\*---------------------------------------------------------------------------\*/

static

PT\_THREAD(command\_in(struct httpd\_state \*s, char \*ptr))

{

PSOCK\_BEGIN(&s->sout);

PSOCK\_GENERATOR\_SEND(&s->sout, gen\_command\_in, s);

PSOCK\_END(&s->sout);

}

/\*---------------------------------------------------------------------------\*/

//////////////////////////////////////////////////////////////////////////////

static unsigned short

generate\_tcp\_stats(void \*arg)

{

struct uip\_conn \*conn;

struct httpd\_state \*s = (struct httpd\_state \*)arg;

conn = &uip\_conns[s->count];

return snprintf((char \*)uip\_appdata, UIP\_APPDATA\_SIZE,

"<tr><td>%d</td><td>%u.%u.%u.%u:%u</td><td>%s</td><td>%u</td><td>%u</td><td>%c %c</td></tr>\r\n",

htons(conn->lport),

htons(conn->ripaddr[0]) >> 8,

htons(conn->ripaddr[0]) & 0xff,

htons(conn->ripaddr[1]) >> 8,

htons(conn->ripaddr[1]) & 0xff,

htons(conn->rport),

states[conn->tcpstateflags & UIP\_TS\_MASK],

conn->nrtx,

conn->timer,

(uip\_outstanding(conn))? '\*':' ',

(uip\_stopped(conn))? '!':' ');

}

/\*---------------------------------------------------------------------------\*/

static

PT\_THREAD(tcp\_stats(struct httpd\_state \*s, char \*ptr))

{

PSOCK\_BEGIN(&s->sout);

for(s->count = 0; s->count < UIP\_CONNS; ++s->count) {

if((uip\_conns[s->count].tcpstateflags & UIP\_TS\_MASK) != UIP\_CLOSED) {

PSOCK\_GENERATOR\_SEND(&s->sout, generate\_tcp\_stats, s);

}

}

PSOCK\_END(&s->sout);

}

/\*---------------------------------------------------------------------------\*/

static unsigned short

generate\_net\_stats(void \*arg)

{

struct httpd\_state \*s = (struct httpd\_state \*)arg;

return snprintf((char \*)uip\_appdata, UIP\_APPDATA\_SIZE,

"%5u\n", ((uip\_stats\_t \*)&uip\_stat)[s->count]);

}

static

PT\_THREAD(net\_stats(struct httpd\_state \*s, char \*ptr))

{

PSOCK\_BEGIN(&s->sout);

#if UIP\_STATISTICS

for(s->count = 0; s->count < sizeof(uip\_stat) / sizeof(uip\_stats\_t);

++s->count) {

PSOCK\_GENERATOR\_SEND(&s->sout, generate\_net\_stats, s);

}

#endif /\* UIP\_STATISTICS \*/

PSOCK\_END(&s->sout);

}

/\*---------------------------------------------------------------------------\*/

extern void vTaskList( signed char \*pcWriteBuffer );

static char cCountBuf[ 32 ];

long lRefreshCount = 0;

static unsigned short

generate\_rtos\_stats(void \*arg)

{

lRefreshCount++;

sprintf( cCountBuf, "<p><br>Refresh count = %d", lRefreshCount );

vTaskList( uip\_appdata );

strcat( uip\_appdata, cCountBuf );

return strlen( uip\_appdata );

}

/\*---------------------------------------------------------------------------\*/

static

PT\_THREAD(rtos\_stats(struct httpd\_state \*s, char \*ptr))

{

PSOCK\_BEGIN(&s->sout);

PSOCK\_GENERATOR\_SEND(&s->sout, generate\_rtos\_stats, NULL);

PSOCK\_END(&s->sout);

}

/\*---------------------------------------------------------------------------\*/

char \*pcStatus;

extern unsigned long uxParTestGetLED( unsigned long uxLED );

static unsigned short generate\_io\_state( void \*arg )

{

if( uxParTestGetLED( 0 ) )

{

pcStatus = "checked";

}

else

{

pcStatus = "";

}

sprintf( uip\_appdata,

"<input type=\"checkbox\" name=\"LED0\" value=\"1\" %s>LED"\

"<p>"\

"<input type=\"text\" name=\"LCD\" value=\"Enter LCD text\" size=\"16\">",

pcStatus );

return strlen( uip\_appdata );

}

/\*---------------------------------------------------------------------------\*/

static PT\_THREAD(led\_io(struct httpd\_state \*s, char \*ptr))

{

PSOCK\_BEGIN(&s->sout);

PSOCK\_GENERATOR\_SEND(&s->sout, generate\_io\_state, NULL);

PSOCK\_END(&s->sout);

}

/\*\* @} \*/

<--!Xinyu Sui

command.shtml>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<title>FreeRTOS.org uIP WEB server demo</title>

</head>

<BODY bgcolor="#CCCCff">

<font face="arial">

<a href="index.shtml">RTOS Stats</a> <b>|</b> <a href="stats.shtml">TCP Stats</a> <b>|</b> <a href="tcp.shtml">Connections</a> <b>|</b> <a href="display.shtml">Display</a> <b>|</b> <a href="command.shtml">Command</a> <b>|</b> <a href="io.shtml">IO</a>

<br><p>

<hr>

<b>Command</b><br>

<p>

To send a command, enter the command into the text box, then click "Update IO".

<p>

<form name="aForm" action="/command.shtml" method="get">

%! command-in

<p>

<input type="submit" value="Update IO">

</form><br><p>

</font>

</body>

</html>

<--!Xinyu Sui

display.shtml>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<title>FreeRTOS.org uIP WEB server demo</title>

</head>

<BODY onLoad="window.setTimeout(&quot;location.href='display.shtml'&quot;,5000)"bgcolor="#CCCCff">

<font face="arial">

<a href="index.shtml">RTOS Stats</a> <b>|</b> <a href="stats.shtml">TCP Stats</a> <b>|</b> <a href="tcp.shtml">Connections</a> <b>|</b> <a href="display.shtml">Display</a> <b>|</b> <a href="command.shtml">Command</a> <b>|</b> <a href="io.shtml">IO</a>

<br><p>

<hr>

<br><p>

<h2>

Satellite:&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;NAME<br>

Date:&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<span id="time"></span><br>

Operator: <br>

</h2>

<script language="javascript">

var today = new Date();

document.getElementById('time').innerHTML=today;

</script>

<table width="300" border="0">

<tr><td align="left"><font face="courier"><pre>

Fuel Low:

Battery Low:

Solar Panel State:

Battery Level:

Fuel Level:

Power Consumption:

Battery Temperature 1:

Battery Temperature 2:

Transport Distance:

Image Data:

</pre></font></td><td><pre>%! communic-out

</pre></table>

<br><p>

</font>

</body>

</html>

<--!Xinyu Sui

index.shtml>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<title>FreeRTOS.org uIP WEB server demo</title>

</head>

<BODY onLoad="window.setTimeout(&quot;location.href='index.shtml'&quot;,5000)"bgcolor="#CCCCff">

<font face="arial">

<a href="index.shtml">RTOS Stats</a> <b>|</b> <a href="stats.shtml">TCP Stats</a> <b>|</b> <a href="tcp.shtml">Connections</a> <b>|</b> <a href="display.shtml">Display</a> <b>|</b> <a href="command.shtml">Command</a> <b>|</b> <a href="io.shtml">IO</a>

<br><p>

<hr>

<br><p>

<h2>Task statistics</h2>

Page will refresh every 5 seconds.<p>

<font face="courier"><pre>Task State Priority Stack #<br>\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*<br>

%! rtos-stats

</pre></font>

</font>

</body>

</html>

<--!Xinyu Sui

io.shtml>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<title>FreeRTOS.org uIP WEB server demo</title>

</head>

<BODY bgcolor="#CCCCff">

<font face="arial">

<a href="index.shtml">RTOS Stats</a> <b>|</b> <a href="stats.shtml">TCP Stats</a> <b>|</b> <a href="tcp.shtml">Connections</a> <b>|</b> <a href="display.shtml">Display</a> <b>|</b> <a href="command.shtml">Command</a> <b>|</b> <a href="io.shtml">IO</a>

<br><p>

<hr>

<b>LED and LCD IO</b><br>

<p>

Use the check box to turn on or off the LED, enter text to display on the OLED display, then click "Update IO".

<p>

<form name="aForm" action="/io.shtml" method="get">

%! led-io

<p>

<input type="submit" value="Update IO">

</form>

<br><p>

</font></body>

</html>

<--!Xinyu Sui

stats.shtml>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<title>FreeRTOS.org uIP WEB server demo</title>

</head>

<BODY bgcolor="#CCCCff">

<font face="arial">

<a href="index.shtml">RTOS Stats</a> <b>|</b> <a href="stats.shtml">TCP Stats</a> <b>|</b> <a href="tcp.shtml">Connections</a> <b>|</b> <a href="display.shtml">Display</a> <b>|</b> <a href="command.shtml">Command</a> <b>|</b> <a href="io.shtml">IO</a>

<br><p>

<hr>

<br><p>

<h2>Network statistics</h2>

<table width="300" border="0">

<tr><td align="left"><font face="courier"><pre>

IP Packets dropped

Packets received

Packets sent

IP errors IP version/header length

IP length, high byte

IP length, low byte

IP fragments

Header checksum

Wrong protocol

ICMP Packets dropped

Packets received

Packets sent

Type errors

TCP Packets dropped

Packets received

Packets sent

Checksum errors

Data packets without ACKs

Resets

Retransmissions

No connection avaliable

Connection attempts to closed ports

</pre></font></td><td><pre>%! net-stats

</pre></table>

</font>

</body>

</html>

<--!Xinyu Sui

command.shtml>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<title>FreeRTOS.org uIP WEB server demo</title>

</head>

<BODY bgcolor="#CCCCff">

<font face="arial">

<a href="index.shtml">RTOS Stats</a> <b>|</b> <a href="stats.shtml">TCP Stats</a> <b>|</b> <a href="tcp.shtml">Connections</a> <b>|</b> <a href="display.shtml">Display</a> <b>|</b> <a href="command.shtml">Command</a> <b>|</b> <a href="io.shtml">IO</a>

<br><p>

<hr>

<br>

<h2>Network connections</h2>

<p>

<table>

<tr><th>Local</th><th>Remote</th><th>State</th><th>Retransmissions</th><th>Timer</th><th>Flags</th></tr>

%! tcp-connections

</pre></font>

</font>

</body>

</html>