RTM Commander Program Manual V 1.0

AUTHOR: Parker Landon - plandon@fnal.gov

AD Instrumentation, Fermi National Accelerator Laboratory

PROGRAM: RTM\_Commander.INO

PROGRAM START DATE: 7/12/2022

DOCUMENTATION START DATE: 7/15/2022

DOCUMENTATION END DATE: 7/15/2022

**Table of Contents:**

|  |  |  |
| --- | --- | --- |
| **Section Name** | **Description** | **Page(s)** |
| Hardware | The list of hardware managed by the program and links to their datasheets | 3 |
| LED Codes | Definitions for lit LEDs | 3 |
| Software Top-Down Review | A simplified overview of the program | 4-6 |
| Program Trace | A full-scale program trace with libraries defined | 7 |
| Library Documentation List | Listed libraries and their documentation | 7 |
| Functional Traces | Traces of each function and subfunction with descriptions, inputs, outputs, and associated library | 8-11 |
| #define/Variable List | A list of all variables in the program, their value, and relation to other variables | 12-13 |
| Assumptions/Things to Know | Assumptions made during programming, and basic concepts to know when using the program | 13 |
| Possible Errors | A list of possible errors, their locations, and their fixes | 13 |

**Hardware:**

 Sparkfun SAMD21G Mini Breakout:

 [SparkFun SAMD21 Mini Breakout - DEV-13664 - SparkFun Electronics](https://www.sparkfun.com/products/13664)



 24LC256 EEPROM:

 [24LC256 | Microchip Technology](https://www.microchip.com/en-us/product/24LC256)



 TMP175 Temperature Device:

 [TMP175-Q1 data sheet, product information and support | TI.com](https://www.ti.com/product/TMP175-Q1)

**LED Codes:**

* Blue solid: Verification completed
* Red solid: Analog/EEPROM/Temp read/write in progress (DO NOT REMOVE CARD)
* Green Solid: Safe for card removal **NOT IMPLEMENTED AT THE TIME OF THIS DOCUMENT**

**Software Top-Down Review:**

This code is written in the C language. Most of the functions used are built-in libraries with documentation listed below.

The program can be thought as two programs that are ran together. We have the main thread (program branch) that does actual processing, and a monitor thread that controls what is being processed.



Stepping into Initialize/Verify, we begin to set up both the Serial (115200 Baud) and I2C (400kHz) buses. Three pins are set as IO outputs to drive the blue, green, and red LEDs. Another pin is set to be an input for a trigger. Setting this trigger high will send data from the four ADCs over serial.



The section defined as “Check if there is EEPROM” will stop the program if no EEPROM is available on the I2C bus. Lastly, Initializing the children produces two child threads: *Monitor* and *I2C*.

After completing the initialization and verification, the blue LED will be set low, turning it on. AT THE TIME OF THIS DOCUMENT LOW IS ON HIGH IS OFF. Two children threads can then be scheduled to run. Either the Sequencer in *loop()* or the FreeRTOS built-in scheduler will decide which thread runs on the CPU. AT THE TIME OF THIS DOCUMENT ONLY THE SEQUENCER IS FULLY IMPLEMENTED. This means that a counter will toggle which branch we take (e.g. *Monitor* runs then *I2C*). However, the functionality of the code is essentially the same using either sequencer or a FreeRTOS schedule.

The child thread *Monitor* will decide which tasks the child thread *I2C* will complete. This is done through global Boolean flags. Below is a figure for a possible handshake for the children.



After completing a cycle of processing, the program repeats. For every cycle *Monitor* is ran, it may change the flags given different inputs. These inputs include the EEPROM read flag, Temperature read flag, the trigger flag. The first two flags outline their purpose, where the trigger flag branches to the ADCs. *I2C* will service each process, then complete the task by setting the flag to FALSE.

**Program Trace:**



**Library Documentation list:**

Arduino – <https://www.arduino.cc/reference/en/libraries/>

SparkFun External EEPROM – <http://librarymanager/All#SparkFun_External_EEPROM>

FreeRTOS – <https://github.com/BriscoeTech/Arduino-FreeRTOS-SAMD21/blob/master/src/FreeRTOS_SAMD21.h>

Wire Library – <https://www.arduino.cc/reference/en/language/functions/communication/wire/>

External Temperature Library – <https://github.com/jeremycole/Temperature_LM75_Derived>

**Functional Traces:**

**void setup()**

// HEADER: void setup()

// DESCRIPTION: Init Monitor thread

// init serial communication for debugging

// init Wire lib

// init pins for LED and Triggering

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Subfunctions** | **Task Description** | **Inputs** | **Outputs** | **Library** |
| SerialUSB.begin | Initialize Serial bus over USB | (BAUD\_RATE); | none | Arduino |
| vNopDelayMS | Delay serial to limit bus crashes | (DELAY\_TIME); | none | Arduino/FreeRTOS |
| pinMode | Set IO pins to either input or output | (PIN\_LOCATION, OUTPUT or INPUT); | none | Arduino |
| Wire.begin | Define an I2C connection called “Wire”. You can also change the name (e.g. Wire1.begin makes a channel called Wire1). | none | none | Wire |
| Wire.setClock | Set the clock speed of the I2C channel. Again the beginning word defines which wire is being called | (CLOCK\_SPEED); | none | Wire |
| myMem.begin | Check the I2C bus to see if an EEPROM is available | (EEPROM\_ADR, I2C\_NAME); | Boolean | External Temperature Library |
| xTaskCreate | Create a child thread given several parameters. | ((TaskHandle\_t) CHILD\_NAME,(char\*) CHILDS\_TASK,STACK\_SIZE (in words),TASK\_INPUTS,TASK\_PRIORITY,&TASK\_HANDLE); | BaseType\_t | FreeRTOS |

// fork children threads

**void loop()**

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

// HEADER: void loop()

// DESCRIPTION: Contains sequencer that controls

// each child

//

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Subfunctions** | **Task Description** | **Inputs** | **Outputs** | **Library** |
| Monitor | Child thread made by xTaskCreate. This thread will monitor I2C and control what functions it uses. | (void \* parameters) | none | Custom/FreeRTOS |
| I2C | Child thread made by xTaskCreate. This thread will manage the I2C bus and ADCs. This thread is controlled by Boolean flags that are changed by Monitor. | (void \* parameters) | none | Custom/FreeRTOS |

**void Monitor(void \* parameters)**

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

// HEADER: void Monitor(void \* parameters)

// DESCRIPTION: This thread will control where

// the I2C thread branches to.

//

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Subfunctions** | **Task Description** | **Inputs** | **Outputs** | **Library** |
| digitalWrite | Writes a voltage value to IO pin | (PIN\_LOCATION, VOLTAGE); | none | Arduino |
| digitalRead | Reads a voltage value from IO pin | (PIN\_LOCATION); | HIGH or LOW | Arduino |

**void I2C(void \* parameters)**

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

// HEADER: void I2C(void \* parameters)

// DESCRIPTION: I2C thread. This thread will branch

// to any I2C follower device function.

// This thread is also controlled by

// the "Monitor" thread.

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Subfunctions** | **Task Description** | **Inputs** | **Outputs** | **Library** |
| digitalWrite | Writes a voltage value to IO pin | (PIN\_LOCATION, VOLTAGE); | none | Arduino |
| ReadEEPROM | This function uses the I2C bus to read the entire memory of the EEPROM | none | none | Custom |
| temperature.readTemperatureF | Read the current temperature in degrees Fahrenheit and print to serial | none | float | External Temperature Library |
| ReadADCs | Read out the current ADC values and print to serial | none | none | Custom |

**void ReadADCs()**

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

// HEADER: void ReadADCs()

// DESCRIPTION: This function reads all ADCs

// and prints it to serial. The amount

// of times it reads is controlled by

// "ADC\_COUNTER"

//

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Subfunctions** | **Task Description** | **Inputs** | **Outputs** | **Library** |
| analogRead | Read the data on an IO pin | (PIN\_LOCATION); | int | Arduino |

**void ReadEEPROM()**

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

// HEADER: void ReadEEPROM()

// DESCRIPTION: This function reads the entire EEPROM

// and prints it to serial

//

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Subfunctions** | **Task Description** | **Inputs** | **Outputs** | **Library** |
| myMem.get | Get the value at the defined memory location. You can have multiple devices and connect to different ones by defining the name (e.g. Mem1.get and Mem2.get) | (long MEMORY\_LOCATION, byte HOLDING\_VAR); | none | SparkFun External EEPROM |

**#define/Variable List**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Value****If device dependent** | **Description** | **Reliant on Other Variables?** |
| EEPROM\_ADR | 0x50 | EEPROM Address. (1010 a0 a1 a2) setting all pins low gives 0x50 | no |
| TEMP\_ADR | 0x48 | Temperature Device Address. (1001 a0 a1 a2) setting all pins low gives 0x48 | no |
| MEMORY\_SIZE | 0x7D00 | 32Kb memory size of the EEPROM | no |
| CLOCK\_SPEED | 400000 | I2C Clock speed. Can be changed, documentation states limitations | no |
| BAUD\_RATE | 115200 or 9600 | Serial Baud rate | no |
| NUMOFSENSORS | 4 | Number of ADCs on Commander | no |
| VOLTAGERANGE | 3.3 | Voltage range of incoming signal | no |
| ADC\_COUNTER | 100 | The number of times ADC values will be sent of serial | no |
| RESOLUTION | 1023.0 | Commander resolution for ADC conversion | no |
| temperature | n/a | Temperature device definition. DO NOT CHANGE UNLESS DEVICE IS CHANGED | Yes, dependent on device initialization: Generic\_LM75 temperature; |
| RED | 2 | Pin location for LED | no |
| GREEN | 3 | Pin location for LED | no |
| BLUE | 4 | Pin location for LED | no |
| TRIGGER | 7 | Pin location for trigger input.TODO: CURRENT PIN IS 7 SHOULD BE 6 FOR FUTURE BOARDS | no |
| BLUE\_FLAG | false | Blue LED flag |  |
| RED\_FLAG | false | Red LED flag |  |
| GREEN\_FLAG | false | Green LED flag |  |
| TRIGGER\_FLAG | false | Trigger flag |  |
| EEPROM\_FLAG | false | EEPROM Read flag |  |
| TEMPER\_FLAG | false | Temperature flag |  |
| SEQ\_COUNT | 0 | Sequencer Count | Yes, needs to be less than SEQ\_LIM |
| SEQ\_LIM | 10 | Sequencer Count limit |  |
| WATCHDOG | 0 | Hard-coded watchdog timer for preliminary testing | Yes, set to zero to prevent runtime errors |

**Assumptions/Things to Know:**

* Many of the loops that run multiple cycles of read/writing data are predefined at the top of the code using #define. Alteration of these values can change code functionality, please read the #define list before-hand.
* If the red LED is on, this means that memory is being read/written. DO NOT REMOVE DEVICES, it can cause data corruption.
* In this program it is assumed that each service is ran to completion. If this does not occur, and or a thread becomes starved, a watchdog timer has been added to reset timers. \*THIS IS NOT FULLY IMPLEMENTED AT THE TIME OF THIS DOCUMENT\*

**Possible Errors:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Error** | **Possible Problem Location** | **Reason** | **Fix** |
| COM failing | Initialization | Faulty USB connection | Checking COM Port for Serial Port Monitor. Unplugging and plugging back in. |
| Spinning in initialization state (no Blue LED) | Initialization | Faulty or incorrect wiring for I2C bus | Check wire connections for SDA and SCL as well as pins a0-a3 for the correct addressing. |
| “No Memory Detected” | Initialization | EEPROM damaged, removed, or incorrectly wired | Check wiring or replace EEPROM. |
| "WATCHDOG ERROR - RESETTING PARAMS" | loop | Child threads not completing service routine.Scheduler may be starving systems.Children may be too large to complete in current schedule. | Change Scheduler or remove long functions from child threads. |
| Continually reading /reading only sections from EEPROM | ReadEEPROM() or #define MEMORY\_SIZE  | No defined memory size or incorrect size | Check documentation before altering #defines as the memory size varies from device. |