Testing

|  |  |
| --- | --- |
| **Verified** | **Function** |
|  | **int main(void);** |
| **X** | **void** **SaveWakeupInfo**(**void**); |
| **X** | **void** **DropOff** (**void**); |
| **X** | **void** **Release**(**unsigned** **int**); |
| **X** | **void** **ShowMenu**(**void**); |
| **X** | **void** **RTCWakeup**(**void**); |
| **X** | **void** **RTCFullWakeup**(**void**); |
| **X** | **inline** **void** **WriteTimeEEPROM**(**struct** RTCTime\*, uint16\_t); |
| **X** | **inline** **void** **ReadTimeEEPROM**(**struct** RTCTime\*, uint16\_t); |
| **X** | **void** **ReadEEPROM**(uint8\_t\*, uint8\_t, uint16\_t); |
| **X** | **void** **WriteEEPROM**(uint8\_t\*, uint8\_t, uint16\_t); |
| X | uint16\_t **GetBatVoltage**(**void**); |
| **X** | **void** **ADC\_Setup**(**void**); |
| X | uint16\_t **ADC\_Read**(); |
| **X** | **void** **ADC\_Stop**(); |
| **X** | **void** **BacktoSleep**(uint8\_t); |
| **X** | **void** **PrintRTC**(**void**); |
| **X** | **static** **inline** **void** **delay\_ms**(uint16\_t); |
| - | uint8\_t **Process\_RX\_Data**(**const** uint32\_t\* , uint8\_t ); |
| - | uint8\_t **RX\_RF\_DATA**(**const** uint32\_t\* , uint8\_t\* , uint8\_t ); |
| - | uint8\_t **TX\_RF\_DATA**(**const** uint32\_t\* , uint8\_t\* , uint8\_t ); |
| **X** | **void** **StartTim10ms**(uint16\_t); |
| **X** | **void** **StopTim10ms**(**void**); |

# 1 General Functions

## 1.1 Main Function

**Synopsis**

**int main(void);**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description/Reasoning | Input | Expected Output | Actual Output | Pass/Fail |
| Class 1 – Visual Code Inspection |  |  |  | Pass |
| Class 2 –Practical Use Test | Code has been used | No Problems Caused | No Problems Caused | Pass |

## 1.2 Record Wakeup Information Function

**Synopsis**

**void** **SaveWakeupInfo**(**void**);

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Description/Reasoning | | Input | Expected Output | Actual Output | Pass/Fail |
| Class 1 – Repeated Entries | |  |  |  |  |
| 1.1 | Repeated entries | 255 Entries | No writing past end of EEPROM or before 0x51 | Started writing at 0x51 and stopped at 501 decimal (7 bytes written here so 501-508) Continued to increment the counter but no further writing to EEPROM. | Pass |
| Class 2 –Practical Use Test | | Code has been used | No Problems Caused | No Problems Caused | Pass |

## 1.3 Start 10 Millisecond Timer Function

**Synopsis**

**void** **StartTim10ms**(uint16\_t Time);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description/Reasoning | Input | Expected Output | Actual Output | Pass/Fail |
| Class 2 –Practical Use Test | Code has been used | No Problems Caused | No Problems Caused | Pass |

## 1.4 Stop 10 Millisecond Timer Function

**Synopsis**

**void** **StopTim10ms**(**void**);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description/Reasoning | Input | Expected Output | Actual Output | Pass/Fail |
| Class 2 –Practical Use Test | Code has been used | No Problems Caused | No Problems Caused | Pass |

## 1.5 Delay in Milliseconds Function

**Synopsis**

**static** **inline** **void** **delay\_ms**(uint16\_t ms);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description/Reasoning | Input | Expected Output | Actual Output | Pass/Fail |
| Class 1 – Visual Code Inspection |  |  |  | Pass |
| Class 2 –Practical Use Test | Code has been used | No Problems Caused | No Problems Caused | Pass |

# 2 Core Functions

## 2.1 Short RTC Wakeup Function

**Synopsis**

**void** **RTCWakeup**(**void**);

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Description/Reasoning | | Input | Expected Output | Actual Output | Pass/Fail |
| Class 1 – Triggers (no RF RX) | |  |  |  |  |
| 1.1 | Not yet wakeup time | Waketime < CurrentTime | Set Alarm for 10s, no full wakeup, No RF RX | Set Alarm for 10s, no full wakeup, No RF RX | Pass |
| 1.2 | After wakeup time | Waketime > CurrentTime | Set Alarm for 10s, do full wakeup, No RF RX | Set Alarm for 10s, do full wakeup, No RF RX | Pass |
|  |  |  |  |  |  |
| Class 2 – Triggers (with RF RX) | |  |  |  |  |
| 2.1 | Not yet wakeup time | Waketime < CurrentTime | Set Alarm for 10s, no full wakeup, do RF RX and process | Set Alarm for 10s, no full wakeup, do RF RX and process | Pass |
| 2.2 | After wakeup time | Waketime > CurrentTime | Set Alarm for 10s, do full wakeup, do RF RX and process | Set Alarm for 10s, do full wakeup, do RF RX and process | Pass |
|  |  |  |  |  |  |

## 2.2 Full RTC Wakeup Function

**Synopsis**

**void** **RTCFullWakeup**(**void**);

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Description/Reasoning | | Input | Expected Output | Actual Output | Pass/Fail |
| Class 1 - Normal | |  |  |  |  |
| 1.1 | Good Bat, Alarm > 1 day | As per description | Alarm set 1 day from now | Alarm set 1 day from now | Pass |
| 1.2 | Good Bat, Alarm < 1 day | As per description | Alarm set to aTime | Alarm set to aTime | Pass |
| 1.3 | Alarm Time Good Bat | As per description | Dropoff, set alarm for 1 min | Dropoff, set alarm for 1 min | Pass |
| 1.4 | Low Bat, Alarm > 4 hrs | As per description | Alarm set 4 hrs from now | Alarm set 4 hrs from now | Pass |
| 1.5 | Low Bat, Alarm < 4 hrs | As per description | Alarm set to aTime | Alarm set to aTime | Pass |
| 1.6 | Alarm Time Low Bat | As per description | Dropoff, set alarm for 1 min | Dropoff, set alarm for 1 min | Pass |
| 1.7 | Alarm Time Crit Bat | As per description | Dropoff, set alarm for 1 min | Dropoff, set alarm for 1 min | Pass |
| 1.8 | Crit Bat | As per description | Dropoff, set alarm for 1 min | Dropoff, set alarm for 1 min | Pass |
|  |  |  |  |  |  |
| Class 2 - | |  |  |  |  |
|  | Invalid Alarm Time and Alarm Time | As per description | Alarm set to INVALID TIME! | Alarm set to INVALID TIME! | Not Good |
|  | Invalid Alarm Time, Good Bat, Alarm < 1 day | As per description | Alarm set to INVALID TIME! | Alarm set to INVALID TIME! | Not Good |
|  | Invalid Alarm Time and Crit Bat | As per description | Dropoff, set alarm for 1 min | Dropoff, set alarm for 1 min | Pass |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Key things to note here is that invalid alarm times are NOT supported and will NOT be automatically fixed by the Nichdrop.**

## 2.3 Drop Off Function

**Synopsis**

**void** **DropOff** (**void**);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description/Reasoning | Input | Expected Output | Actual Output | Pass/Fail |
| Class 1 – Visual Code Inspection |  |  |  | Pass |

## 2.4 Release Function

**Synopsis**

**void** **Release**(**unsigned** **int ms**);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description/Reasoning | Input | Expected Output | Actual Output | Pass/Fail |
| Class 1 – Validation from experience – has never failed in the many tests it’s been in |  |  |  | Pass |

## 2.5 Go Back to Sleep Function

**Synopsis**

**void** **BacktoSleep**(uint8\_t skip\_SleepMSG);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description/Reasoning | Input | Expected Output | Actual Output | Pass/Fail |
| Class 1 – Visual Code Inspection |  |  |  | Pass |
| Class 2 –Practical Use Test | Code has been used | No Problems Caused | No Problems Caused | Pass |

# 3 USART Communication Functions

## 3.1 Show Menu Function

**Synopsis**

**void** **ShowMenu**(**void**);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description/Reasoning | Input | Expected Output | Actual Output | Pass/Fail |
| Class 1 – Visual Code Inspection |  |  |  | Pass |
| Class 2 –Practical Use Test | Code has been used | No Problems Caused | No Problems Caused | Pass |

## 3.2 Print RTC Function

**Synopsis**

**void** **PrintRTC**(**void**);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description/Reasoning | Input | Expected Output | Actual Output | Pass/Fail |
| Class 1 – Visual Code Inspection |  |  |  | Pass |
| Class 2 –Practical Use Test | Code has been used | No Problems Caused | No Problems Caused | Pass |

# 4 EEPROM Functions

## 4.1 Write Time to EEPROM Function

**Synopsis**

**inline** **void** **WriteTimeEEPROM**(**struct** RTCTime\* mTime, uint16\_t Address);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description/Reasoning | Input | Expected Output | Actual Output | Pass/Fail |
| Class 1 – Visual Code Inspection |  |  |  | Pass |
| Class 2 –Practical Use Test | Code has been used | No Problems Caused | No Problems Caused | Pass |

## 4.2 Read Time From EEPROM Function

**Synopsis**

**inline** **void** **ReadTimeEEPROM**(**struct** RTCTime\* mTime, uint16\_t Address);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description/Reasoning | Input | Expected Output | Actual Output | Pass/Fail |
| Class 1 – Visual Code Inspection |  |  |  | Pass |
| Class 2 –Practical Use Test | Code has been used | No Problems Caused | No Problems Caused | Pass |

## 4.3 General Read from EEPROM Function

**Synopsis**

**void** **ReadEEPROM**(uint8\_t\* Data, uint8\_t Length, uint16\_t Address);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description/Reasoning | Input | Expected Output | Actual Output | Pass/Fail |
| Class 1 – Visual Code Inspection |  |  |  | Pass |
| Class 2 –Practical Use Test | Code has been used | No Problems Caused | No Problems Caused | Pass |

## 4.4 General Write to EEPROM Function

**Synopsis**

**void** **WriteEEPROM**(uint8\_t\* Data, uint8\_t Length, uint16\_t Address);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description/Reasoning | Input | Expected Output | Actual Output | Pass/Fail |
| Class 1 – Visual Code Inspection |  |  |  | Pass |
| Class 2 –Practical Use Test | Code has been used | No Problems Caused | No Problems Caused | Pass |

# 5 ADC Functions

## 5.1 Get Battery Voltage Function

**Synopsis**

uint16\_t **GetBatVoltage**(**void**);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description/Reasoning | Input | Expected Output | Actual Output | Pass/Fail |
| Class 2 –Practical Use Test | Code has been used | No Problems Caused | No Problems Caused | Pass |

## 5.2 Setup ADC Function

**Synopsis**

**void** **ADC\_Setup**(**void**);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description/Reasoning | Input | Expected Output | Actual Output | Pass/Fail |
| Class 2 –Practical Use Test | Code has been used | No Problems Caused | No Problems Caused | Pass |

## 5.3 Take ADC Reading Function

**Synopsis**

uint16\_t **ADC\_Read**();

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description/Reasoning | Input | Expected Output | Actual Output | Pass/Fail |
| Class 2 –Practical Use Test | Code has been used | No Problems Caused | No Problems Caused | Pass |

## 5.4 Disable ADC Function

**Synopsis**

**void** **ADC\_Stop**();

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description/Reasoning | Input | Expected Output | Actual Output | Pass/Fail |
| Class 2 –Practical Use Test | Code has been used | No Problems Caused | No Problems Caused | Pass |

# 7 RF Functions

## 7.1 Process Received RF Data Function

**Synopsis**

uint8\_t **Process\_RX\_Data**(**const** uint32\_t\* part1, uint8\_t AllowStayOn);

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Description/Reasoning | | Input | Expected Output | Actual Output | Pass/Fail |
| Class 1 - | |  |  |  |  |
| 1.1 |  |  |  |  |  |
| 1.2 |  |  |  |  |  |
| 1.3 |  |  |  |  |  |
| Class 2 - | |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Not Presently Verified**

## 7.2 Receive RF Data Function

**Synopsis**

uint8\_t **RX\_RF\_DATA**(**const** uint32\_t\* part1, uint8\_t\* DestData, uint8\_t b\_len);

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Description/Reasoning | | Input | Expected Output | Actual Output | Pass/Fail |
| Class 1 - | |  |  |  |  |
| 1.1 |  |  |  |  |  |
| 1.2 |  |  |  |  |  |
| 1.3 |  |  |  |  |  |
| Class 2 - | |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Not Presently Verified**

## 7.3 Transmit RF Data Function

**Synopsis**

uint8\_t **TX\_RF\_DATA**(**const** uint32\_t\* part1, uint8\_t\* SrcData, uint8\_t b\_len);

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Description/Reasoning | | Input | Expected Output | Actual Output | Pass/Fail |
| Class 1 - | |  |  |  |  |
| 1.1 |  |  |  |  |  |
| 1.2 |  |  |  |  |  |
| 1.3 |  |  |  |  |  |
| Class 2 - | |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Not Presently Verified

# 6 RTC Functions

## 6.1 Get Byte from RTC Function

**Synopsis**

**unsigned** **char** **RTC\_get**(**unsigned** **char pos**);

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Description/Reasoning | | Input | Expected Output | Actual Output | Pass/Fail |
| Class 1 - | |  |  |  |  |
| 1.1 |  |  |  |  |  |
| 1.2 |  |  |  |  |  |
| 1.3 |  |  |  |  |  |
| Class 2 - | |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

## 6.2 Get Flags from RTC Function

**Synopsis**

**unsigned** **char** **RTC\_getflags**(**void**);

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Description/Reasoning | | Input | Expected Output | Actual Output | Pass/Fail |
| Class 1 - | |  |  |  |  |
| 1.1 |  |  |  |  |  |
| 1.2 |  |  |  |  |  |
| 1.3 |  |  |  |  |  |
| Class 2 - | |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

## 6.3 Set Byte on RTC Function

**Synopsis**

**void** **RTC\_set**(**unsigned** **char** pos, **unsigned** **char val**);

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Description/Reasoning | | Input | Expected Output | Actual Output | Pass/Fail |
| Class 1 - | |  |  |  |  |
| 1.1 |  |  |  |  |  |
| 1.2 |  |  |  |  |  |
| 1.3 |  |  |  |  |  |
| Class 2 - | |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

## 6.4 Set Byte on RTC with Mask Function

**Synopsis**

**void** **RTC\_set\_msk**(**unsigned** **char** pos, **unsigned** **char** val, **unsigned** **char** bMsk);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description/Reasoning | Input | Expected Output | Actual Output | Pass/Fail |
| Class 2 –Practical Use Test | Code has been used | No Problems Caused | No Problems Caused | Pass |

## 6.5 Enable RTC Alarm Function

**Synopsis**

**void** **RTC\_EnableAlarm**(**void**);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description/Reasoning | Input | Expected Output | Actual Output | Pass/Fail |
| Class 2 –Practical Use Test | Code has been used | No Problems Caused | No Problems Caused | Pass |

## 6.6 Print Time to USART Function

**Synopsis**

**void** **PrintTime**(**struct** RTCTime\*pTime , **char**\* lbl);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description/Reasoning | Input | Expected Output | Actual Output | Pass/Fail |
| Class 2 –Practical Use Test | Code has been used | No Problems Caused | No Problems Caused | Pass |

## 6.7 Get RTC Time Function

**Synopsis**

**void** **RTC\_GetTime**(**struct** RTCTime\* Time);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description/Reasoning | Input | Expected Output | Actual Output | Pass/Fail |
| Class 2 –Practical Use Test | Code has been used | No Problems Caused | No Problems Caused | Pass |

## 6.8 Get RTC Alarm Function

**Synopsis**

**void** **RTC\_GetAlarm**(**struct** RTCTime\* Alarm);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description/Reasoning | Input | Expected Output | Actual Output | Pass/Fail |
| Class 2 –Practical Use Test | Code has been used | No Problems Caused | No Problems Caused | Pass |

## 6.9 Set RTC Time Function

**Synopsis**

**void** **RTC\_SetTime**(**struct** RTCTime\* Time);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description/Reasoning | Input | Expected Output | Actual Output | Pass/Fail |
| Class 2 –Practical Use Test | Code has been used | No Problems Caused | No Problems Caused | Pass |

## 6.10 Set RTC Alarm Function

**Synopsis**

**void** **RTC\_SetAlarm**(**struct** RTCTime\* Alarm);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description/Reasoning | Input | Expected Output | Actual Output | Pass/Fail |
| Class 2 –Practical Use Test | Code has been used | No Problems Caused | No Problems Caused | Pass |

## 6.11 Convert Integer to BCD Function

**Synopsis**

uint8\_t **IntToBCD**(uint8\_t val);

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Description/Reasoning | | Input | Expected Output | Actual Output | Pass/Fail |
| Class 1 – Random Valid Inputs | |  |  |  |  |
| 1.1 | 9 | 09 | 0x09 | 0x09 | Pass |
| 1.2 | 18 | 18 | 0x18 | 0x18 | Pass |
| 1.3 | 27 | 27 | 0x27 | 0x27 | Pass |
| 1.4 | 36 | 36 | 0x36 | 0x36 | Pass |
| 1.5 | 45 | 45 | 0x45 | 0x45 | Pass |
| 1.6 | 99 | 99 | 0x99 | 0x99 | Pass |
| 1.7 | 0 | 0 | 0x00 | 0x00 | Pass |
|  |  |  |  |  |  |
| Class 2 –Erroneous Inputs | |  |  |  |  |
| 2.1 | 100 | 100 | 0x99 | 0x99 | Pass |
| 2.2 | 123 | 123 | 0x99 | 0x99 | Pass |
| 2.3 | 255 | 255 | 0x99 | 0x99 | Pass |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

## 6.12 Convert BCD to Integer Function

**Synopsis**

uint8\_t **BCDToInt**(uint8\_t val);

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Description/Reasoning | | Input | Expected Output | Actual Output | Pass/Fail |
| Class 1 – Random Valid Inputs | |  |  |  |  |
| 1.1 | 0x09 | 0x09 | 9 | 9 | Pass |
| 1.2 | 0x18 | 0x18 | 18 | 18 | Pass |
| 1.3 | 0x27 | 0x27 | 27 | 27 | Pass |
| 1.4 | 0x36 | 0x36 | 36 | 36 | Pass |
| 1.5 | 0x45 | 0x45 | 45 | 45 | Pass |
| 1.6 | 0x99 | 0x99 | 99 | 99 | Pass |
| 1.7 | 0x00 | 0x00 | 0 | 0 | Pass |
|  |  |  |  |  |  |
| Class 2 –Erroneous Inputs | |  |  |  |  |
| 2.1 | 0xA0 | 0xA0 | 99 | 99 | Pass |
| 2.2 | 0xFF | 0xFF | 99 | 99 | Pass |
| 2.3 | 0xC4 | 0xC4 | 99 | 99 | Pass |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

## 6.13 Copy Time Function

**Synopsis**

**void** **CopyTime** (**struct** RTCTime\* SrcTime, **struct** RTCTime\* DstTime);

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Description/Reasoning | | Input | Expected Output | Actual Output | Pass/Fail |
| Class 1 – Random Time/Date | |  |  |  |  |
| 1.1 | Now | 13:32:25 on 13/7/10 | 13:32:25 on 13/7/10 | 13:32:25 on 13/7/10 | Pass |

## 6.14 Add Times Together Function

**Synopsis**

**void** **AddTimes**(**struct** RTCTime\* DateTime, **struct** RTCTime\* AddTime);

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Description/Reasoning | | Input | Expected Output | Actual Output | Pass/Fail |
| Class 1 - Normal | | Input 1 = 10-10-10 10:10:10 | (Input 1) |  |  |
| 1.1 | Add Second | Input 2 = 00-00-00 00:00:35 | 10-10-10 10:10:45 | 10-10-10 10:10:45 | Pass |
| 1.2 | Add Minute | Input 2 = 00-00-00 00:24:00 | 10-10-10 10:34:10 | 10-10-10 10:34:10 | Pass |
| 1.3 | Add Hour | Input 2 = 00-00-00 12:00:00 | 10-10-10 22:10:10 | 10-10-10 22:10:10 | Pass |
| 1.4 | Add Day | Input 2 = 00-00-08 00:00:00 | 10-10-18 10:10:10 | 10-10-18 10:10:10 | Pass |
| 1.5 | Add Month | Input 2 = 00-01-00 00:00:00 | 10-11-10 10:10:10 | 10-11-10 10:10:10 | Pass |
| 1.6 | Add Year | Input 2 = 03-00-00 00:00:00 | 13-10-10 10:10:10 | 13-10-10 10:10:10 | Pass |
| 1.7 | Add All | Input 2 = 03-01-08 12:24:35 | 13-11-18 22:34:45 | 13-11-18 22:34:45 | Pass |
|  |  |  |  |  |  |
| Class 3 – Basic Overflow | | Input 1 = 10-10-10 10:10:10 | (Input 1) |  |  |
| 3.1 | Second Overflow | Input 2 = 00-00-00 00:00:55 | 10-10-10 10:11:05 | 10-10-10 10:11:05 | Pass |
| 3.2 | Minute Overflow | Input 2 = 00-00-00 00:53:00 | 10-10-10 11:03:10 | 10-10-10 11:03:10 | Pass |
| 3.3 | Hour Overflow | Input 2 = 00-00-00 18:00:00 | 10-10-11 04:10:10 | 10-10-11 04:10:10 | Pass |
| 3.4 | Day Overflow | Input 2 = 00-00-25 00:00:00 | 10-11-04 10:10:10 | 10-11-04 10:10:10 | Pass |
| 3.7 | Month Overflow | Input 2 = 00-05-00 00:00:00 | 11-03-10 10:10:10 | 11-03-10 10:10:10 | Pass |
| 3.8 | All Overflow | Input 2 = 00-05-25 18:53:55 | 11-04-05 05:04:05 | 11-04-05 05:04:05 | Pass |
|  |  |  |  |  |  |
| Class 4 – Month Overflows | | Input 2 = 00-00-20 00:00:00 | (Input 1) |  |  |
| 4.1 | 30 Day Month Overflow | Input 1 = 10-04-15 00:00:00 | 10-05-05 00:00:00 | 10-05-05 00:00:00 | Pass |
| 4.2 | 31 Day Month Overflow | See 3.7 | See 3.7 | See 3.7 | Pass |
| 4.3 | February Overflow Non Leap Year | Input 1 = 10-02-20 00:00:00 | 10-03-12 00:00:00 | 10-03-12 00:00:00 | Pass |
| 4.4 | February Overflow Leap Year (%4) | Input 1 = 08-02-20 00:00:00 | 08-03-11 00:00:00 | 08-03-11 00:00:00 | Pass |
|  | February Overflow Leap Year (%400) | Input 1 = 00-02-20 00:00:00 | 00-03-11 00:00:00 | 00-03-11 00:00:00 | Pass |
|  | No Leap Year (%4 && %100) | Input 1 = 100-02-20 00:00:00 | 100-03-12 00:00:00 | 100-03-12 00:00:00 | Pass |
|  |  |  |  |  |  |
| Class 5 – Rolling Overflow | | Input 2 = 00-00-00 00:00:10 | (Input 1) |  |  |
| 5.1 | Seconds Through To Year | Input 1 = 10-12-31 23:59:53 | 11-01-01 00:00:03 | 11-01-01 00:00:03 | Pass |
|  |  |  |  |  |  |
| Class 6 – Multiple Overflows | | Input 1 = 10-10-10 10:10:10 | (Input 1) |  |  |
| 6.1 | Seconds | Input 2 = 00-00-00 00:00:200 | 10-10-10 10:13:30 | 10-10-10 10:13:30 | Pass |
| 6.2 | Minutes | Input 2 = 00-00-00 00:200:00 | 10-10-10 13:30:10 | 10-10-10 13:30:10 | Pass |
| 6.3 | Hours | Input 2 = 00-00-00 56:00:00 | 10-10-12 18:10:10 | 10-10-12 18:10:10 | Pass |
| 6.4 | Days | Input 2 = 00-00-60 00:00:00 | 10-12-09 10:10:10 | 10-12-09 10:10:10 | Pass |
| 6.5 | Months | Input 2 = 00-27-00 00:00:00 | 13-01-10 10:10:10 | 13-01-10 10:10:10 | Pass |
| 6.6 | All | Input 2 = 00-27-60 56:200:200 | 13-03-11 21:33:30 | 13-03-11 21:33:30 | Pass |

## 6.15 Clear Time Function

**Synopsis**

**void** **ClearTime**(**struct** RTCTime\* Time1);

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Description/Reasoning | | Input | Expected Output | Actual Output | Pass/Fail |
| Class 1 – Clearing Times | |  |  |  |  |
| 1.1 | Clear Random Time | 13:32:25 on 13/7/10 (13/7/2010) | 00:00:00 on 0/0/00 | 00:00:00 on 0/0/00 | Pass |
| 1.2 | Clear Empty Time | 00:00:00 on 0/0/00 | 00:00:00 on 0/0/00 | 00:00:00 on 0/0/00 | Pass |
|  |  |  |  |  |  |
| Class 2 – Erroneous Inputs | |  |  |  |  |
| 2.1 | Pass Invalid Pointer | Invalid Pointer | Garbage/Corruption of data | Garbage/Corruption of data | Pass |
|  |  |  |  |  |  |

## 6.16 First Time Greater Function

**Synopsis**

uint8\_t **CompareTimesGreater** (**struct** RTCTime\* Time1, **struct** RTCTime\* Time2);

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Description/Reasoning | | Input | Expected Output | Actual Output | Pass/Fail |
| Class 1 – Time1 < Time2 | | Time1 = 10-10-10 10:10:10 |  |  |  |
| 1.1 | Seconds Less Than | Time2 = 10-10-10 10:10:11 | 0 | 0 | Pass |
| 1.2 | Minutes Less Than | Time2 = 10-10-10 10:11:10 | 0 | 0 | Pass |
| 1.3 | Hours Less Than | Time2 = 10-10-10 11:10:10 | 0 | 0 | Pass |
| 1.4 | Days Less Than | Time2 = 10-10-11 10:10:10 | 0 | 0 | Pass |
| 1.5 | Months Less Than | Time2 = 10-11-10 10:10:10 | 0 | 0 | Pass |
| 1.6 | Years Less Than | Time2 = 11-10-10 10:10:10 | 0 | 0 | Pass |
| 1.7 | All Less Than | Time2 = 11-11-11 11:11:11 | 0 | 0 | Pass |
|  |  |  |  |  |  |
| Class 2 – Time 1 > Time 2 | | Time2 = 10-10-10 10:10:10 |  |  |  |
| 1.1 | Seconds Less Than | Time1 = 10-10-10 10:10:11 | 1 | 1 | Pass |
| 1.2 | Minutes Less Than | Time1 = 10-10-10 10:11:10 | 1 | 1 | Pass |
| 1.3 | Hours Less Than | Time1 = 10-10-10 11:10:10 | 1 | 1 | Pass |
| 1.4 | Days Less Than | Time1 = 10-10-11 10:10:10 | 1 | 1 | Pass |
| 1.5 | Months Less Than | Time1 = 10-11-10 10:10:10 | 1 | 1 | Pass |
| 1.6 | Years Less Than | Time1 = 11-10-10 10:10:10 | 1 | 1 | Pass |
| 1.7 | All Less Than | Time1 = 11-11-11 11:11:11 | 1 | 1 | Pass |
|  |  |  |  |  |  |
| Class 3 – Time 1 == Time 2 | | Time1 = 10-10-10 10:10:10 |  |  |  |
| 3.1 | Equal Times | Time2 = 10-10-10 10:10:10 | 2 | 2 | Pass |

## 6.17 DateTime is Valid Function

**Synopsis**

uint8\_t **ValidDateTime**(**struct** RTCTime\* Time1);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description/Reasoning | Input | Expected Output | Actual Output | Pass/Fail |
| Class 1 – Visual Inspection of Code |  |  |  | Pass |

## 6.18 Compare Times Function

**Synopsis**

uint8\_t **CompareTimes** (**struct** RTCTime\* Time1, **struct** RTCTime\* Time2);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description/Reasoning | Input | Expected Output | Actual Output | Pass/Fail |
| Class 1 - Visual Inspection of Code |  |  |  | Pass |