* Program flow is as follows
  + Setup program logging (console and file)
  + Setup program parameters
  + Setup SPI device
  + Setup X2 minimalmodbus device
  + Setup T-Node FR minimalmodbus device
  + Setup RPi’s GPIOs
  + Create Test Result output .csv
  + Setup the test module sections
    - Allows the user to disable certain sections if desired by editing a flag in a file that appears on the Desktop
    - These configured modules are what will be run for all PCBs until the program is re-run
    - Initiates the active module to 0
  + Get the first board’s serial number
  + Start the individual PCB Test loop
    - * For each module the test is run, the results are reported on the console and the results are written to the result .csv
      * Then if the test was successful it clears the module test flag so it isn’t run if the board is re-tested
      * If a module is skipped it writes skipped to the Test Results file and prints this out on the console
      * Finally the active module number is incremented before moving onto the next module
    - Test the 3V LDO
    - Test the RS-485 driver, EE and processor
    - Test the RTC clock and RTC clock battery
    - Test the 3.3V SEPIC converter
    - Test the serial flash
    - Test the SD card
    - Test the priority power switch
    - Test the system current
    - Test the 12V SEPIC converter
    - Test the 5V LDO
    - Test the 12V sensor switch
    - Test the sensor current
    - Test priority power out switch
    - Test the sensor port
    - Test the pressure, temperature and humidity chip
    - Test the trigger lines
    - Test the RTU RS-485 passthrough
    - Test the magnetic switches
    - Test the K64 LEDs
    - Test the Wi-Fi module and LEDs
    - The time to test the board is displayed with a completion message
    - The program checks to see if any modules failed. If there were any failures, these are shown to the user and they are prompted to retry if desired
    - The board is powered down and the user is prompted to remove the board
    - The user is then prompted to connect a new board and enter its serial number to re-initiate the loop or exit the program.
  + Before ending the program the power is disabled, the output and log files are closed and the GPIOs are de-initialized.
* Function definitions are below
  + combineFrom16Bits(separate)
    - Use: Takes up to (4) separate 16-bit values and combines them into a single value
    - Inputs:
      * separate: list of 16-bit integers
        + [int1,int2,int3,int4]
    - Outputs:
      * combined integer (integer)
  + checkStatus(x2,mbRetries,mbDictName,clearText)
    - Use: Check a Modbus status register on the X2 and returns the result
    - Inputs:
      * x2: simplemodbus device
      * mbRetries: number of times to retry sending a Modbus command (integer)
      * mbDictName: the dictionary name that holds the desired parameter’s register and number of registers (string)
      * clearText: the display text that should appear in the phrase “the \_\_\_ status is/was” (string)
    - Output:
      * A list with the following values (list)
        + [Element 0] = the status value returned by the device

Should be 0 or 1

* + - * + [Element 1] = the “Pass” or “Fail - …” message
  + enableDisable(x2,mbRetries,mbDictName,clearText,onOff)
    - Use: Used to enable or disable one of the X2s switches
    - Inputs:
      * x2: simplemodbus device
      * mbRetries: number of times to retry sending a Modbus command (integer)
      * mbDictName: the dictionary name that holds the desired parameter’s register and number of registers (string)
      * clearText: the display text that should appear in the phrase “the \_\_\_ was successfully/not successfully…” (string)
      * onOff: binary value to indicate on or off
        + 0: Turn the switch off (integer)
        + 1: Turn the switch on (integer)
    - Output:
      * True if it successfully toggle the switch (bool)
      * False if it did not successfully toggle the switch (bool)
  + getModulesToTest()
    - Use: Prompts the user to modify a file that is created on the desktop if they would like to deviate from the standard test and only test certain modules. Then it reads from this file to determine which modules to run.
    - Inputs:
      * Nothing
    - Output:
      * A list with elements for each module (20 of them) that contains a 0 or 1 (list)

0=don’t run

1=do run

* + - * + [Element 0]=First module
        + [Element 1]=Second module
        + …
        + [Element 19]=Twentieth module
  + getSN(snlen)
    - Use: Used to prompt the user for a serial number and confirm that it is the correct number of characters
    - Inputs:
      * snlen: the number of required characters for the serial number (integer)
    - Output:
      * The serial number as a string
  + magSWCheck(x2,mbRetries,mbDictName,magSWNum)
    - Use: Asks the user to trigger the magnetic switch and prompt to see if the LED lights up
    - Inputs:
      * x2: simplemodbus device
      * mbRetries: number of times to retry sending a Modbus command (integer)
      * mbDictName: the dictionary name that holds the desired parameter’s register and number of registers (string)
      * magSWNum: text for which magnetic switch is being tested (string)
    - Output:
      * A list with the following values (list)
        + [Element 0]=”Pass” or “Fail - …” message for the read status (string)
        + [Element 1]=time between magnet trigger and user acknowledgement (float)
        + [Element 2]=”Pass” or Fail - …” message for the LED status (string)
  + mbReadFloatRetries(device,reg,numReg=2,retries=5)
    - Use: Reads an IEEE floating point value from the minimalmodbus device. Provides for a specified number of retries and allows the program to continue if the read fails (minimalmodbus returns an exception and exits the program).
    - Inputs:
      * device: minimalmodbus device
      * reg: the register from which to read (integer)
      * numReg: the number of registers from which to read [default is 2] (integer)
      * retries: number of times to attempt the read before giving up [default is 5] (integer)
    - Output:
      * False if it fails (bool)
      * If successful, a list with the following values (list)
        + [Element 0]=the read float value (float)
  + mbReadRetries(device,reg,numReg=1,retries=5)
    - Use: Reads specified number of registers from the minimalmodbus device. Provides for a specified number of retries and allows the program to continue if the read fails (minimalmodbus returns an exception and exits the program).
    - Inputs:
      * device: minimalmodbus device
      * reg: the register from which to read (integer)
      * numReg: the n number of registers from which to read [default is 1] (integer)
      * retries: number of times to attempt the read before giving up [default is 5] (integer)
    - Output:
      * False if it fails (bool)
      * If successful, a list with the following values (list)
        + [Element 0]=the first 16-bit value read (integer)
        + [Element 1]=the second 16-bit value read (integer)
        + …
        + [Element n]=the last 16-bit value read (integer)
  + mbWriteRetries(device,reg,value,retries=5)
    - Use: Writes the specified values to the minimalmodbus device. Provides for a specified number of retries and allows the program to continue if the write fails (minimalmodbus returns an exception and exits the program).
    - Inputs:
      * device: minimalmodbus device
      * reg: the register from which to read (integer)
      * value: a list of 16-bit values to write (list of integers)
      * retries: number of times to attempt the read before giving up [default is 5] (integer)
    - Output:
      * False if it fails (bool)
      * If successful, it returns the list of values provided to it as an input (list of integers)
  + powerOff(GPIO,pinDict,pinValue,delay=5)
    - Use: Checks the current status of a GPIO and turns it off if it is on. Also provides a delay before the program continues to ensure it is fully off
    - Inputs:
      * GPIO: the GPIO library instance
      * pinDict: the dictionary that contains the readable text equivalents for the used GPIO pins (dictionary)
      * pinValue: the dictionary name of which pin to toggle (string)
      * delay: the amount of time (seconds) to wait after toggling before returning [default is 5] (integer)
    - Output:
      * Always returns True
  + powerOn(x2,mbRetries,GPIO,pinDict,pinValue,delay=3)
    - Use: Checks the current status of a GPIO and turns it on if it is off. Also provides a delay before the program continues to ensure it is fully on
    - Inputs:
      * x2: simplemodbus device
      * mbRetries: number of times to retry sending a Modbus command (integer)
      * GPIO: the GPIO library instance
      * pinDict: the dictionary that contains the readable text equivalents for the used GPIO pins (dictionary)
      * pinValue: the dictionary name of which pin to toggle (string)
      * delay: the amount of time (seconds) to wait after toggling before returning [default is 3] (integer)
    - Output:
      * Always returns True
  + prioPwrChannelTest(GPIO,pinDict,x2,mbRetries,mbDictName,validCheck,validValue)
    - Use: Tests an individual priority power path channels status, voltage and valid output
    - Inputs:
      * GPIO: the GPIO library instance
      * pinDict: the dictionary that contains the readable text equivalents for the used GPIO pins (dictionary)
      * x2: simplemodbus device
      * mbRetries: number of times to retry sending a Modbus command (integer)
      * mbDictName: the dictionary name that holds the desired parameter’s register and number of registers (string)
      * validCheck: True if the valid lines should be checked or False if not (bool)
      * validValue: decimal equivalent of the binary mask that represents the valid lines’ state (integer)
    - Output:
      * A list with the following values (list)
        + [Element 0]=”Pass” or “Fail - …” status of the channel for both the voltage and valid lines (string)
        + [Element 1]=the read voltage (float)
        + [Element 2]=the decimal equivalent of the read binary mask the represents the valid lines’ state (integer)
  + readAnalog(spi,ch,scale=1)
    - Use: Reads the analog voltage from the MCP3008 chip connected to spi channel 0
    - Inputs:
      * spi: the spi instance
      * ch: channel value [must be 0-7] (integer)
      * scale: the scaling value to multiple the read value by to get a true result [default is 1] (float)
    - Output:
      * Read and scaled voltage value (float)
  + scaleValue(R1,R2)
    - Use: Calculates a voltage dividers scaling value
    - Inputs:
      * R1: top resistor value of voltage divider [connected to input] (float)
      * R2: bottom resistor value of voltage divider [connected to ground] (float)
    - Output:
      * Scaling value (float)
  + Sensor12VSW(GPIO,pinDict,x2,mbRetries,spi,mbDictName,clearText,spiCh)
    - Use: Reads an individual port’s voltage and check the level is in range
    - Inputs:
      * GPIO: the GPIO library instance
      * pinDict: the dictionary that contains the readable text equivalents for the used GPIO pins (dictionary)
      * x2: simplemodbus device
      * mbRetries: number of times to retry sending a Modbus command (integer)
      * spi: the spi instance
      * mbDictName: the dictionary name that holds the desired parameter’s register and number of registers (string)
      * clearText: the display text that should appear in the phrase “Reading 12V \_\_\_\_ Voltage” and “Unable to enable \_\_\_” (string)
      * spiCh: channel value [must be 0-7] (integer)
    - Output:
      * A list with the following values (list)
        + [Element 0]=”Pass” or “Fail - …” status of port (string)
        + [Element 1]= the read voltage (float)
  + sensorTest(x2,mbRetries,RS232Channel)
    - Use: Tests RS-485, RS-232, and SDI-12 for a sensor port
    - Inputs:
      * x2: simplemodbus device
      * mbRetries: number of times to retry sending a Modbus command (integer)
      * RS232Channel: the Modbus dictionary name for the specific RS-232 channel [RS232AComTest, RS232BComTest, RS232CComTest] (string)
    - Output:
      * A list with the following values (list)
        + [Element 0]=”Pass” or “Fail - …” status of RS-485 (string)
        + [Element 1]=”Pass” or “Fail - …” status of RS-232 (string)
        + [Element 2]=”Pass” or “Fail - …” status of SDI-12 (string)
  + splitInto16Bits(combined)
    - Use: Takes a number and splits it into individual 16-bit values
    - Inputs:
      * combined: a single 16,32,48, or 64-bit number (integer)
    - Output:
      * A list with the following values (list)
        + [Element 0]=lowest 16-bit value (integer)
        + [Element 1]=second 16-bit value (integer)
        + [Element 2]=third 16-bit value (integer)
        + [Element 3]=highest 16-bit value (integer)
  + triggerLineTest(GPIO,pinDict,x2,mbRetries,mbDictName,clearText,IOName)
    - Use: Tests if a trigger line enables and disables successfully
    - Inputs:
      * GPIO: the GPIO library instance
      * pinDict: the dictionary that contains the readable text equivalents for the used GPIO pins (dictionary)
      * x2: simplemodbus device
      * mbRetries: number of times to retry sending a Modbus command (integer)
      * mbDictName: the dictionary name that holds the desired parameter’s register and number of registers (string)
      * clearText: the display text that should appear in the phrase “the \_\_\_ was successfully/not successfully enabled/disabled” (string)
    - Output:
      * Pass” or “Fail - …” status of the test (string)
  + valueRangeCheck(level,threshold,read)
    - Use: Tests if a provided value is within a given range of a reference value
    - Inputs:
      * level: the reference level to compare agains
      * threshold: the amount of +/- tolerance that is allowable
      * read: the variable value to compare against level
    - Output:
      * A list with the following values (list)
        + [Element 0]=True if it passes or False if it does not pass (bool)
        + [Element 1]= Pass” or “Fail - …” status of the test (string)
  + wifiNetworkSearch(wifiNetwork,wifiRetries,sleepSec=2)
    - Use: Searches for a string of text in all of the RPi’s available Wi-Fi networks
    - Inputs:
      * wifiNetwork: the name to scan for (string)
      * wifiRetries: the number of times to attempt to scan for new networks in range and check for a matching name (integer)
      * sleepSec: the number of seconds to sleep between retries [default is 2] (integer)
    - Output:
      * Pass” or “Fail - …” status of the search results (string)