RTOS Functional Test Procedure
Revision 14
PCA 1261320

Revision Table

Revision	Changes	Engineer	Date
11	Update Firmware REV Check Update Membrane LED test Add test for Flash 2 Update Propel Interface Test	SHB / BAP	2022/07/25
12	Update CAN test with scrub board simulation option Add section Switch Matrix (alternate 1)	BAP	2022/07/29
13	Update EEPROM test parameter Update resistor value for the LED test and voltage to 3.3V	BAP	2022/08/10
14	Update acceptance criteria in step 15 of Propel Interface test	BAP	2023/04/11
15	Update acceptance criteria in steps 5, 9, 13, 17, 21, and 15 in Low Side Driver Output Test	ALM	2025/05/12

Setup:

A fixture has connections with the ChUI user interface for the membrane, and with pogo pins for all other necessary points. Test commands are sent to the board using the serial debug interface. This is a logic level UART with the baud rate set to 115200. The board executes the commands and returns status over the same interface. A VW scrub board is also included in the fixture and is connected to the UI board via the CAN interface.

Test Sequence:

1. Install user interface assembly in machine.
2. Turn power on. Power supply is set to 24.0V and should be capable of driving a 1A load. Power is applied as indicated: COM goes to J7-8. +24V goes to J7-10.
3. Connect the USB cable from test PC to UUT ChUI board.
4. ChUI board must enumerate on the PC as “new T350”.
5. Set to functional test mode by sending the following command over the debug interface: “machine t=1”.

Firmware Rev Check:

1. Read the firmware revision by sending the following command over the debug interface: **“system ?”** UUT will return a response similar to the following:

```
>system ?

sysmgr_Software_Version(cM_USER_INTERFACE) = 1.2.0.221.FCT
```

sysmgr_Hardware_Version(cM_USER_INTERFACE) = 1.00
sysmgr_Software_Version(cM_SCRUB_CONTROLLER) = N/A
sysmgr_Hardware_Version(cM_SCRUB_CONTROLLER) = N/A
sysmgr_Software_Version(cM_LION_PACK_BMS) = N/A
sysmgr_Hardware_Version(cM_LION_PACK_BMS) = N/A
sysmgr_Software_Version(cM_SPE_CHARGER) = N/A
sysmgr_Software_Version(cM_ECH2O) = N/A
sysmgr_Hardware_Version(cM_ECH2O) = N/A
sysmgr_Software_Version(cM_BATTERY_WATERING) = N/A
sysmgr_Hardware_Version(cM_BATTERY_WATERING) = N/A

2. The User Interface Hardware Version shall match the revision number listed in the released drawing of 1261320. The User Interface Software Version may differ. Also check the Scrub or Sweep Controller Software Version against the correct revision. This confirms CAN interface operation.

   If no scrub board is in the fixture, it is necessary to simulate the scrub board CAN commands via a computer.

   1. Test the UI transmit capability. Read CAN message ID 0x701 via a computer. This is the UI heartbeat message and shows the operational state, the data field shall have a non-zero value.
   2. Send the CAN message 0x182 with a data field of 0x 00 10 60 00 00 00 00 00 via a computer. Message interval is every 1 second. This sends the scrub motor current to the UI board.
   3. Read the scrub motor current by sending the following command via the debug interface: **“motor ?”.**
   4. DUT will return a response similar to:

      IL_Motor_Is_Running() = False:0
      IL_Motor_Current_Raw() = 0.0
      IL_Motor_Current_Offset() = 0.0
      IL_Motor_Current() = 9.6
      IL_Motor_Current_Average() = 0.0
      IL_Motor_Duty_Cycle() = 0.0
      IL_Motor_Clipped_Pulses() = 0
      IL_Motor_Clipped_Pulses_Current_Limit_Threshold() = 0
      IL_Motor_Clipped_Pulses_Short_Threshold() = 0
      IL_Motor_Battery_Voltage() = 0.00
      IL_Motor_Current_Threshold() = 0.0
      IL_Motor_Fault_Status() = 32
        Fault: OVER_CURRENT_2
      IL_Motor_State() = MOTOR_POWER_UP:0

5. The response must include:

   IL_Motor_Current() = 9.6

Serial Flash Test:

1. Write test string to the serial flash by sending the following command over the debug interface: "**dlog w=1**".
2. Read the data back from the serial flash by sending the following command over the debug interface: "**dlog r=0**". The response will look similar to the following:

   >dlog r=0

   dataLogger_Read_Log()
   0x00006ABB [0x06] 0x6004CF6C Monitor inserted test message for diagnostic purposes.

3. Clear the test string by sending the following command over the debug interface: "**dlog c**".

Initial Power/Switch Input Tests:

1. J7 pins 2,3,4 should be floating. J7 pin 1 should be at 24V.
2. Read status inputs by sending the following command over the debug interface: **"inputs ?".** UUT will return a response similar to the following:

   >inputs ?

   inputs_State_Raw(J7-03 IN_SW_1:**XX**) = False
   inputs_State (J7-03 IN_SW_1:**XX**) = False
   inputs_State_Raw(J7-01 IN_SW_3:**XX**) = False
   inputs_State (J7-01 IN_SW_3:**XX**) = False
   inputs_State_Raw(J7-02 IN_SW_5:**XX**) = False
   inputs_State (J7-02 IN_SW_5:**XX**) = False
   inputs_State_Raw(J7-04 IN_SW_6:**XX**) = False
   inputs_State (J7-04 IN_SW_6:**XX**) = False
   inputs_State_Raw(J4-04 IN_BAIL_SWITCH:**XX**) = False
   inputs_State (J4-04 IN_BAIL_SWITCH:**XX**) = False

3. All inputs displayed with the Pin Number must return "False". Note there will be other inputs displayed that are from the VW controller whose status in inconsequential. Also number after the colon, denoted with "**XX"** in this document is inconsequential.
4. Connect J7-2 to GND.
5. Read status inputs by sending the following command over the debug interface: **"inputs ?".**
6. Response must include:

   inputs_State_Raw(J7-02 IN_SW_5:**XX**) = True

7. Connect J7-3 to GND. Disconnect J7-2 from GND.
8. Read status inputs by sending the following command over the debug interface: **"inputs ?".**
9. Response must include:

   inputs_State_Raw(J7-03 IN_SW_1:**XX**) = True

10. Connect J7-4 to GND. Disconnect J7-3 from GND.
11. Read status inputs by sending the following command over the debug interface: **“inputs ?”.**
12. Response the debug interface e must include:

    inputs_State_Raw(J7-04 IN_SW_6:**XX**) = True
13. Disconnect J7-4 from GND.
14. Read membrane status inputs by sending the following command over the debug interface: **“inputs m”.** UUT will return a response similar to the following:

    >inputs m

    inputs_State_button(0 , State=False)
    inputs_State_button(1 , State=False)
    inputs_State_button(2 , State=False)
    inputs_State_button(3 , State=False)
    inputs_State_button(4 , State=False)
    inputs_State_button(5 , State=False)
    inputs_State_button(6 , State=False)
    inputs_State_button(7 , State=False)
    inputs_State_button(8 , State=False)
    inputs_State_button(9 , State=False)
    inputs_State_button(10 , State=False)
    inputs_State_button(11 , State=False)
    inputs_State_button(12 , State=False)
    inputs_State_button(13 , State=False)
    inputs_State_button(14 , State=False)
    inputs_State_button(15 , State=False)

15. Connect J8-2 to J8-3. Read status inputs by sending the following command over the debug interface: **“inputs m”.**
16. Response must include leading “0 “State=True” on the same output line. There may be other text in output.

    inputs_State_button(0 , State=True)
17. Disconnect J8-2 to J8-3.
18. Connect J8-3 to J8-4. Read status inputs by sending the following command over the debug interface: **“inputs m”.**
19. Response must include leading “1” and “State=True” on the same output line. There may be other text in output.

    inputs_State_button(1 , State=True)
20. Disconnect J8-3 to J8-4.
21. Connect J8-3 to J8-6. Read status inputs by sending the following command over the debug interface: **“inputs m”.**
22. Response must include leading “2” and “State=True” on the same output line. There may be other text in output.

    inputs_State_button(2 , State=True)
23. Disconnect J8-3 to J8-6.
24. Connect J8-3 to J8-8. Read status inputs by sending the following command over the debug

interface: **“inputs m”.**

25. Response must include leading “3” and “State=True” on the same output line. There may be other text in output.

    inputs_State_button(3 , State=True)

26. Disconnect J8-3 to J8-8.
27. Connect J8-5 to J8-2. Read status inputs by sending the following command over the debug interface: **“inputs m”.**
28. Response must include leading “4” and “State=True” on the same output line. There may be other text in output.

    inputs_State_button(4 , State=True)

29. Disconnect J8-5 to J8-2.
30. Connect J8-5 to J8-4. Read status inputs by sending the following command over the debug interface: **“inputs m”.**
31. Response must include leading “5” and “State=True” on the same output line. There may be other text in output.

    inputs_State_button(5 , State=True)

32. Disconnect J8-5 to J8-4.
33. Connect J8-5 to J8-6. Read status inputs by sending the following command over the debug interface: **“inputs m”.**
34. Response must include leading “6” and “State=True” on the same output line. There may be other text in output.

    inputs_State_button(6 , State=True)

35. Disconnect J8-5 to J8-6.
36. Connect J8-5 to J8-8. Read status inputs by sending the following command over the debug interface: **“inputs m”.**
37. Response must include leading “7” and “State=True” on the same output line. There may be other text in output.

    inputs_State_button(7 , State=True)

38. Disconnect J8-5 to J8-8.
39. Connect J8-7 to J8-2. Read status inputs by sending the following command over the debug interface: **“inputs m”.**
40. Response must include leading “8” and “State=True” on the same output line. There may be other text in output.

    inputs_State_button(8 , State=True)

41. Disconnect J8-7 to J8-2.
42. Connect J8-7 to J8-4. Read status inputs by sending the following command over the debug interface: **“inputs m”.**
43. Response must include leading “9” and “State=True” on the same output line. There may be other text in output.

    inputs_State_button(9 , State=True)

44. Disconnect J8-7 to J8-4.
45. Connect J8-7 to J8-6. Read status inputs by sending the following command over the debug interface: **“inputs m”.**
46. Response must include leading “10” and “State=True” on the same output line. There may be other text in output.

inputs_State_button(10 , State=True)

47. Disconnect J8-7 to J8-6.
48. Connect J8-7 to J8-8. Read status inputs by sending the following command over the debug interface: **"inputs m".**
49. Response must include leading "11" and "State=True" on the same output line. There may be other text in output.

    inputs_State_button(11 , State=True)

50. Disconnect J8-7 to J8-8.
51. Connect J8-9 to J8-2. Read status inputs by sending the following command over the debug interface: **"inputs m".**
52. Response must include leading "12" and "State=True" on the same output line. There may be other text in output.

    inputs_State_button(12 , State=True)

53. Disconnect J8-9 to J8-2.
54. Connect J8-9 to J8-4. Read status inputs by sending the following command over the debug interface: **"inputs m".**
55. Response must include leading "13" and "State=True" on the same output line. There may be other text in output.

    inputs_State_button(13 , State=True)

56. Disconnect J8-9 to J8-4.
57. Connect J8-9 to J8-6. Read status inputs by sending the following command over the debug interface: **"inputs m".**
58. Response must include leading "14" and "State=True" on the same output line. There may be other text in output.

    inputs_State_button(14 , State=True)

59. Disconnect J8-9 to J8-6.
60. Connect J8-9 to J8-8. Read status inputs by sending the following command over the debug interface: **"inputs m".**
61. Response must include leading "15" and "State=True" on the same output line. There may be other text in output.

    inputs_State_button(15 , State=True)

62. Disconnect J8-9 to J8-8.
63. Read power inputs by sending the following command over the debug interface: **"power ?".** UUT will return a response similar to the following:

    power ?

    power_Battery_Level() = 86
    power_Battery_Is_Empty() = False:0
    power_ADC_Battery_Level() = 100
    power_propel_BDI_Battery_Level() = 100
    power_Battery_Is_Mimimal_One_Step() = False:0
    power_Battery_Is_Mimimal_Hopper() = False:0
    power_Battery_Voltage() = 23.94
    hw_Power_Battery_Voltage() = 23.96
    power_Charger_Is_On_Board() = False:0

power_Source() = POWER_SOURCE_KEYSWITCH:1
power_Battery_Type() = BAT_TYPE_LEAD_ACID:1

64. Battery voltage will be returned. Battery voltage reading must both reflect actual (24.0V) within ± 1%.
65. Apply +24V power to J7-9. Remove power from J7-10.
66. Wait 10 seconds.
67. Read power inputs by sending the following command over the debug interface: **"power ?".** UUT will return a response similar to the following:

    power ?

    power_Battery_Level() = 90
    power_Battery_Is_Empty() = False:0
    power_ADC_Battery_Level() = 100
    power_propel_BDI_Battery_Level() = 100
    power_Battery_Is_Mimimal_One_Step() = False:0
    power_Battery_Is_Mimimal_Hopper() = False:0
    power_Battery_Voltage() = 23.97
    hw_Power_Battery_Voltage() = 3.54
    power_Charger_Is_On_Board() = False:0
    power_Source() = POWER_SOURCE_CHARGER:2
    power_Battery_Type() = BAT_TYPE_LEAD_ACID:1

68. Battery voltage will be returned. "power_Battery_Voltage()" reading must both reflect actual (24.0V) within ± 1%.
69. Apply +24V power to J7-10. Remove power from J7-9.
70. Set J7-1 to 24V.
71. Read status inputs by sending the following command over the debug interface: **"inputs ?".**
72. Response must include:

    inputs_State_Raw(J7-01 IN_SW_3:**XX**) = False

73. Set J7-1 to 0V.

74. Response must include:

    inputs_State_Raw(J7-01 IN_SW_3:**XX**) = True

Switch Matrix Test (Alternate 1):

This test can be run instead of items 14 through 62 in section "Initial Power/Switch Input Tests".

1. Read membrane status inputs by sending the following command over the debug interface: **"inputs m".** UUT will return a response similar to the following:

    >inputs m

    inputs_State_button(0 , State=False)
    inputs_State_button(1 , State=False)

```
inputs_State_button( 2 , State=False )
inputs_State_button( 3 , State=False )
inputs_State_button( 4 , State=False )
inputs_State_button( 5 , State=False )
inputs_State_button( 6 , State=False )
inputs_State_button( 7 , State=False )
inputs_State_button( 8 , State=False )
inputs_State_button( 9 , State=False )
inputs_State_button( 10 , State=False )
inputs_State_button( 11 , State=False )
inputs_State_button( 12 , State=False )
inputs_State_button( 13 , State=False )
inputs_State_button( 14 , State=False )
inputs_State_button( 15 , State=False )
```

2. Connect J8-2 to J8-3. Read status inputs by sending the following command over the debug interface: **"inputs m".**
3. Response must include leading "0 "State=True" on the same output line. There may be other text in output.

```
inputs_State_button( 0 , State=True )
```

4. Disconnect J8-2 to J8-3.
5. Connect J8-5 to J8-4. Read status inputs by sending the following command over the debug interface: **"inputs m".**
6. Response must include leading "5" and "State=True" on the same output line. There may be other text in output.

```
inputs_State_button( 5 , State=True )
```

7. Disconnect J8-5 to J8-4.
8. Connect J8-7 to J8-6. Read status inputs by sending the following command over the debug interface: **"inputs m".**
9. Response must include leading "10" and "State=True" on the same output line. There may be other text in output.

```
inputs_State_button( 10 , State=True )
```

10. Disconnect J8-7 to J8-6.
11. Connect J8-9 to J8-8. Read status inputs by sending the following command over the debug interface: **"inputs m".**
12. Response must include leading "15" and "State=True" on the same output line. There may be other text in output.

```
inputs_State_button( 15 , State=True )
```

13. Disconnect J8-9 to J8-8.

Propel Interface Test:

1. Connect J5-3 to J5-4 to create a loopback on the Propel serial interface.

2. Do the UART loopback test by sending the following command over the debug interface:

   >idi l=1

   idi_Set_Loopback_Test(True)
3. After a second, read the result:

   >idi ?

   idi_Error() = PGER_NO_COMMUNICATIONS:65535
   idi_Status_Get() = 0
   idi_Loopback_Result() = True:1
   idi_Read_Passive_Tag(IDPT_STATUS) = 0
   idi_Read_Passive_Tag(IDPT_ERROR) = 65535
   idi_Read_Passive_Tag(IDPT_ANLG_TILLER_FLOAT) = 0
   idi_Read_Passive_Tag(IDPT_ANLG_SPEED_FLOAT) = 0
   idi_Read_Passive_Tag(IDPT_ANLG_THROTTLE_DEMAND) = 0
   idi_Read_Passive_Tag(IDPT_ANLG_MOTOR_CURRENT) = 0

4. Connect J4-1 to J4-4 to simulate the bail switch being closed.
5. Read status inputs by sending the following command over THE DEBUG INTERFACE: **"inputs ?"**.
6. Response must include:

   inputs_State_Raw(J4-04 IN_BAIL_SWITCH:**XX**) = True
7. Disconnect J4-1 from J4-4.
8. Set to functional test mode by sending the following command over THE DEBUG INTERFACE: "machine t=1".
9. Apply 4.0V to J4-4.
10. Set the bail attenuator to part scale by sending the following command over THE DEBUG INTERFACE: **"**propel b=500**".**
11. Measure the voltage at J5-2 output. Voltage must measure 1.7V±5%.
12. Set the bail attenuator to zero by sending the following command over THE DEBUG INTERFACE: **"**propel b=0**".**
13. Measure the voltage at J5-2 output. Voltage must measure 0V±0.4V.
14. Test current limiting bail switch by applying a 50Ω 1W load between J4-1 and common.
15. Measure pin J4-1 WRT common with a DMM. Must measure 2.34V±0.4V.
16. Remove load from J4-1.

EEPROM Test:

1. Write board serial number (where **[SN]** is the serial number on the board) to EEPROM by sending the following command over THE DEBUG INTERFACE: **config p=499,[SN]**
   Response must include:

   Write Complete.
2. Wait 5 seconds and cycle power to board.
3. Read the board serial number from flash by sending the following command over THE DEBUG INTERFACE: **"config r=499"**
4. Response must include the serial number written to the board.
5. Set to functional test mode by sending the following command over THE DEBUG INTERFACE: "machine t=1".

Low Side Driver Output Test:

1. Connect a 1200 ohm 1 watt resistor load from the following:
    a. J7-5 to J7-10
    b. J7-6 to J7-10
    c. J7-7 to J7-10
    d. J10-6 to J7-10
    e. J10-7 to J7-10
    f. J10-8 to J7-10

2. Turn on low side driver 0 output by sending the following command over THE DEBUG INTERFACE: **"lsd a=1"**.
3. Confirm the driver turned on by measuring J7-5. J7-5 must measure less than 1V wrt GND.
4. Turn off low side driver 0 output by sending the following command over THE DEBUG INTERFACE: **"lsd a=0"**.
5. Confirm the driver turned off by measuring J7-5. J7-5 must measure more than 19V wrt GND.
6. Turn on low side driver 1 output by sending the following command over THE DEBUG INTERFACE: **"lsd b=1"**.
7. Confirm the driver turned on by measuring J7-6. J7-6 must measure less than 1V wrt GND.
8. Turn off low side driver 1 output by sending the following command over THE DEBUG INTERFACE: **"lsd b=0"**.
9. Confirm the driver turned off by measuring J7-6. J7-6 must measure more than 19V wrt GND.
10. Turn on low side driver 2 output by sending the following command over THE DEBUG INTERFACE: **"lsd c=1"**.
11. Confirm the driver turned on by measuring J7-7. J7-7 must measure less than 1V wrt GND.
12. Turn off low side driver 2 output by sending the following command over THE DEBUG INTERFACE: **"lsd c=0"**.
13. Confirm the driver turned off by measuring J7-7. J7-7 must measure more than 19V wrt GND.
14. Turn on low side driver 3 output by sending the following command over THE DEBUG INTERFACE: **"lsd x=1"**.
15. Confirm the driver turned on by measuring J10-6. J10-6 must measure less than 1V wrt GND.
16. Turn off low side driver 3 output by sending the following command over THE DEBUG INTERFACE: **"lsd x=0"**.
17. Confirm the driver turned off by measuring J10-6. J10-6 must measure more than 19V wrt GND.
18. Turn on low side driver 4 output by sending the following command over THE DEBUG INTERFACE: **"lsd y=1"**.
19. Confirm the driver turned on by measuring J10-7. J10-7 must measure less than 1V wrt GND.
20. Turn off low side driver 4 output by sending the following command over THE DEBUG INTERFACE: **"lsd y=0"**.
21. Confirm the driver turned off by measuring J10-7. J10-7 must measure more than 19V wrt GND.
22. Turn on low side driver 5 output by sending the following command over THE DEBUG INTERFACE: **"lsd z=1"**.
23. Confirm the driver turned on by measuring J10-8. J10-8 must measure less than 1V wrt GND.
24. Turn off low side driver 5 output by sending the following command over THE DEBUG INTERFACE: **"lsd z=0"**.

25. Confirm the driver turned off by measuring J10-8. J10-8 must measure more than 19V wrt GND.

Membrane LED Test:

1. The test setup should have a 1.5k ohm load resistor from each LED output LED1 through LED32 to 3.3V.
2. Turn off all LEDs by sending the following command over THE DEBUG INTERFACE: **"led o=0"**
3. Confirm that all membrane panel LEDs are turned off by sending the following command over THE DEBUG INTERFACE: **"led o=0"**.
   The response must include (later versions may include additional output from this command):

```
>led o=0

led_Set(0, 0:LED_MODE_OFF, False )
led_Set(1, 0:LED_MODE_OFF, False )
led_Set(2, 0:LED_MODE_OFF, False )
led_Set(3, 0:LED_MODE_OFF, False )
led_Set(4, 0:LED_MODE_OFF, False )
led_Set(5, 0:LED_MODE_OFF, False )
led_Set(6, 0:LED_MODE_OFF, False )
led_Set(7, 0:LED_MODE_OFF, False )
led_Set(8, 0:LED_MODE_OFF, False )
led_Set(9, 0:LED_MODE_OFF, False )
led_Set(10, 0:LED_MODE_OFF, False )
led_Set(11, 0:LED_MODE_OFF, False )
led_Set(12, 0:LED_MODE_OFF, False )
led_Set(13, 0:LED_MODE_OFF, False )
led_Set(14, 0:LED_MODE_OFF, False )
led_Set(15, 0:LED_MODE_OFF, False )
led_Set(16, 0:LED_MODE_OFF, False )
led_Set(17, 0:LED_MODE_OFF, False )
led_Set(18, 0:LED_MODE_OFF, False )
led_Set(19, 0:LED_MODE_OFF, False )
led_Set(20, 0:LED_MODE_OFF, False )
led_Set(21, 0:LED_MODE_OFF, False )
led_Set(22, 0:LED_MODE_OFF, False )
led_Set(23, 0:LED_MODE_OFF, False )
led_Set(24, 0:LED_MODE_OFF, False )
led_Set(25, 0:LED_MODE_OFF, False )
led_Set(26, 0:LED_MODE_OFF, False )
led_Set(27, 0:LED_MODE_OFF, False )
led_Set(28, 0:LED_MODE_OFF, False )
led_Set(29, 0:LED_MODE_OFF, False )
led_Set(30, 0:LED_MODE_OFF, False )
led_Set(31, 0:LED_MODE_OFF, False )
led_Set(32, 0:LED_MODE_OFF, False )
```

led_Set(33, 0:LED_MODE_OFF, False)
led_Set(34, 0:LED_MODE_OFF, False)
led_Set(35, 0:LED_MODE_OFF, False)
led_Set(36, 0:LED_MODE_OFF, False)
led_Set(37, 0:LED_MODE_OFF, False)
led_Set(38, 0:LED_MODE_OFF, False)
led_Set(39, 0:LED_MODE_OFF, False)
led_Set(40, 0:LED_MODE_OFF, False)
led_Set(41, 0:LED_MODE_OFF, False)
led_Set(42, 0:LED_MODE_OFF, False)
led_Set(43, 0:LED_MODE_OFF, False)
led_Set(44, 0:LED_MODE_OFF, False)
led_Set(45, 0:LED_MODE_OFF, False)
led_Set(46, 0:LED_MODE_OFF, False)
led_Set(47, 0:LED_MODE_OFF, False)
led_Set(48, 0:LED_MODE_OFF, False)
led_Set(49, 0:LED_MODE_OFF, False)
led_Set(50, 0:LED_MODE_OFF, False)
led_Set(51, 0:LED_MODE_OFF, False)
led_Set(52, 0:LED_MODE_OFF, False)
led_Set(53, 0:LED_MODE_OFF, False)
led_Set(54, 0:LED_MODE_OFF, False)
led_Set(55, 0:LED_MODE_OFF, False)
led_Set(56, 0:LED_MODE_OFF, False)
led_Set(57, 0:LED_MODE_OFF, False)
led_Set(58, 0:LED_MODE_OFF, False)
led_Set(59, 0:LED_MODE_OFF, False)
led_Set(60, 0:LED_MODE_OFF, False)
led_Set(61, 0:LED_MODE_OFF, False)

4. Confirm that J8-2,3,4,5,6,7,8,9 and J9-2,3,4,5,6,7,8,9,10,12,13,14,15,16,17,18,19 are all turned off.
5. Turn on LED 1 by sending the **“led x=1”** command over THE DEBUG INTERFACE.
   The response must include:
   led_Set(47, 1:LED_MODE_ON, False)
6. Confirm that J9-21 is turned on.
7. Turn off LED 1 by sending the **“led y=1”** command over THE DEBUG INTERFACE.
   The response must include:
   led_Set(47, 1:LED_MODE_OFF, False)
8. Confirm that J9-21 is turned off.
9. Turn on LED 2 by sending the **“led x=2”** command over THE DEBUG INTERFACE.
   The response must include:

led_Set(30, 1:LED_MODE_ON, False)

10. Confirm that J9-20 is turned on.
11. Turn off LED 2 by sending the **“led y=2”** command over THE DEBUG INTERFACE. The response must include:

    led_Set(30, 1:LED_MODE_OFF, False)

12. Confirm that J9-20 is turned off.
13. Turn on LED 3 by sending the **“led x=3”** command over THE DEBUG INTERFACE. The response must include:

    led_Set(25, 1:LED_MODE_ON, False)

14. Confirm that J9-19 is turned on.
15. Turn off LED 3 by sending the **“led y=3”** command over THE DEBUG INTERFACE. The response must include:

    led_Set(25, 1:LED_MODE_OFF, False)

16. Confirm that J9-19 is turned off.
17. Turn on LED 4 by sending the **“led x=4”** command over THE DEBUG INTERFACE. The response must include:

    led_Set(26, 1:LED_MODE_ON, False)

18. Confirm that J9-18 is turned on.
19. Turn off LED 4 by sending the **“led y=4”** command over THE DEBUG INTERFACE. The response must include:

    led_Set(26, 1:LED_MODE_OFF, False)

20. Confirm that J9-18 is turned off.
21. Turn on LED 5 by sending the **“led x=5”** command over THE DEBUG INTERFACE. The response must include:

    led_Set(15, 1:LED_MODE_ON, False)

22. Confirm that J9-17 is turned on.
23. Turn off LED 5 by sending the **“led y=5”** command over THE DEBUG INTERFACE. The response must include:

    led_Set(15, 1:LED_MODE_OFF, False)

24. Confirm that J9-17 is turned off.
25. Turn on LED 6 by sending the **“led x=6”** command over THE DEBUG INTERFACE. The response must include:

    led_Set(14, 1:LED_MODE_ON, False)

26. Confirm that J9-16 is turned on.
27. Turn off LED 6 by sending the **“led y=6”** command over THE DEBUG INTERFACE. The response must include:

    led_Set(14, 1:LED_MODE_OFF, False)

28. Confirm that J9-16 is turned off.
29. Turn on LED 7 by sending the **“led x=7”** command over THE DEBUG INTERFACE. The response must include:

    led_Set(22, 1:LED_MODE_ON, False)

30. Confirm that J9-15 is turned on.

31. Turn off LED 7 by sending the **“led y=7”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(22, 1:LED_MODE_OFF, False)
32. Confirm that J9-15 is turned off.
33. Turn on LED 8 by sending the **“led x=8”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(21, 1:LED_MODE_ON, False)
34. Confirm that J9-14 is turned on.
35. Turn off LED 8 by sending the **“led y=8”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(21, 1:LED_MODE_OFF, False)
36. Confirm that J9-14 is turned off.
37. Turn on LED 9 by sending the **“led x=9”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(20, 1:LED_MODE_ON, False)
38. Confirm that J9-13 is turned on.
39. Turn off LED 9 by sending the **“led y=9”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(20, 1:LED_MODE_OFF, False)
40. Confirm that J9-13 is turned off.
41. Turn on LED 10 by sending the **“led x=10”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(6, 1:LED_MODE_ON, False)
42. Confirm that J9-12 is turned on.
43. Turn off LED 10 by sending the **“led y=10”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(6, 1:LED_MODE_OFF, False)
44. Confirm that J9-12 is turned off.
45. Turn on LED 11 by sending the **“led x=11”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(7, 1:LED_MODE_ON, False)
46. Confirm that J9-11 is turned on.
47. Turn off LED 11 by sending the **“led y=11”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(7, 1:LED_MODE_OFF, False)
48. Confirm that J9-11 is turned off.
49. Turn on LED 12 by sending the **“led x=12”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(8, 1:LED_MODE_ON, False)
50. Confirm that J9-10 is turned on.
51. Turn off LED 12 by sending the **“led y=12”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(8, 1:LED_MODE_OFF, False)
52. Confirm that J9-10 is turned off.

53. Turn on LED 13 by sending the **“led x=13”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(9, 1:LED_MODE_ON, False)
54. Confirm that J9-09 is turned on.
55. Turn off LED 13 by sending the **“led y=13”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(9, 1:LED_MODE_OFF, False)
56. Confirm that J9-09 is turned off.
57. Turn on LED 14 by sending the **“led x=14”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(10, 1:LED_MODE_ON, False)
58. Confirm that J9-08 is turned on.
59. Turn off LED 14 by sending the **“led y=14”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(10, 1:LED_MODE_OFF, False)
60. Confirm that J9-08 is turned off.
61. Turn on LED 15 by sending the **“led x=15”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(31, 1:LED_MODE_ON, False)
62. Confirm that J9-07 is turned on.
63. Turn off LED 15 by sending the **“led y=15”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(31, 1:LED_MODE_OFF, False)
64. Confirm that J9-07 is turned off.
65. Turn on LED 16 by sending the **“led x=16”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(11, 1:LED_MODE_ON, False)
66. Confirm that J9-06 is turned on.
67. Turn off LED 16 by sending the **“led y=16”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(11, 1:LED_MODE_OFF, False)
68. Confirm that J9-06 is turned off.
69. Turn on LED 17 by sending the **“led x=17”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(28, 1:LED_MODE_ON, False)
70. Confirm that J9-05 is turned on.
71. Turn off LED 17 by sending the **“led y=17”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(28, 1:LED_MODE_OFF, False)
72. Confirm that J9-05 is turned off.

73. Turn on LED 18 by sending the **“led x=18”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(29, 1:LED_MODE_ON, False)
74. Confirm that J9-04 is turned on.
75. Turn off LED 18 by sending the **“led y=18”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(29, 1:LED_MODE_OFF, False)
76. Confirm that J9-04 is turned off.
77. Turn on LED 19 by sending the **“led x=19”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(12, 1:LED_MODE_ON, False)
78. Confirm that J9-03 is turned on.
79. Turn off LED 19 by sending the **“led y=19”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(12, 1:LED_MODE_OFF, False)
80. Confirm that J9-03 is turned off.
81. Turn on LED 20 by sending the **“led x=20”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(13, 1:LED_MODE_ON, False)
82. Confirm that J8-22 is turned on.
83. Turn off LED 20 by sending the **“led y=20”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(13, 1:LED_MODE_OFF, False)
84. Confirm that J8-22 is turned off.
85. Turn on LED 21 by sending the **“led x=21”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(46, 1:LED_MODE_ON, False)
86. Confirm that J8-21 is turned on.
87. Turn off LED 21 by sending the **“led y=21”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(46, 1:LED_MODE_OFF, False)
88. Confirm that J8-21 is turned off.
89. Turn on LED 22 by sending the **“led x=22”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(19, 1:LED_MODE_ON, False)
90. Confirm that J8-20 is turned on.
91. Turn off LED 22 by sending the **“led y=22”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(19, 1:LED_MODE_OFF, False)
92. Confirm that J8-20 is turned off.
93. Turn on LED 23 by sending the **“led x=23”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(45, 1:LED_MODE_ON, False)
94. Confirm that J8-19 is turned on.

95. Turn off LED 23 by sending the **“led y=23”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(45, 1:LED_MODE_OFF, False)
96. Confirm that J8-19 is turned off.
97. Turn on LED 24 by sending the **“led x=24”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(34, 1:LED_MODE_ON, False)
98. Confirm that J8-18 is turned on.
99. Turn off LED 24 by sending the **“led y=24”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(34, 1:LED_MODE_OFF, False)
100. Confirm that J8-18 is turned off.
101. Turn on LED 25 by sending the **“led x=25”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(56, 1:LED_MODE_ON, False)
102. Confirm that J8-17 is turned on.
103. Turn off LED 25 by sending the **“led y=25”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(56, 1:LED_MODE_OFF, False)
104. Confirm that J8-17 is turned off.
105. Turn on LED 26 by sending the **“led x=26”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(57, 1:LED_MODE_ON, False)
106. Confirm that J8-16 is turned on.
107. Turn off LED 26 by sending the **“led y=26”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(57, 1:LED_MODE_OFF, False)
108. Confirm that J8-16 is turned off.
109. Turn on LED 27 by sending the **“led x=27”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(58, 1:LED_MODE_ON, False)
110. Confirm that J8-15 is turned on.
111. Turn off LED 27 by sending the **“led y=27”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(58, 1:LED_MODE_OFF, False)
112. Confirm that J8-15 is turned off.
113. Turn on LED 28 by sending the **“led x=28”** command over THE DEBUG INTERFACE.
    The response must include:
    led_Set(59, 1:LED_MODE_ON, False)
114. Confirm that J8-14 is turned on.

115. Turn off LED 28 by sending the **“led y=28”** command over THE DEBUG INTERFACE. The response must include:

led_Set(59, 1:LED_MODE_OFF, False)

116. Confirm that J8-14 is turned off.
117. Turn on LED 29 by sending the **“led x=29”** command over THE DEBUG INTERFACE. The response must include:

led_Set(60, 1:LED_MODE_ON, False)

118. Confirm that J8-13 is turned on.
119. Turn off LED 29 by sending the **“led y=29”** command over THE DEBUG INTERFACE. The response must include:

led_Set(60, 1:LED_MODE_OFF, False)

120. Confirm that J8-13 is turned off.
121. Turn on LED 30 by sending the **“led x=30”** command over THE DEBUG INTERFACE. The response must include:

led_Set(61, 1:LED_MODE_ON, False)

122. Confirm that J8-12 is turned on.
123. Turn off LED 30 by sending the **“led y=30”** command over THE DEBUG INTERFACE. The response must include:

led_Set(61, 1:LED_MODE_OFF, False)

124. Confirm that J8-12 is turned off.
125. Turn on LED 31 by sending the **“led x=31”** command over THE DEBUG INTERFACE. The response must include:

led_Set(62, 1:LED_MODE_ON, False)

126. Confirm that J8-11 is turned on.
127. Turn off LED 31 by sending the **“led y=31”** command over THE DEBUG INTERFACE. The response must include:

led_Set(62, 1:LED_MODE_OFF, False)

128. Confirm that J8-11 is turned off.
129. Turn on LED 32 by sending the **“led x=32”** command over THE DEBUG INTERFACE. The response must include:

led_Set(63, 1:LED_MODE_ON, False)

130. Confirm that J8-10 is turned on.
131. Turn off LED 32 by sending the **“led y=32”** command over THE DEBUG INTERFACE. The response must include:

led_Set(63, 1:LED_MODE_OFF, False)

132. Confirm that J8-10 is turned off.

END OF TEST.