Test Station Design Document

PB540/PB560 CPU PCB

Revision History

Revision	Change Order	Author	Approved By	Approval Date	Summary of Change
00	ECO-R163871	Bill Hurd	Refer to Agile	Refer to Agile	Initial Draft Release
A	ECO-R167338	Bill Hurd	Refer to Agile	Refer to Agile	Initial Release
B	ECO-R181354	Bill Hurd	Refer to Agile	Refer to Agile	Update to add new PCB P/N 3845800 in Section 1.1 and references to PB560
C	ECO-R180285	Bill Hurd	Refer to Agile	Refer to Agile	Add missing reference to fixed resistor for turbine temperature test in Sec 2.2.5/Appendix B.

Table of Contents

1.0 Introduction

1.1 Purpose

The purpose of this document is to describe the test hardware and software used for the functional testing of the PB540/PB560 CPU PCB P/N 3821300/3845800.

1.2 Scope

This document applies to the PB540/PB560 CPU PCB Test Station using test procedure INS-CQ-66.

1.3 References

[1] PB540/PB560 CPU PCB, Test Station Requirements Doc, p/n 10021586
[2] PB540/PB560 CPU PCB, Manufacturing Test Specification Doc, p/n 10020880

1.4 Definitions, Acronyms and Abbreviations

UUT	Unit under Test
PCBA	Printed Circuit Board Assembly
TSRQDOC	Test Station Requirements Documents
MTS	Manufacturing Test Specification
DVM	Digital Volt Meter
EEPROM	Electrically Erasable Programmable Read Only Memory
()	denotes software function
TRQ	Test Requirement

2.0 Test Station Hardware Design

The PB540/PB560 CPU PCB Functional Test Station has been developed around using the PB540/PB560 product components including PCBs and the ventilator case. The hardware for this test station and the function of each hardware component is described in the following sections. Tracing to the particular sections of the Test Station Requirements Document is given in the headings [TSRQDOC X.XX].

2.1 Test Station PC [TSRQDOC 2.3]

The Test Station PC is a Pentium computer. The Test Station PC is used to download the test software to the UUT requiring a USB Type A/Mini B cable between the PC's USB Port and the UUT USB Port at J1. The Operating System is Win XP.

2.2 Test Fixture [TSRQDOC 2.2]

The test fixture is used to provide a convenient method of mounting, electrical interfacing and providing test access to the PCB during test. The test fixture is the PB540/PB560 product case and contains the following product modules. See Appendix 1 for UUT connections to support testing.

2.2.1 PB540/PB560 Power Management Board [TSRQDOC 2.2.1]

The PB540/PB560 Power Management board is connected in the same manner as the product to supply power and test the CPU-Power Management interface communications. The PB540/PB560 Power Management Board is located permanently in the test fixture and connects to the UUT at J7.

2.2.2 PB540/PB560 AC-DC Power Module [TSRQDOC 2.2.2]

The AC-DC Power Module is the standard product power module and is installed permanently on the Power Management Board in the same manner as the product for providing DC power from the AC mains source.

2.2.3 Keypad [TSRQDOC 2.2.3]

The PB540/PB560 Keypad is connected to the UUT at J1 in the same manner as the product and is used to test the keypad interface and execute the functional tests.

2.2.4 Buzzer Board [TSRQDOC 2.2.4]

The PB540/PB560 product Buzzer board is connected to the UUT in the same manner as the product at J18 for testing the buzzer board interface.

2.2.5 Turbine with Control Board [TSRQDOC 2.2.5]

The PB540/PB560 Turbine assembly including the Turbine Control Board is connected to the UUT in the same manner as the product at J4 for testing the Turbine Interface. On the Turbine Control Board, J3 pin1, 2, a 10KΩ, 1% resistor is used in place of the turbine thermistor for the turbine temperature test (see Appendix B).

2.3 DVM [TSRQDOC 2.1]

A Digital Voltmeter (DVM) is used to measure test point DC voltages as directed by the test program. The DVM Voltage basic accuracy is ±0.3% of reading or better and 4-1/2 digits or more of display.

3 Test Station Software Design

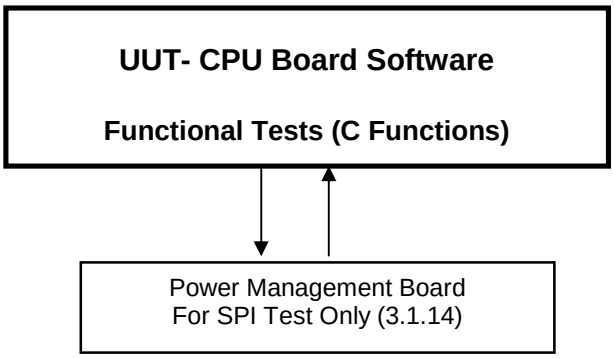
The CPU Board Test Software consists of: UUT test software, Power Management PCB software and the UUT Software download application. Tracing to the particular sections of the Test Station Requirements Document is given in the headings [TSRQDOC X.XX].

The CPU Board Test Software P/N 3832200, contains the functional tests to be executed and is written in the C Language. The UUT software uses the LCD to display the test messages and results and the keypad is used for test operator input and testing the keypad interface. [TSRQDOC 3.1].

The Power Management Board P/N 3833100 software is a derivative of the PB540/PB560 Product Software and described in the product software documentation. The Power Management Board software is only used in this application to test the SPI interface (See 3.1.14). [TSRQDOC 3.2]. The Power Management Board also uses the USB controller software p/n 3835500 which is vendor supplied software used to send data between the PC connected to the Power Management Board and the CPU board for test software download [TSRQDOC 3.2].

The UUT Software download application P/N 3832500 ST10FLASHER is used to download the UUT software at the beginning of the functional test. This application resides on the PC so that it can be executed by clicking on an icon on the PC Desktop. Refer to the ST10FLASHER download application for documentation details. [TSRQDOC 3.3].

The diagram below depicts how the Test Software is structured.



3.1 UUT Test Software

The PC Application Test Software consists of automated and manual tests for the test operator to execute. The testing begins with four DC voltage measurements which are performed manually by the test operator using a DVM. To begin the automated section, the UUT software is downloaded using the PC Download Application ST10Flasher.exe. Refer to the CPU Board Test Procedure INS-CQ-66 for test execution details.

Once the UUT software is downloaded, the CPU test program will start and display the test software version and then a test menu when the test fixture/UUT is powered up. The test menu offers the choice of two languages, French and English, and the selection to run the tests in the partial or complete modes. The test operator selects the desired language and mode. The file Message.c contains the strings for the supported languages used in the testing.

The 'complete' test mode executes all the tests in a batch manner. The 'partial' mode allows the test operator to select an individual test to run. The 'partial' mode does not allow the keypad test to be run.

The following sections are the detailed descriptions of the tests. Note the displayed message text contained in the following sections are translations from French and may not be the exact word content verbatim.

3.1.1 LCD Contrast Test [TSRQDOC, TRQ 3.5.1]

The test calls TEST_reglage_contraste() to perform the following:

- Enter a While loop while the Valid key is detected as pressed
- Call TIM_Temporize(200000) to delay 200mSec
- Enter a While loop while the Valid key is detected as pressed
- Call TIM_Temporize(200000) to delay 200mSec
- Call DisplayBrightness(33) to set the default value Contraste to 33 which - calls DIS_FifoWrite(DRV_BRIGHTNESSAJUST, Value, 0,0,0,0,0,0,0,0,0,0)
- Call DisplayString to write 'Contrast Adjustment' to LCD
- Call DisplayString to write 'Press on the key + to increase contrast' to LCD
- Call DisplayString to write 'Press on the key - to decrease contrast' to LCD
- Call DisplayString to write 'Press on key VALID to continue'
- While the Valid key is not pressed, do the following:
  1. Check for the UP or DOWN key to be pressed.
  2. If the UP key is pressed, increment variable Contraste, Call TIM_Temporize(200000) to delay 200mSec, check Contraste value if greater than 40 – set to 40, Call DisplayBrightness() to set the new Contraste value
  3. If the DOWN key is pressed, decrement variable Contraste, Call TIM_Temporize(200000) to delay 200mSec, check Contraste value if less than or equal to 25 – set to 25, Call DisplayBrightness() to set the new Contraste value
- Enter a While loop while the Valid key is detected as pressed
- Call TIM_Temporize(200000) to delay 200mSec
- Enter a While loop while the Valid key is detected as pressed
- Call TIM_Temporize(200000) to delay 200mSec
- Call DisplayErasePage() to clear the LCD

3.1.2 Keypad Interface Test [TSRQDOC, TRQ 3.5.2]

The test calls TEST_InterfaceClavier () to perform the following:

- Call OUTPUT_PORT_Reset() to turn OFF O2 led
- Call OUTPUT_PORT_Reset() to turn OFF Ventilator led
- Set LED_AL_R to 0 to turn OFF Red Alarm led
- Set LED_AL_O to 0 to turn OFF Orange Alarm led
- Call DisplayString() to display 'Test Keypad Interface'
- Call DisplayString() to display 'Press the Ventilate Key'
- Call TIM_ Temporize() to delay 1 sec

- Wait for PUSHED_KEYS to be non-zero to check for one of the six keys to be recognized as pressed

- If TOUCHE_MA is true for the Ventilate key press, and PUSHED_KEYS is 1 then call BUZ_Beep() to sound security buzzer briefly and Call DisplayString() to display 'Test OK', else: set erreur_clavier to 1, call DisplayString() to display 'Ventilate Button Error', and call BUZ_ThreeBeep()
- Call TIM_ Temporize() to delay .8 sec
- Call DisplayString() to remove all test instruction and result messages

- Call DisplayString() to display 'Press the + Key'
- Wait for PUSHED_KEYS to be non-zero to check for one of the six keys to be recognized as pressed

- If TOUCHE_HAUT is true for the Ventilate key press, and PUSHED_KEYS is 1 then call BUZ_Beep() to sound security buzzer briefly and Call DisplayString() to display 'Test OK', else: set erreur_clavier to 1, call DisplayString() to display '+ Button Error', and call BUZ_ThreeBeep()
- Call TIM_ Temporize() to delay .8 sec
- Call DisplayString() to remove all test instruction and result messages
- Call DisplayString() to display 'Press the - Key'
- Wait for PUSHED_KEYS to be non-zero to check for one of the six keys to be recognized as pressed
- If TOUCHE_BAS is true for the Ventilate key press, and PUSHED_KEYS is 1 then call BUZ_Beep() to sound security buzzer briefly and Call DisplayString() to display 'Test OK', else: set erreur_clavier to 1, call DisplayString() to display '- Button Error', and call BUZ_ThreeBeep()
- Call TIM_ Temporize() to delay .8 sec
- Call DisplayString() to remove all test instruction and result messages

- Call DisplayString() to display 'Press the Valid Key'
- Wait for PUSHED_KEYS to be non-zero to check for one of the six keys to be recognized as pressed
- If TOUCHE_VALID is true for the Ventilate key press, and PUSHED_KEYS is 1 then call BUZ_Beep() to sound security buzzer briefly and Call DisplayString() to display 'Test OK', else: set erreur_clavier to 1, call DisplayString() to display 'Valid Button Error', and call BUZ_ThreeBeep()
- Call TIM_ Temporize() to delay .8 sec
- Call DisplayString() to remove all test instruction and result messages

- Call DisplayString() to display 'Press the Inhib Key'
- Wait for PUSHED_KEYS to be non-zero to check for one of the six keys to be recognized as pressed
- If TOUCHE_INHIB is true for the Ventilate key press, and PUSHED_KEYS is 1 then call BUZ_Beep() to sound security buzzer briefly and Call DisplayString() to display 'Test OK', else: set erreur_clavier to 1, call DisplayString() to display 'INHIB Button Error', and call BUZ_ThreeBeep()
- Call TIM_ Temporize() to delay .8 sec
- Call DisplayString() to remove all test instruction and result messages

- If the DeviceType is SPPORTAIR_M2 then check the O2 button which is not applicable for the PB540/PB560

- Call DisplayString() to display 'Test the Leds' and 'Verify that'
- Call DisplayString() to display 'The red Led is ON'
- Set LED_AL_R to 1 to turn ON Red Alarm led
- Call TIM_ Temporize() to delay 2.5 sec
- Call BUZ_Beep() to sound the security buzzer briefly
- Call DisplayString() to remove the 'The red Led is ON' message
- Call DisplayString() to display 'The red Led is OFF'
- Set LED_AL_R to 0 to turn OFF Red Alarm led
- Call TIM_ Temporize() to delay 2.5 sec
- Call BUZ_Beep() to sound the security buzzer briefly
- Call DisplayString() to remove the 'The red Led is OFF' message

- Call DisplayString() to display 'The orange Led is ON'
- Set LED_AL_R to 1 to turn ON Orange Alarm led
- Call TIM_ Temporize() to delay 2.5 sec
- Call BUZ_Beep() to sound the security buzzer briefly
- Call DisplayString() to remove the 'The red Led is ON' message
- Call DisplayString() to display 'The orange Led is OFF'
- Set LED_AL_O to 0 to turn OFF Orange Alarm led
- Call TIM_ Temporize() to delay 2.5 sec
- Call BUZ_Beep() to sound the security buzzer briefly
- Call DisplayString() to remove the 'The orange Led is OFF' message

- Call DisplayString() to display 'The Ventilate Led is ON'

- Call OUTPUT_PORT_Set to turn ON Ventilate led
- Call TIM_ Temporize() to delay 2.5 sec
- Call BUZ_Beep() to sound the security buzzer briefly
- Call DisplayString() to remove the 'The Ventilate Led is ON' message
- Call DisplayString() to display 'The Ventilate Led is OFF'
- Call OUTPUT_PORT_Reset to turn OFF Ventilate led
- Call TIM_ Temporize() to delay 2.5 sec
- Call BUZ_Beep() to sound the security buzzer briefly
- Call DisplayString() to remove the 'The Ventilate Led is OFF' message

- If the DeviceType is SPPORTAIR_M2 then check the O2 led which is not applicable for the PB540/PB560
- Call DisplayErasePage() to clear LCD screen

3.1.3 RAM Memory Test [TSRQDOC, TRQ 3.5.3]

The test calls TEST_ram() to perform the following:

- Set flag ErreurRam = 0
- Call DisplayString to write 'Checking of the RAM by writing of 0XAAAA' to LCD
- Setup the PtrRamWord pointer to iterate through the memory addresses from 0x200000 to 0x280000, and write 0xAAAA to each location.
- Setup the PtrRamWord pointer to iterate through the memory addresses from 0x200000 to 0x280000, and read each location:
  1. Check read location to be equal to 0xAAAA
  2. If a read location fails, Call DisplayString to write the address and data value that failed, set ErreurRam flag to 1, and set the PtrRamWord pointer to the end address to exit.
- Call TIM_Temporize() to delay 1Sec
- Call DisplayString to write 'Checking of the RAM by writing of 0X5555' to LCD
- Setup the PtrRamWord pointer to iterate through the memory addresses from 0x200000 to 0x280000, and write 0x5555 to each location.
- Setup the PtrRamWord pointer to iterate through the memory addresses from 0x200000 to 0x280000, and read each location:
  1. Check read location to be equal to 0x5555
  2. If a read location fails, Call DisplayString to write the address and data value that failed, set ErreurRam flag to 1, and set the PtrRamWord pointer to the end address to exit.
- Call DisplayString to write 'Checking of the RAM by writing of 0XAA' to LCD
- Setup the PtrRamByte pointer to iterate through the memory addresses from 0x200000 to 0x280000, and write 0xAA to each location.
- Setup the PtrRamByte pointer to iterate through the memory addresses from 0x200000 to 0x280000, and read each location:
  1. Check read location to be equal to 0xAA
  2. If a read location fails, Call DisplayString to write the address and data value that failed, set ErreurRam flag to 1, and set the PtrRamByte pointer to the end address to exit.

- Call DisplayString to write 'Checking of the RAM by writing of 0X55' to LCD
- Setup the PtrRamByte pointer to iterate through the memory addresses from 0x200000 to 0x280000, and write 0x55 to each location.
- Setup the PtrRamByte pointer to iterate through the memory addresses from 0x200000 to 0x280000, and read each location:
  3. Check read location to be equal to 0x55
  4. If a read location fails, Call DisplayString to write the address and data value that failed, set ErreurRam flag to 1, and set the PtrRamByte pointer to the end address to exit.
- Call TIM_Temporize() to delay 1Sec
- If ErreurRam flag = 0, then the RAM test passed, call DisplayString to write 'RAM OK', set BUZ_SEC = 1, then set BUZ_SEC = 0, to sound security buzzer briefly, and call DisplayErasePage() to clear the LCD

- If ErreurRam flag = 1, then the RAM test failed, call DisplayString to write 'RAM DEFECT', set BUZ_SEC = 1 to sound security buzzer, enter forever while loop

3.1.4 Real Time Cock Test [TSRQDOC, TRQ 3.5.4]

The test calls TEST_rtc() to perform the following:

- Set flag ErreurRtc = 0
- Call the DRV_RTC_Init() to do the following:
  1. De-activate SPI: SSCCON **=** 0x0000
  2. De-activate Interrupt: SSCTIC **=** 0x0000
  3. De-activate Interrupt: SSCRIC **=** 0x0000
  4. Configure SPI Clock for 1Mhz : SSCBR **=** 0x0014
  5. Configure in Full Duplex mode and activate SPI: SSCCON **=** 0xF057
  6. Set SCLK and MTSR levels: P3 **|=** 0x2200

- Call BUZ_Beep() to briefly beep security buzzer
- Call DisplayErase Page() to erase screen
- Call DisplayString() to write 'REAL-TIME CLOCK TEST (DS1305)'
- Call DRV_RTC_control() with the STOP_RTC_AND_WRITE parameter to stop RTC
- Write the Rtc data structure members, 0 mSec, 30 Sec, 30 Min, 12 Hour, 06 date, 25 Day, 12 Month, 07 year
- Call RTC_WriteTime() to write the Rtc data structure to the real time clock
- Call DRV_RTC_control() with the RUN_RTC_AND_PROTECT parameter to start the RTC
- Set the members of the RtcSoft structure which is the Software Clock driven by Timer 6 in RTC_SoftLaunch(): RtcSoft**.**Second **=** 30, RtcSoft**.**Minute**=** 30, RtcSoft**.**Hour **=** 12, MilliSeconde **=** 0
- Call DisplayString to write the time header for hours, minutes, seconds for the DS1305 and the simulated clock (software clock) to the LCD
- Call TIM_Temporize() to delay 1Sec
- Perform a loop while the RtcSoft**.**Second is < 35 seconds to do the following:
  1. Call RTC_ReadTime() to update the Rtc data structure members
  2. Call DisplayNumber() to write Rtc.Day to LCD
  3. Call DisplayString to write the Rtc.Month to LCD
  4. Call DisplayNumber() to write Rtc.Year to LCD
  5. Call DisplayNumber() to write Rtc.Hour to LCD
  6. Call DisplayNumber() to write Rtc.Min to LCD
  7. Call DisplayNumber() to write Rtc.Second to LCD
  8. Call DisplayNumber() to write RtcSoft.Hour to LCD
  9. Call DisplayNumber() to write RtcSoft.Min to LCD
  10. Call DisplayNumber() to write RtcSoft.Second to LCD
  11. Check that Rtc.Second = RtcSoft.Second ± 1 sec, else set flag ErreurRtc =1
  12. Check that Rtc.Min = RtcSoft.Min ± 1 min, else set flag ErreurRtc =1
  13. Check that Rtc.Hour = RtcSoft.Hour ± 1 hour, else set flag ErreurRtc =1
  14. Call TIM_ Temporize() to delay 0.2sec
- If there are no errors (ErreurRtc = 0) then call BUZ_Beep() to briefly beep security buzzer, call DisplayString() to write 'RTC Test OK' to the LCD, call TIM_ Temporize() to delay 1.5sec
- If there are errors (ErreurRtc = 1) then call DisplayString() to write 'RTC Error' to the LCD, call Buz_ThreeBeep(), call DisplayString() to write 'Press Valid Key to Continue', loop while waiting for the Valid key press, call TIM_ Temporize() to delay 0.2sec, loop while waiting for the Valid key press, call BUZ_Beep() to briefly sound buzzer
- Call DisplayErasePage() to clear LCD screen

3.1.5 EEPROM Test [TSRQDOC, TRQ 3.5.5]

The test calls TEST_eeprom() to perform the following:

- Set Page variable to 0
- Call DRV_EEP_Init() to perform initialization of the EEPROM registers
- Call DisplayString to write 'Test EEPROM Memory' to LCD
- Call DisplayString to write 'IC32 Parameters' to LCD
- Call EEP_write_eeprom() to:
  1. Call DisplayString to write 'WRITING EEPROM 0xFF on page' to LCD
  2. Call DisplayNumber to write Page variable value to LCD
  3. Iterate from Page 0 to Page 127 performing steps 4-7
  4. Set AddressPage to Page x 32
  5. Send EEPROM instruction to activate chip enable
  6. Send EEPROM write instruction
  7. Iterate through the Address values from AddressPage to (Page +1)x32 to write 0xFF as the data value (AddressPage is Page * 32).
  8. Iterate from Page 0 to Page 127 performing steps 9-12
  9. Set AddressPage to Page x 32
  10. Send EEPROM instruction to activate chip enable
  11. Send EEPROM write instruction
  12. Iterate through the Address values from AddressPage to (Page +1)x32 incrementing by 2, to write the Address as the data value (AddressPage is Page * 32).
  13. Return 0 for no errors
- If EEP_write_eeprom() returns 1 for error, the call BUZ_Beep() to sound security buzzer briefly, Call DisplayString to write 'Error writing EEPROM IC32' to LCD, Call DisplayString to write 'Press Valid key to continue' to LCD, loop while waiting for the Valid key press
- If EEP_write_eeprom() returns 0 for pass, call BUZ_Beep() to sound security buzzer briefly, Call DisplayString to write 'EEPROM IC32 Write OK' to LCD, call TIM_ Temporize() to delay 1sec, Call DisplayString to write 'EEPROM IC32 Write OK' to LCD
- Call EEP_read_eeprom() to read back the address stored in the EEP_write_eeprom(). If the address read is correct return 0, if the address read is not correct return the fail address
- If EEP_read_eeprom() returns 0 for successful read, call Buz_Beep(), call DisplayString() to write 'EEPROM Write OK', call TIM_ Temporize() to delay 2sec
- If EEP_read_eeprom() returns 1 for read error, call Buz_ThreeBeep(), call DisplayString() to write 'EEPROM Read Error', call TIM_ Temporize() to delay 2sec, call DisplayString() to write address that failed, call DisplayString() to write 'Press Valid Key to Continue', wait for Valid key to be pressed
- Call EEP_write_eepromFF() to write 0xFF in all locations to clear the EEPROM memory:
  1. Call DisplayString to write 'WRITING EEPROM 0xFF on page' to LCD
  2. Call DisplayNumber to write Page variable value to LCD
  3. Iterate from Page 0 to Page 127 performing steps 4-7
  4. Set AddressPage to Page x 32
  5. Send EEPROM instruction to activate chip enable
  6. Send EEPROM write instruction
  7. Iterate through the Address values from AddressPage to (Page +1)x32 to write 0xFF as the data value (AddressPage is Page * 32).
- If EEP_write_eepromFF() returns 1 indicating an error occurred then call Buz_ThreeBeep(), call DisplayString() to write 'EEPROM Write Error', call DisplayString() to write 'Press Valid Key to Continue', wait for Valid key to be pressed
- If EEP_write_eepromFF() returns 0 indicating write was successful, call BUZ_Beep() to sound security buzzer briefly, Call DisplayString to write 'EEPROM IC32 Write OK' to LCD, call TIM_ Temporize() to delay 1sec, Call DisplayString to write 'EEPROM IC32 Write OK' to LCD
- Call TIM_ Temporize() to delay 1sec
- Call DisplayErasePage() to clear LCD screen

3.1.6 NAND-FLASH Test [TSRQDOC, TRQ 3.5.6]

The test calls TEST_flash_Monitoring() to perform the following:

- Set flag IdentificationError to false
- Call DisplayString() to write 'Test Monitoring Memory' to LCD
- Call MONITFLASH_ReadID() to read the monitoring flash ID consisting of 4 bytes starting at address 0x500000 into Input_Buffer unsigned byte array
- Check the Manufacturer ID at Input_Buffer[0] location to match one of the three recognized IDs (0x20, 0xEC, 0x2C). If a match is found, call DisplayString() to write the manufacturer otherwise set IdentificationError to true and call DisplayString() to write 'Unknown Manufacturer'
- Check the Memory Characteristics ID at Input_Buffer[1] location to match one of four recognized IDs (0xF1, 0xDA, 0xDC, 0xD3). If a match is found, call DisplayString() to write the memory characteristics otherwise set IdentificationError to true and call DisplayString() to write 'Unknown Characteristics'
- If IdentificationError is true indicating an error occurred then call Buz_ThreeBeep(), call DisplayString() to write 'Monitoring Flash Error', call DisplayString() to write 'Press Valid Key to Continue', wait for Valid key to be pressed, call TIM_ Temporize() to delay 0.2sec, wait if Valid key is pressed, call TIM_ Temporize() to delay 0.2sec
- If IdentificationError is false indicating test was successful, call BUZ_Beep() to sound security buzzer briefly, Call DisplayString to write 'Monitoring Flash OK' to LCD, call TIM_ Temporize() to delay 2sec
- Call DisplayErasePage() to clear LCD screen

3.1.7 Event FLASH Test [TSRQDOC, TRQ 3.5.7]

The test calls TEST_flash_Evenements() to perform the following:

- Call EVFLASH_init_flash() which has no active code
- Call DisplayString to write 'Memory Test' to LCD
- Call DisplayString to write 'Event Saving' to LCD
- Set erreur_flash flag to 0
- Call EVFLASH_identifier_flash() to read the flash ID at 0x100000 and return the flash ID in variable type_memoire
- If EVFLASH_identifier_flash() returns:
  1. 0x2223 then call DisplayString to write 'AM29F400BT AMD' to LCD
  2. 0x22AB then call DisplayString to write 'AM29F400BB AMD' to LCD
  3. 0x22D6 then call DisplayString to write 'AM29F400BB ST' to LCD
  4. 0x22D5 then call DisplayString to write 'Memory Error AM29F400BT ST' to LCD, call DisplayString() to write 'Press Valid key to continue' and check for keypress

  5. If none of these are returned, set erreur_flash to 1, call DisplayString() to write 'Memory Identification Error, call Buz_Beep() to briefly sound security buzzer, call DisplayString() to write 'Press Valid key to continue' and check for Valid keypress
- Call TIM_ Temporize() to delay 1sec
- Call EVFLASH_effacer_flash() to erase flash device
- Call Buz_Beep() to briefly sound security buzzer
- Call DisplayString() to write 'Erase Flash End', then Call TIM_ Temporize() to delay 1sec, then Call DisplayString() to remove 'Erase Flash End'
- Call DisplayString() to write 'Verify Flash', then Call TIM_ Temporize() to delay 1sec, then Call DisplayString() to remove 'Verify Flash'
- Call EVFLASH_verifier_effacement() to verify flash erase by checking all the flash addresses for 0xFFFF. If EVFLASH_verifier_effacement() returns 0, an error occurred, if EVFLASH_verifier_effacement() returns 1, the test passed
- If EVFLASH_verifier_effacement() returns 0, then set erreur_flash = 10, write 'Non blank error to the LCD, beep the security buzzer, write 'Press Valid key to continue'
- Call TIM_ Temporize() to delay 1sec
- Call DisplayString() to write 'Flash Programming'
- For adr_flash starting at ADRESSE_DEBUT_FLASH_EVENEMENTS (0x100000) to ADRESSE_FIN_FLASH_EVENEMENTS (0x17FFFF) call EVFLASH_programmer_flash() to write the (unsigned int) flash address as data.

- If EVFLASH_programmer_flash() does not return 0, then call DisplayString() to write error information to the LCD, call BUZ_ThreeBeep() to sound security buzzer, and ask to press Valid key to continue
- Call DisplayString() to remove 'Flash Programming', then display 'Programming Finished' for 1 sec
- Call DisplayString() to write 'Programming Verification', and display for 1 sec
- Call EVFLASH_verifier_programmation() to verify flash addresses written at ADRESSE_DEBUT_FLASH_EVENEMENTS (0x100000) to ADRESSE_FIN_FLASH_EVENEMENTS (0x17FFFF) that the data equals write the (unsigned int) flash address as data.
- If EVFLASH_verifier_programmation() returns 1 indicating verification was successful, then call DisplayString() to write 'Programming Correct', if EVFLASH_verifier_programmation() returns 0, indicating an error then set erreur_flash = 11, and call DisplayString() to write 'Programming Incorrect', call BUZ_ThreeBeep() to sound security buzzer, and ask to press Valid key to continue
- Call Buz_Beep() and Call TIM_ Temporize() to delay 1sec
- Call DisplayString() to remove messages
- Call DisplayErasePage() to clear LCD screen

3.1.8 Buzzers Test [TSRQDOC, TRQ 3.5.8]

The test calls TEST_buzzers () to perform the following:

- Call BUZ_init_() setup the buzzer characteristics:
  1. Setup PWM Control Register 1
  2. Setup PWM Control Register 0
  3. Set PP3 PWM 3 period register to the sound level setting
  4. Setup CAPCOM timer 7 and 8 control register to use Timer T8 generate buzzer freq and T7
  5. Setup CAPCOM mode control register 7
  6. Set CAPCOM timer 7 reload register to set the PWM adjustment based on the buzzer freq
  7. Set the SoundPowerMin and SoundPowerMax values
  8. Set CAPCOM register 28 interrupt control register to set PWM cycle at 50%
  9. Set CAPCOM timer 7 reload register to start Timer 7
  10. Deactivate PWM3
- Call BUZ_PwmBuzzer() to deactivate PWM3
- Call InitialiserConvertisseur to initialize the ST Analog to Digital converter registers
- Call DisplayString() to write 'Buzzer Card Test' and 'Automatic Stop Function'
- Set ARRET_INV to 0 to turn OFF buzzer on buzzer pcb
- Call TIM_ Temporize() to delay .02 sec
- Set ARRET_INV to 1 to turn ON buzzer on buzzer pcb
- Call TIM_ Temporize() to delay .5 sec
- Call DisplayString() to write 'A/D value with buzzer on' and call DisplayNumber() to display buzzer A/D value AD_Digit.TestBuzzer
- If the buzzer A/D value AD_Digit.TestBuzzer is < 300, then the test fails, call BUZ_ThreeBeep() to sound the security buzzer, call DisplayNumber() to write the Automatic Buzzer Stop command failed and ask for Valid keypress
- Set ARRET_INV to 0 to turn OFF buzzer on buzzer pcb
- Call TIM_ Temporize() to delay .5 sec
- Call DisplayString() to write 'A/D value with buzzer off' and call DisplayNumber() to display buzzer A/D value AD_Digit.TestBuzzer
- If the buzzer A/D value AD_Digit.TestBuzzer is >= 300, then the test fails, call BUZ_ThreeBeep() to sound the security buzzer, call DisplayNumber() to write the Automatic Buzzer Stop command failed and ask for Valid keypress
- Call TIM_ Temporize() to delay 1 sec
- Call DisplayErasePage() to clear LCD screen
- Call DisplayString() to write 'Buzzer Card Test' and 'Command by PWM–Buz'
- Call BUZ_PwmBuzzer() to activate PWM3 to turn on buzzer
- Call TIM_ Temporize() to delay 0.5 sec
- Call DisplayString() to write 'A/D value with buzzer on' and call DisplayNumber() to display buzzer A/D value AD_Digit.TestBuzzer

- If the buzzer A/D value AD_Digit.TestBuzzer is < 300, then the test fails, call BUZ_ThreeBeep() to sound the security buzzer, call DisplayNumber() to write the PWM Buzzer command failed and ask for Valid keypress
- Call BUZ_PwmBuzzer() to de-activate PWM3 to turn OFF buzzer
- Call TIM_ Temporize() to delay 0.5sec
- Call DisplayString() to write 'A/D value with buzzer off' and call DisplayNumber() to display buzzer A/D value AD_Digit.TestBuzzer
- If the buzzer A/D value AD_Digit.TestBuzzer is > 300, then the test fails, call BUZ_ThreeBeep() to sound the security buzzer, call DisplayNumber() to write the PWM Buzzer command failed and ask for Valid keypress
- Call TIM_ Temporize() to delay 1sec
- Call DisplayErasePage() to clear LCD screen
- Call DisplayString() to write 'Security Buzzer Test'
- Set BUZ_SEC = 1 to turn ON security buzzer
- Call DisplayString() to write 'Verify that Security Buzzer is active'
- Call DisplayString() to write 'Press Valid Key to continue
- Call KEY_WaitToucheValid() to wait for Valid key press
- Set BUZ_SEC = 0 to turn OFF security buzzer
- Call DisplayString() to write 'Verify that Security Buzzer is not active'
- Call DisplayString() to write 'Press Valid Key to continue
- Call KEY_WaitToucheValid() to wait for Valid key press
- Call DisplayErasePage() to clear LCD screen

3.1.9 O2 Valve Test [TSRQDOC, TRQ 3.5.9]

The test calls TEST_VanneO2 () to perform the following:

- Call DisplayString() to display 'O2 Valve Test Command'
- Set CD_O2 signal to 1 to set the O2 signal VAL-O2 LOW
- Call BUZ_Beep() to briefly sound the security buzzer
- Call DisplayString() to display 'Verify that the O2 Valve Led is ON'
- Call DisplayString() to display 'Press Valid Key to continue'
- Call KEY_WaitToucheValid() to wait for the Valid keypress
- Call DisplayString() to remove message 'Verify that the O2 Valve Led is ON'
- Set CD_O2 signal to 0 to set the O2 signal VAL-O2 HIGH
- Call BUZ_Beep() to briefly sound the security buzzer
- Call DisplayString() to display 'Verify that the O2 Valve Led is OFF'
- Call DisplayString() to display 'Press Valid Key to continue'
- Call DisplayErasePage() to clear LCD screen

3.1.10 Exhalation Valve Control Test [TSRQDOC, TRQ 3.5.10]

The test calls TEST_Valve() to perform the following:

- Call DisplayString() to display 'Exhalation Valve Test Command'
- Set CD_Valve signal to 1 to turn V2 Valve ON
- Call TIM_ Temporize() to delay 1sec
- Call DisplayString() to display 'MES_I_VALVE A/D value with valve ON'
- Call DisplayNumber() to display the AD_Digit.MesIValve value from the ST10 A/D converter MES_I_VALVE signal
- If AD_Digit.MesIValve is less than < 500 A/D value, then call BUZ_ThreeBeep() to sound security buzzer, Call DisplayString() to display 'Valve Command Error', call KEY_WaitToucheValid() to wait for the Valid keypress
- Set CD_Valve signal to 0 to turn V2 Valve OFF
- Call TIM_ Temporize() to delay 1sec
- Call DisplayString() to display 'MES_I_VALVE A/D value with valve OFF'
- Call DisplayNumber() to display the AD_Digit.MesIValve value from the ST10 A/D converter MES_I_VALVE signal
- If AD_Digit.MesIValve is less than >= 500 A/D value, then call BUZ_ThreeBeep() to sound security buzzer, Call DisplayString() to display 'Valve Command Error', call KEY_WaitToucheValid() to wait for the Valid keypress
- Call TIM_ Temporize() to delay 2 sec
- Call DisplayErasePage() to clear LCD screen

3.1.11 Flow Sensors Test [TSRQDOC, TRQ 3.5.11]

The test calls TEST_CapteursDebit() to perform the following:

- Set DigitDebitInspMin to 1023
- Set DigitDebitExpMin to 1023
- Set DigitDebitO2Min to 1023
- Set PhaseAffichage to 0
- Set DigitDebitInspMax, DigitDebitExpMax, DigitDebitO2Max to 0
- Call DisplayString() to display 'Test Flow Sensors'
- Call DisplayString() to display 'Inspiratory Flow IC15'
- If DeviceType is not 'Legendair_S2', call DisplayString() to display 'Expiratory Flow IC17'
- If DeviceType is 'Supportair_M2', call DisplayString() to display 'O2 Flow'
- Call DisplayString() to display 'Press VALID key to continue'
- While TOUCH_VALID is false (VALID key not pressed), perform the following:
  1. If AD_Digit.FlowInsp < DigitDebitInspMin set AD_Digit.FlowInsp to Min value
  2. If AD_Digit.FlowInsp > DigitDebitInspMax set AD_Digit.FlowInsp to Max value
  3. If AD_Digit.FlowExhal < DigitDebitExpMin set AD_Digit.FlowExhal to Min value
  4. If AD_Digit.FlowExhal > DigitDebitExpMax set AD_Digit.FlowExhal to Max value
  5. If AD_Digit.FlowO2 < DigitDebitO2Min set FlowO2 to Min value
  6. If AD_Digit.FlowO2 > DigitDebitO2Min set FlowO2 to Max value

  7. If PhaseAffichage is 0 (Inspiratory Flow), then call DisplayNumber() to display AD_Digit.FlowInsp, DigitDebitInspMin and DigitDebitInspMax values, and check that AD_Digit.FlowInsp is < 225 and AD_Digit.FlowInsp is > 184,
  7a. If AD_Digit.FlowInsp is out of limits, call DisplayString() to display 'Sensor Error', call BUZ_ThreeBeep(), and call DisplayString() to remove 'Sensor Error' message
  7b. Set PhaseAffichage to 1

  8. If PhaseAffichage is 1 (Exhalation Flow) and DeviceType is not SUPPORTAIR_M2, then call DisplayNumber() to display AD_Digit.FlowExhal, DigitDebitExhMin and DigitDebitExhMax values, and check that AD_Digit.FlowExhal is < 225 and AD_Digit.Flow Exhal is > 184,
  8a. If AD_Digit.FlowExh is out of limits, call DisplayString() to display 'Sensor Error', call BUZ_ThreeBeep(), and call DisplayString() to remove 'Sensor Error' message
  8b. Set PhaseAffichage to 2

  9. If PhaseAffichage is 2 (O2 Flow) and DeviceType is SUPPORTAIR_M2, then call DisplayNumber() to display AD_Digit.FlowO2, DigitDebitO2Min and DigitDebitO2Max values, and check that AD_Digit.FlowO2 is < 225 and AD_Digit.Flow O2 is > 184,
  9a. If AD_Digit.Flow O2 is out of limits, call DisplayString() to display 'Sensor Error', call BUZ_ThreeBeep(), and call DisplayString() to remove 'Sensor Error' message
  9b. Set PhaseAffichage to 0

- Call TIM_ Temporize() to delay .1 sec
- Call KEY_WaitToucheValid to wait for Valid key
- Call DisplayErasePage() to clear LCD screen

3.1.12 Pressure Sensors Test [TSRQDOC, TRQ 3.5.12]

The test calls TEST_CapteursDebit() to perform the following:

- Set DigitPressionIntMin to 1023
- Set DigitPressionValveMin to 1023
- Set DigitPressionProxMin to 1023
- Set DigitPressionO2Min to 1023
- Set DigitPressionAbsMin to 1023
- Set PhaseAffichage to 0
- Call DisplayString() to display 'Pressure Sensors Test'
- Call DisplayString() to display 'Internal Pressure IC16'

- Call DisplayString() to display 'Valve Pressure IC18'
- Call DisplayString() to display 'Proximal Pressure IC22'
- Call DisplayString() to display 'Check that absence of sensor error'
- If DeviceType is SUPPORTAIR_M2, call DisplayString() to display 'O2 Pressure IC19'
- Call DisplayString() to display 'Atmospheric Pressure IC33'
- Call DisplayString() to display 'Press Valid Key to continue'
- Call TIM_ Temporize() to delay 1 sec
- Call DisplayString() to display 'mbar' for the units field in three locations
- While TOUCH_VALID is False waiting for Valid keypress, perform the following:

1. If AD_Digit.PatientPressure < DigitPressionIntMin, set PatientPressure to Min value
2. If AD_Digit.PatientPressure > DigitPressionIntMax, set PatientPressure to Max value
3. Set PressionInt = PatientPressure – 175
4. Set PressionInt = PatientPressure * 1344
5. Set PressionInt = PatientPressure/1000
6. If AD_Digit.ValvePressure < DigitPressionValveMin set AD_Digit.ValvePressure to Min value
7. If AD_Digit.ValvePressure > DigitPressionValveMax set AD_Digit.ValvePressure to Max value
8. Set PressionValve = ValvePressure – 175
9. Set PressionValve = PressionValve * 1344
10. Set PressionValve = PressionValve /1000
11. If AD_Digit.ProxPressure < DigitPressionProxMin set AD_Digit.ProxPressure to Min value
12. If AD_Digit.ProxPressure > DigitPressionProxMax set AD_Digit.ProxPressure to Max value
13. Set PressionProx = ProxPressure – 175
14. Set PressionProx = PressionProx * 1344
15. Set PressionProx = PressionProx /1000
16. If AD_Digit.O2Pressure < DigitPressionO2Min set AD_Digit.O2Pressure to Min value
17. If AD_Digit.O2Pressure > DigitPressionO2Max set AD_Digit.O2Pressure to Max value
18. Set PressionO2 = O2Pressure - 102
19. Set PressionO2 = PressionO2 * 8424
20. Set PressionO2 = PressionO2 /1000
21. If AD_Digit.AbsPressure < DigitPressionAbsMin set AD_Digit.AbsPressure to Min value
22. If AD_Digit.AbsPressure > DigitPressionAbsMax set AD_Digit.AbsPressure to Max value
23. Set PressionAbs = AbsPressure - 102
24. Set PressionAbs = PressionAbs * 610
25. Set PressionAbs = 600 + (PressionAbs/1000)
26. If PhaseAffichage = 0 (Internal/Patient Pressure), perform the following:
    a. Call DisplayNumber() to display AD_Digit.PatientPressure, DigitPressionIntMax, DigitPressionIntMin, PressionInt values

    b. Check AD_Digit.PatientPressure to be > 152 and AD_Digit.PatientPressure < 201, else call DisplayString() to display 'Sensor Error' and call BUZ_ThreeBeep()
    c. Set PhaseAffichage to 1

27. If PhaseAffichage = 1 (Valve Pressure), perform the following:
    a. Call DisplayNumber() to display AD_Digit. ValvePressure , DigitPressionValveMax, DigitPressionValveMin, PressionValve values
    b. Check AD_Digit.ValvePressure to be > 152 and AD_Digit.ValvePressure < 201, else call DisplayString() to display 'Sensor Error' and call BUZ_ThreeBeep()
    c. Set PhaseAffichage to 2

28. If PhaseAffichage = 2 (Proximal Pressure), perform the following:
    a. Call DisplayNumber() to display AD_Digit.ProxPressure, DigitPressionProxMax, DigitPressionProxMin, PressionProx values

b. Check AD_Digit.ProxPressure to be > 152 and AD_Digit.ProxPressure < 201, else call DisplayString() to display 'Sensor Error' and call BUZ_ThreeBeep()
c. Set PhaseAffichage to 3

29. If PhaseAffichage = 3 (O2 Pressure) perform the following:
If DeviceType = SUPPORTAIR_M2 perform steps a-b:
a. Call DisplayNumber() to display AD_Digit.O2Pressure, DigitPressionO2Max DigitPressionO2Min, PressionO2 values
b. Check AD_Digit.O2Pressure to be > 86 and AD_Digit.O2Pressure < 119, else call DisplayString() to display 'Sensor Error' and call BUZ_ThreeBeep()
c. Set PhaseAffichage to 4

30. If PhaseAffichage = 4 (Barometric Pressure) perform the following:
a. Call DisplayNumber() to display AD_Digit.AbsPressure, DigitPressionAbsMax DigitPressionAbsMin, PressionAbs values
b. Check AD_Digit.AbsPressure to be > 640 and AD AD_Digit.AbsPressure < 820, else call DisplayString() to display 'Sensor Error' and call BUZ_ThreeBeep()
c. Set PhaseAffichage to 0

- Call TIM_ Temporize() to delay .1 sec
- Call KEY_WaitToucheValid to wait for Valid key
- Call DisplayErasePage() to clear LCD screen

3.1.13 FIO2 Test [TSRQDOC, TRQ 3.5.13]

The test calls TEST_Fio2() to perform the following:

- Set DigitFio2Min to 1023
- Set DigitFio2Max to 0
- Set OffsetFIO2 to 199
- Call DisplayString() to display 'FI02 Test' and 'Measure FI02'
- Call DisplayString() to display 'Check that absence of sensor error'
- Call DisplayString() to display 'Press on Valid key to continue'
- Call TIM_ Temporize() to delay 1 sec
- Call DisplayString() to display 'FI02 CAN'
- While TOUCH_VALID is False waiting for Valid keypress, perform the following:
  1. If AD_Digit.Fio2 < DigitFio2Min set AD_Digit.Fio2 to Min value
  2. If AD_Digit.Fio2 > DigitFio2Max set AD_Digit.Fio2 to Max value
  4. Set Fio2 = ((AD_Digit.Fio2 - OffsetFIO2 * 1226519)/1000000) + 209
  5. Call DisplayNumber() to display AD_Digit.Fio2, DigitFio2Max, DigitFio2Min, Fio2
  6. Check AD_Digit.Fio2 to be > 188 and AD_Digit.Fio2< 209, else call DisplayString() to display 'Sensor Error' and call BUZ_ThreeBeep()
- Call TIM_ Temporize() to delay .5 sec
- Call KEY_WaitToucheValid to wait for Valid key
- Call DisplayErasePage() to clear LCD screen

3.1.14 Power Management PCB Interface Test [TSRQDOC, TRQ 3.5.14]

The test checks the interface to the Power Management Board by sending a basic data frame over the two SPI communication channels which also utilize the SPI functions in the Power Management Board (see following section). In the CPU code the test calls TEST_InterfaceAlimCPU() to perform the following:

- Set Error to False
- Set USB_Reply to 0
- Set MessageDisplay to ON

- Set USB_ReplyMemo to 0xAA
- Set Nb_Test to 0
- Call USB_Init() to setup in Master mode, 1Mhz speed, 8 Bits transfer, MSB first, Ignore error
- Call DisplayString() to display 'Test Interface' and 'Power Management Card'
- Call DisplayString() to display 'Test SP1 USB Key'
- Call DisplayString() to display 'Value returned by USB bridge'
- Call USB_Key_Connected() to check if a key USB is connected on port 1. Set USB_Reply to the value returned by this function. If the function:
- Call TIM_ Temporize() to delay .5 sec
- Set Nb_Test to 0
- While Nb_Test < 500 and Error = False, perform the following:

1. Call DisplayNumber() to display Nb_Test (test number)
2. Call USB_Key_Connected() to check if a key USB is connected on port 1. Set USB_Reply to the value returned by this function.
3. If USB_Reply does not equal USB_Reply_Memo then do the following:
    a. Set USB_Reply_Memo = USB_Reply
    b. Call DisplayNumber() to display USB_Reply
    c. If USB_Reply = 12, then display 'Key not connected'
    d. If USB_Reply = 6, then display 'USB Bridge busy'
    e. If USB_Reply = 3, then display 'Error in data sent by USB bridge'
    f. If USB_Reply = 4, then display 'Error in data sent by USB bridge'
    g. If USB_Reply = 5, then display 'Error in data sent by USB bridge'
    h. If USB_Reply = 0, then display 'Key connected'
    i. Else, display no message data

3. If USB_Reply does not equal 12 then do the following:
    a. Set Error to True
    b. Call BUZ_ThreeBeep()
    c. Call DisplayString() to display 'USB Interface Error'
    d. Call DisplayString() to display 'Press Valid key to continue'
    e. Call KEY_WaitToucheValid() to wait for Valid key

4. Increment Nb_Test
6. Call TIM_ Temporize() to delay .005 sec

- Call TIM_ Temporize() to delay 1 sec
- Call BUZ_Beep()
- Call DisplayErasePage() to clear LCD screen
- Call DisplayString() to display 'Test Interface' and 'Power Management Card'
- Call DisplayString() to display 'Test SP0'
- Call DisplayString() to display 'Value sent to the Power Management Card''
- Call DisplayString() to display 'Value received from the Power Management Card''
- Setup the registers for the SPI communication to configure the SPI clock for 1.8 MHz, full duplex master mode and activate the SPI channel
- Set Nb_Test to 0
- Set Error to False
- While Nb_Test < 500 and Error = False, perform the following:

1. Set the CE_SPI_PIC enable signal low (active low), and set the CE_SPIPIC_IC44 high to allow the SPI signal to be recv'd
2. Call TIM_ Temporize() to delay 0.0001 sec
3. Set SSCTB = NB_Test to send SPI data
4. Wait for the SPI transfer to complete
5. Set ReceivedData = SSCRB register to get SPI received data
6. Call DisplayNumber() to display Nb_Test and ReceivedData values
7. If Nb_Test does not equal ReceivedData + 1, and Nb_Test >0 then:
    a. a. Set Error to True
    b. Call BUZ_ThreeBeep()
    c. Call DisplayString() to display 'SPI 0 Interface Error'
    d. Call DisplayString() to display 'Press Valid key to continue'
    e. Call KEY_WaitToucheValid() to wait for Valid key

8. Set the CE_SPI_PIC enable signal high, and set the CE_SPIPIC_IC44 low
9. Call TIM_ Temporize() to delay 0.005 sec
10 . Set Nb_Test = (Nb_Test +1) modulus 256

- Call TIM_ Temporize() to delay 1.5 sec
- Call DisplayErasePage() to clear LCD screen

Power Management Board Functions

The two SPI channels are checked in this test, SPI 0 and SPI 1. SPI 1 uses the firmware residing on the EEPROM for the Cypress USB Host Controller and is referenced in the product software documentation. SPI 0 is checked by receiving SPI data which causes an interrupt and the service routine DRV_SPI_ST10_Interrupt() is called to perform the following:

1. Set SpiBufferOfReceipt[SpiPtrWriteBufferRx] to the PIC SPI Receive Buffer/Transmit register SSP1BUF to receive the data byte
2. Set SSP1BUF to SpiBufferOfReceipt[SpiPtrWriteBufferRx] to transmit back
3. Increment the SpiPtrWriteBufferRx index
4. Check the SpiPtrWriteBufferRx to limit to the maximum buffer size allowed
5. Increment SpiArrayTransmitDataId variable
6. If SpiArrayTransmitDataId is equal to SPI_ST10_EndOfTableTransmit constant then set SpiArrayTransmitDataId to SPI_ST10_MSB_START_FRAME and call DRV_SPI_ST10_Transmit() to set the SpiArrayTransmitData array to the values for the data frame

3.1.15 Alarm Repeater Test [TSRQDOC, TRQ 3.5.15]

The test calls TEST_RappelAlarme () to perform the following:

- Set DataTransmmitted variable to 0
- Call UART_Initialisation() to configure ST10 Port 8.7 as UART Interrupt input, and call UART_SetSerialParameters() to set the UART baud rate to 36800, 8 data bits, no parity, and 1 stop bit
- Call DisplayString() to display ‘Test Alarm Repeater’, ‘Press Valid key to continue’, ‘Test RAP_ALARM state’, ‘
- Call BUZ_Beep() to sound security buzzer briefly
- While the Valid key is not pressed, perform the following:
  1. Set RAP_AL to 1
  2. Call UART_Transmit() to send DataTransmmitted (0)
  3. Call TIM_ Temporize() to delay .5 sec
  4. Call DisplayNumber() to display receive.value1, receive.value2, receive.value3, receive.value4, receive.value5 received in UART
  5. Call UART_ReadDate() to read Data.value1, Data.value2, Data.value3, Data.value4, Data.value5
  6. Call DisplayNumber() to display receive.value1, receive.value2, receive.value3, receive.value4, receive.value5 received in UART
  7. If receive.value1 does not equal DataTransmited, or If receive.value2 does not equal DataTransmited + 1, or If receive.value3 does not equal DataTransmited + 2, or If receive.value4 does not equal DataTransmited + 3, or If receive.value5 does not equal DataTransmited + 4, then call BUZ_ThreeBeep(), call DisplayString() to display ‘Error’, call TIM_ Temporize() to delay 1 sec, call DisplayString() to remove ‘Error’ message
  8. Increment DataTransmitted variable
  9. Set DataTransmitted variable to modulus 200
  10. Test TEST_RAP_ALARM and if equal to 1, then call DisplayString() to display’1’, otherwise then call BUZ_ThreeBeep(), call DisplayString() to display ‘Error’, call TIM_ Temporize() to delay 1 sec, call DisplayString() to remove ‘Error’ message
  11. Set RAP_AL to 0
  12. Call TIM_ Temporize() to delay .5 sec
  13. Call DisplayNumber() to display receive.value1, receive.value2, receive.value3, receive.value4, receive.value5 received in UART
  14. Call UART_ReadDate() to read Data.value1, Data.value2, Data.value3, Data.value4, Data.value5

15. Call DisplayNumber() to display receive.value1, receive.value2, receive.value3, receive.value4, receive.value5 received in UART
16. If receive.value1 does not equal DataTransmited, or If receive.value2 does not equal DataTransmited + 1, or If receive.value3 does not equal DataTransmited + 2, or If receive.value4 does not equal DataTransmited + 3, or If receive.value5 does not equal DataTransmited + 4, then call BUZ_ThreeBeep(), call DisplayString() to display 'Error', call TIM_ Temporize() to delay 1 sec, call DisplayString() to remove 'Error' message
17. Increment DataTransmitted variable
18. Set DataTransmitted variable to modulus 200
19. Test TEST_RAP_ALARM and if equal to 0, then call DisplayString() to display'0', otherwise then call BUZ_ThreeBeep(), call DisplayString() to display 'Error', call TIM_ Temporize() to delay 1 sec, call DisplayString() to remove 'Error' message

- KEY_WaitToucheValid() to wait while Valid key is pressed
- Call DisplayErasePage() to clear LCD screen
- Set RAP_AL to 1

3.1.16 Turbine Test [TSRQDOC, TRQ 3.5.16]

The test calls TEST_InterfaceTurbine () to perform the following:

- A two-dimensional array TempConvTable is setup with the Temperature versus Resistance linearization curve data for the thermistor
- Set IndexSegment to 0
- Set ENABLE_TURB signal to 1 to enable the Turbine
- Call TIM_ Temporize() to delay .5 sec
- Set FREIN signal to 1 (BRAKE/ Signal) to turn OFF brake
- Call TIM_ Temporize() to delay .1 sec
- Set FREIN signal to 0 (BRAKE/ Signal) to turn ON brake
- Set the PWM 0 Period Register PPO to 4096 for the PWM period: F=1/(4096(12bits)*50ns(1/Fcpu)) = 4,882KHz Note: PWM 0 P7.0 is used to drive the Turbine Speed-Setpoint
- Set PWM Pulse Width Register PWO to 4095
- Set FREIN to 1 (BRAKE/ Signal) to turn OFF brake
- Setup timer T0 and T1 CAPCOM Timer 0/1 control register T01CON for the Timer Mode
- Set reload registerT1REL to 0
- Setup Capture compare mode register CCM0 trigger on negative edge
- Setup interrupt capture register CCOIC
- Init Peripheral Event Controller PECC1
- Setup P2.0 as input for motor speed measurement
- Setup source pointer to measurement source CAPCOM0 Register CC0
- Setup destination pointer to the conversion table in RAM
- Start Timer 1
- Enable Capture Interrupt
- Call DisplayString() to display 'Test Turbine Interface', 'Set Speed PW0:', 'Turbine Speed', 'rev/min', 'Turbine Temperature', 'ºC'
- Call BUZ_Beep()
- Set PW0 to 2500
- Call DisplayString() to display 'Test Sets and Measures Speed and Temperature'
- Set Nb_Test to 0
- While Nb_Test is less than 8, do the following:
  1. Increment Nb_Test
  2. If AvailableSpeedBlowerDigit = True, then MeasuredSpeedBlower = 600000000 **/** (SpeedBlowerDigit ***** 4) else then MeasuredSpeedBlower = 0
  3. While AD_Digit**.**TurbineTemp **<** TempConvTable[IndexSegment][0], Increment IndexSegment until the temperature is located on the theoretical thermistor temp. curve
  4. Set Compute2 = 100000 / (TempConvTable[IndexSegment-1][0] - TempConvTable[IndexSegment][0]
  5. Set Compute1 = Compute2 * (TempConvTable[IndexSegment-1][0] - AD_Digit.TurbineTemp) / 100 + TempConvTable[IndexSegment-1][1]
  6. Set Temperature = Compute1 / 100

7. Call DisplayNumber() to display PW0, MeasuredSpeedBlower, and Temperature
8. Call BLOWER_LaunchingSpeedBlowerMeasure() to set AvailableSpeedBlowerDigit = False, re-init pointer to PECC1 (Peripherhal Event Controller), re-init pointer to CAPCOM Register 0, enable capture interrupt
9. Call TIM_ Temporize() to delay .5 sec

- If MeasuredSpeedBlower < 15000 (rev/min) then call BUZ_ThreeBeep(), call DisplayString() to display 'Turbine Speed Error' and 'Press Valid Key to Continue', call KEY_WaitToucheValid() to wait for Valid key, Call DisplayString() to remove 'Turbine Speed Error' message, call DisplayString() to remove 'Turbine Speed Error', 'Press Valid Key to Continue' message

- If Temperature **>** 30 or Temperature **<** 20, then perform the following:
  1. Call BUZ_ThreeBeep()
  2. Call DisplayString() to display 'Turbine Temperature measure Error'
  3. Call DisplayString() to display 'Press Valid Key to Continue'
  4. Do the following until the Valid key is pressed:
     a. While AD_Digit**.**TurbineTemp **<** TempConvTable[IndexSegment][0], Increment IndexSegment until the temperature is located on the theoretical thermistor temp. curve
     b. Set Compute2 = 100000 / (TempConvTable[IndexSegment-1][0] - TempConvTable[IndexSegment][0]
     c . Set Compute1 = Compute2 * (TempConvTable[IndexSegment-1][0] - AD_Digit.TurbineTemp) / 100 + TempConvTable[IndexSegment-1][1]
     c. Set Temperature = Compute1 / 100
     d. Call DisplayNumber() to display Temperature
     e. Call TIM_ Temporize() to delay .5 sec

  5. Wait while TOUCHE_VALID is not true
  6. Call KEY_WaitToucheValid() to wait for Valid key
  7. Call DisplayString() to display 'Turbine Temperature measure Error'
  8. Call DisplayString() to display 'Press Valid Key to Continue'

- Call DisplayString() to display 'Test Sets and Measures Speed and Temperature'
- Call DisplayString() to display 'Test ENABLE_TURBINE'
- Set ENABLE_TURB to 0
- Set Nb_Test to 0
- Set MeasuredSpeedBlower to 0
- Set SpeedBlowerDigit to 65535

- While Nb_Test is less than 8, do the following:
  1. Increment Nb_Test
  2. If AvailableSpeedBlowerDigit = True, then MeasuredSpeedBlower = 600000000 **/** (SpeedBlowerDigit ***** 4) else then MeasuredSpeedBlower = 0
  3. While AD_Digit**.**TurbineTemp **<** TempConvTable[IndexSegment][0], Increment IndexSegment until the temperature is located on the theoretical thermistor temp. curve
  4. Set Compute2 = 100000 / (TempConvTable[IndexSegment-1][0] - TempConvTable[IndexSegment][0]
  5. Set Compute1 = Compute2 * (TempConvTable[IndexSegment-1][0] - AD_Digit.TurbineTemp) / 100 + TempConvTable[IndexSegment-1][1]
  6. Set Temperature = Compute1 / 100
  7. Call DisplayNumber() to display PW0, MeasuredSpeedBlower, and Temperature
  8. Call BLOWER_LaunchingSpeedBlowerMeasure() to set AvailableSpeedBlowerDigit = False, re-init pointer to PECC1 (Peripherhal Event Controller), re-init pointer to CAPCOM Register 0, enable capture interrupt
  9. Call TIM_ Temporize() to delay .5 sec

- If MeasuredSpeedBlower > 15000 (rev/min) then call BUZ_ThreeBeep(), call DisplayString() to display 'Turbine Speed Error' and 'Press Valid Key to Continue', call KEY_WaitToucheValid() to wait for Valid key, Call DisplayString() to remove 'Turbine

Speed Error' message, call DisplayString() to remove 'Turbine Speed Error', 'Press Valid Key to Continue' message

- Call DisplayString() to remove 'Test ENABLE_TURBINE' message
- Set ENABLE_TURB to 1
- Set Nb_Test to 0
- Set MeasuredSpeedBlower to 0

- While Nb_Test is less than 8, do the following:
  1. Increment Nb_Test
  2. If AvailableSpeedBlowerDigit = True, then MeasuredSpeedBlower = 600000000 **/** (SpeedBlowerDigit ***** 4) else then MeasuredSpeedBlower = 0
  3. While AD_Digit**.**TurbineTemp **<** TempConvTable[IndexSegment][0], Increment IndexSegment
  4. Set Compute2 = 100000 / (TempConvTable[IndexSegment-1][0] - TempConvTable[IndexSegment][0]
  5. Set Compute1 = Compute2 * (TempConvTable[IndexSegment-1][0] - AD_Digit.TurbineTemp) / 100 + TempConvTable[IndexSegment-1][1]
  6. Set Temperature = Compute1 / 100
  7. Call DisplayNumber() to display PW0, MeasuredSpeedBlower, and Temperature
  8. Call BLOWER_LaunchingSpeedBlowerMeasure() to set AvailableSpeedBlowerDigit = False, re-init pointer to PECC1 (Peripherhal Event Controller), re-init pointer to CAPCOM Register 0, enable capture interrupt
  9. Call TIM_ Temporize() to delay .5 sec

  - Set PW0 to 4096
  - Set Nb_Test to 0

- While Nb_Test is less than 20, do the following:
  1. Increment Nb_Test
  2. FREIN = FREIN Ex-OR'd with 1 to toggle the brake on/off
  3. . If AvailableSpeedBlowerDigit = True, then MeasuredSpeedBlower = 600000000 **/** (SpeedBlowerDigit ***** 4) else then MeasuredSpeedBlower = 0
  4. While AD_Digit**.**TurbineTemp **<** TempConvTable[IndexSegment][0], Increment IndexSegment
  5. Set Compute2 = 100000 / (TempConvTable[IndexSegment-1][0] - TempConvTable[IndexSegment][0]
  6. Set Compute1 = Compute2 * (TempConvTable[IndexSegment-1][0] - AD_Digit.TurbineTemp) / 100 + TempConvTable[IndexSegment-1][1]
  7. Set Temperature = Compute1 / 100
  8. Call DisplayNumber() to display PW0, MeasuredSpeedBlower, and Temperature
  9. Call BLOWER_LaunchingSpeedBlowerMeasure() to set AvailableSpeedBlowerDigit = False, re-init pointer to PECC1 (Peripherhal Event Controller), re-init pointer to CAPCOM Register 0, enable capture interrupt
  10. Call TIM_ Temporize() to delay .5 sec

  - Set FREIN = 1 to turn brake OFF

  - If MeasuredSpeedBlower > 5000 (rev/min) then call BUZ_ThreeBeep(), call DisplayString() to display 'Turbine Speed Error' and 'Press Valid Key to Continue', call KEY_WaitToucheValid() to wait for Valid key, Call DisplayString() to remove 'Turbine Speed Error' message, call DisplayString() to remove 'Turbine Speed Error', 'Press Valid Key to Continue' message

  - Call DisplayErasePage() to clear LCD screen

3.1.17 Watchdog Timer Test [TSRQDOC, TRQ 3.5.17]

The test calls TEST_Watchdog () to perform the following:

- Call DisplayString() to display 'Watchdog Function Test'
- Call DisplayString() to display 'The Board is about to reset in a few moment
- Call TIM_ Temporize() to delay 2 sec
- Set the WatchdogAuthorization flag to false to prevent the watchdog input from changing state in the Timer 6, 1mS interrupt service routine
- Call TIM_ Temporize() to delay 2 sec to allow time for the watchdog timeout
- The unit would expect to have reset at this point. Upon reset the WatchdogAuthorization flag is set to true
- If there was no reset, call DisplayString() to display 'Watchdog Function Error'
- Call DisplayString() to display 'Press Valid key to continue'
- Call KEY_WaitToucheValid() to wait for the Valid key press
- Call DisplayErasePage() to clear LCD screen

3.1.18 DC Voltage Measurement Tests [TSRQDOC, TRQ 3.5.19]

The DC Voltage Measurement Tests are performed manually by the test operator using a DVM. The voltage measurements are made with the black (-) lead of the DVM connected to the ground provided on the test fixture and the red (+) lead of the DVM as follows:

- **+5VDC** regulator supply output: Connect (+) lead of DVM to C106 (+)
- **+3.3VDC** regulator supply output: Connect (+) lead of DVM to C92 (+)
- **+5VDC Reference**: Connect (+) lead of DVM to T11 Collector (tab)
- **+10VDC Reference**: Connect (+) lead of DVM to T12 Collector (tab)

3.1.19 DisplayErasePage()

This is a function frequently called by the test functions to erase the LCD screen.

Format:
void DisplayErasePage(UWORD16 value)

The function performs the following:

- Call DIS_FifoWrite(DRV_ERASE_PAGE, value, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
  DRV_ERASE_PAGE constant is the display function number to erase the page, value is equal to 1
  DIS_FifoWrite() is referenced in the product software.

3.1.20 DisplayString()

This is a function frequently called by the test functions to display a character string.
Format:

void DisplayString(UBYTE *ptr_char,UWORD16 Font,UWORD16 Line,UWORD16 Column,
UWORD16 PrintOn,UWORD16 Page)

The function performs the following:

- Call DIS_FifoWrite (DRV_STRING, Address of string to display >> 16, Address of string to display, Font, Line position, Column position, PrintOn, Page, 0, 0, 0, 0, 0)

  DRV_STRING constant is the display function number. The address of the string to be displayed is passed, as well as the line (row) and column position. The PrintOn flag is either to to ON for display or OFF to remove the text displayed. DIS_FifoWrite() is referenced in the product software.

3.1.21 DisplayNumber()

This is a function frequently called by the test functions to display a numerical value.
Format:
void DisplayNumber(SWORD32 Value,UWORD16 Nature,UWORD16 Line
,UWORD16 Column,UWORD16 Font,UWORD16 Nb_Digit,e_BOOL Zero,UWORD16 PrintOn
,UWORD16 Page)

The function performs the following:

- Call DIS_FifoWrite (DRV_NUMBER, unsigned word number to display >> 16, unsigned word number to display, nature, line position, column position, font, Number of digits, zero, PrintOn, Page, 0, 0)

  DRV_NUMBER constant is the display function number. The value of the number to be displayed is passed, as well as the line (row) and column position. The PrintOn flag is either to to ON for display or OFF to remove the text displayed. DIS_FifoWrite() is part of the product software.

3.1.22 BUZ_Beep()

This is a function frequently called by the test functions to sound the security buzzer on the CPU board itself.

Format:
void BUZ_Beep(void)

The function performs the following:

- Set BUZ_SEC = 1 (uC P7.5) to turn ON the security buzzer
- Call TIM_ Temporize() to delay 0.08 Sec
- Set BUZ_SEC = 0 (uC P7.5) to turn OFF the security buzzer

3.1.23 BUZ_ThreeBeep()

This is a function frequently called by the test functions to sound the security buzzer for three short beeps.

Format:
void BUZ_ThreeBeep(void)

The function performs the following:

- Set BUZ_SEC = 1 (uC P7.5) to turn ON the security buzzer
- Call TIM_ Temporize() to delay 0.1 Sec
- Set BUZ_SEC = 0 (uC P7.5) to turn OFF the security buzzer
- Call TIM_ Temporize() to delay 0.1 Sec
- Set BUZ_SEC = 1 (uC P7.5) to turn ON the security buzzer
- Call TIM_ Temporize() to delay 0.1 Sec
- Set BUZ_SEC = 0 (uC P7.5) to turn OFF the security buzzer
- Call TIM_ Temporize() to delay 0.1 Sec
- Set BUZ_SEC = 1 (uC P7.5) to turn ON the security buzzer
- Call TIM_ Temporize() to delay 0.1 Sec
- Set BUZ_SEC = 0 (uC P7.5) to turn OFF the security buzzer

3.1.24 TIM_Temporize()

This is a function frequently called by the test functions to provide a delay. The function is called with the number of uSeconds to delay as the parameter passed. The TIM_Temporize is referenced in the product software

Format:
void TIM_Temporize(UWORD32 useconde)

3.1.25 KEY_WaitToucheValid()

This is a function frequently called by the test functions to wait for a recognized key press of the Valid key button. The TIM_Temporize is referenced in the product software

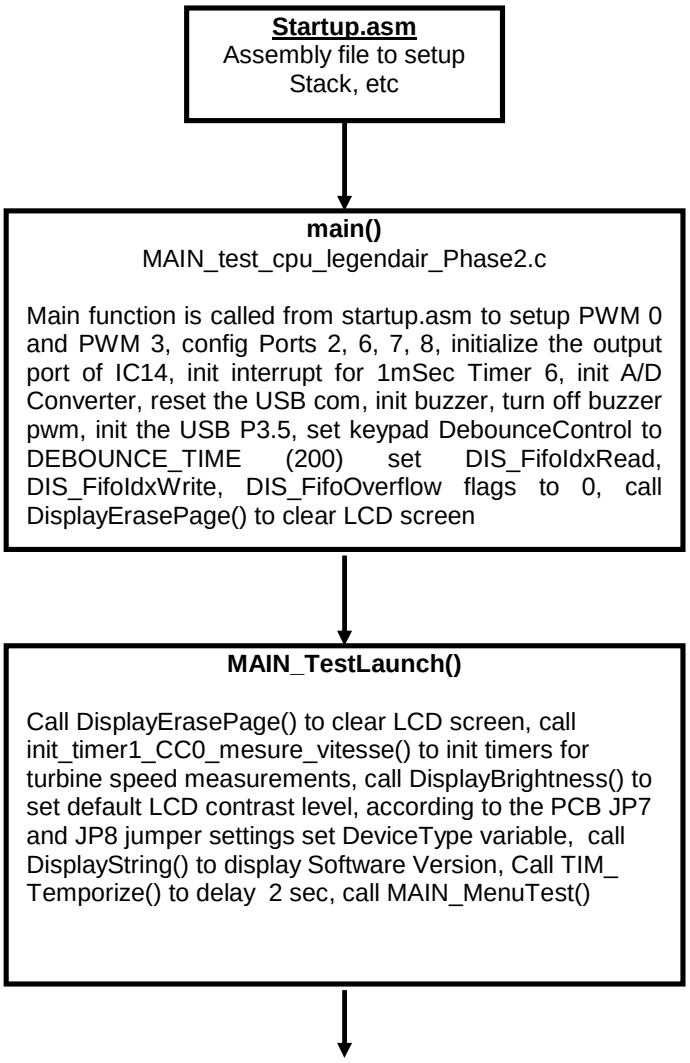
Format:
void KEY_WaitToucheValid(void)

The function performs the following:

- While the TOUCHE_VALID variable indicating the Valid key press is not true or pressed, loop to waiting. Note: TOUCHE_VALID is set in the KEY_ReadKeyboard()
- Call TIM_ Temporize() to delay 0.2 Sec
- While the TOUCHE_VALID variable indicating the Valid key press is not true or pressed, loop to waiting. Note: TOUCHE_VALID is set in the KEY_ReadKeyboard()
- Call TIM_ Temporize() to delay 0.2 Sec

3.1.26 Test Startup [TSRQDOC, GRQ[3.4.1]]

On UUT power-up the following modules are executed in this order:



MAIN_MenuTest()

Set Choix and TypeChoix to 0, Call DisplayString () to display 'Complete Test' and 'Yes' and 'No', enter a forever loop to do the following:

- If the up arrow key is pressed, if TypeChoix > 0 then decrement TypeChoix else TypeChoix = 3, if TypeChoix = 0 or 1 then set Language to Francais else set language to Anglais , call TIM_ Temporize() to delay .2 sec, loop while up arrow key is pressed, call TIM_ Temporize() to delay .2 sec

- If the down arrow key is pressed, then increment TypeChoix if TypeChoix < 3 else TypeChoix = 0, if TypeChoix = 0 or 1 then set Language to Francais else set language to Anglais, call TIM_ Temporize() to delay .2 sec, loop while down arrow key is pressed, call TIM_ Temporize() to delay .2 sec

- loop while waiting for the Valid key to be pressed
- call TIM_ Temporize() to delay .2 sec
- loop while waiting for the Valid key to be un-pressed
- call TIM_ Temporize() to delay .2 sec
- call DisplayErasePage(1) to clear LCD screen

- If TypeChoix = 0 or 2, then do the following:
    1. call DisplayErasePage()
    2. call TEST_SupportairM2() to execute the following tests:
        TEST_reglage_contraste();
        TEST_InterfaceClavier();
        TEST_ram();
        TEST_rtc();
        TEST_eeprom();
        TEST_flash_Monitoring();
        TEST_flash_Evenements();
        TEST_buzzers();
        TEST_VanneO2();
        TEST_Valve();
        TEST_CapteursDebit();
        TEST_CapteursPression();
        TEST_Fio2();
        TEST_InterfaceAlimCPU();
        TEST_RappelAlarme();
        TEST_InterfaceTurbine();
        TEST_Watchdog();
    3. Call DisplayString() to display 'Test End' and 'Press Valid key to start tests again'
    4. Test and loop if TOUCHE_VALID to not equal 0 while the valid key is pressed
    5. Test and loop if TOUCHE_VALID to equal 0 to wait for valid key to be not pressed
    6. Call BUZ_ThreeBeep() to sound the security buzzer briefly

- If TypeChoix = 1 or 3, then do the following:
    1. Set ExitTestsUnitaires flag to False
    2. call MAIN_First_Page()

MAIN_First_Page ()

Set Choix to 0, Call DisplayErasePage() to clear display, while ExitTestsUnitaires flag is false perform the following:

- call DisplayErasePage() to clear display
- If Choix > 8, Call DisplayString () to display the following (tests):
    - VALVE TEST
    - FLOW SENSORS TEST
    - PRESSURE SENSORS TEST
    - FI02 TEST
    - POWER BOARD INTERFACE TEST
    - EXTERNAL ALARM TEST
    - TURBINE INTERFACE TEST
    - WATCHDOG TEST
    - END OF THE UNIT TESTS
    - >
- Otherwise, call DisplayString () to display the following (tests):
    - LCD CONTRAST TEST
    - RAM TEST
    - RTC TEST
    - EEPROM TEST
    - MONITORING FLASH TEST
    - EVENT FLASH TEST
    - BUZZER TEST
    - 02 VALVE TEST
    - >

- do the following while the Valid key is not pressed:
    1. If the up arrow key is pressed:
        - and Choix > 8 then call DisplayString() to remove '>' at ((Choix**-**8**+**1))*****20
        - else if Choix is not > 8 then call DisplayString() to remove '>' at ((Choix**+**1))*****20
        - if Choix > 1 then decrement Choix
        - else if Choix is not > 1, then Choix = 17
        - if Choix = 8 then DisplayErasePage() to clear display, call DisplayString() to display the following:
            - LCD CONTRAST TEST
            - RAM TEST
            - RTC TEST
            - EEPROM TEST
            - MONITORING FLASH TEST
            - EVENT FLASH TEST
            - BUZZER TEST
            - 02 VALVE TEST

        - if Choix = 8 then DisplayErasePage() to clear display, call DisplayString() to display the following:
            - VALVE TEST
            - FLOW SENSORS TEST
            - PRESSURE SENSORS TEST
            - FI02 TEST
            - POWER BOARD INTERFACE TEST
            - EXTERNAL ALARM TEST
            - TURBINE INTERFACE TEST
            - WATCHDOG TEST
            - END OF THE UNIT TESTS

MAIN_First_Page () continued

- if Choix > 8 then call DisplayString() to display '>' at ((Choix**-**8**+**1))*****20
- else if Choix does not > 8 then call DisplayString() to display '>' at ((Choix**+**1))*****20

- call TIM_ Temporize() to delay .2 sec
- loop while waiting for the Valid key to be un-pressed
- call TIM_ Temporize() to delay .2 sec

2. if the down arrow key is pressed then do the following:
   a. call DisplayString() to remove '>' at line 40
   b. and if Choix > 8 then call DisplayString() to remove '>' at ((Choix**-**8**+**1))*****20
   c. else if Choix is not > 8 then call DisplayString() to remove '>' at ((Choix**+**1))*****20

3. if Choix < 17 then increment Choix
4. if Choix = 9 then , call DisplayString() to display the following:
   - VALVE TEST
   - FLOW SENSORS TEST
   - PRESSURE SENSORS TEST
   - FI02 TEST
   - POWER BOARD INTERFACE TEST
   - EXTERNAL ALARM TEST
   - TURBINE INTERFACE TEST
   - WATCHDOG TEST
   - END OF THE UNIT TESTS

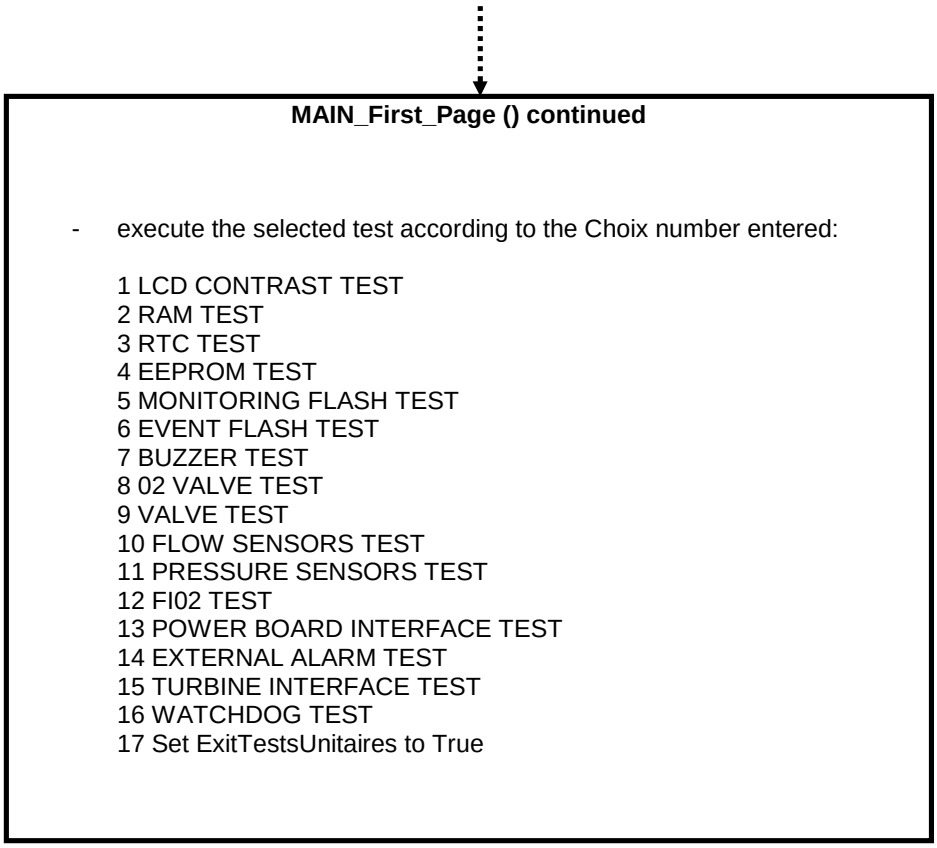
5. if Choix = 1 then , call DisplayString() to display the following
   - LCD CONTRAST TEST
   - RAM TEST
   - RTC TEST
   - EEPROM TEST
   - MONITORING FLASH TEST
   - EVENT FLASH TEST
   - BUZZER TEST
   - 02 VALVE TEST

6. if Choix > 8 then call DisplayString() to display '>' at ((Choix**-**8**+**1))*****20
7. else if Choix does not > 8 then call DisplayString() to display '>' at ((Choix**+**1))*****20

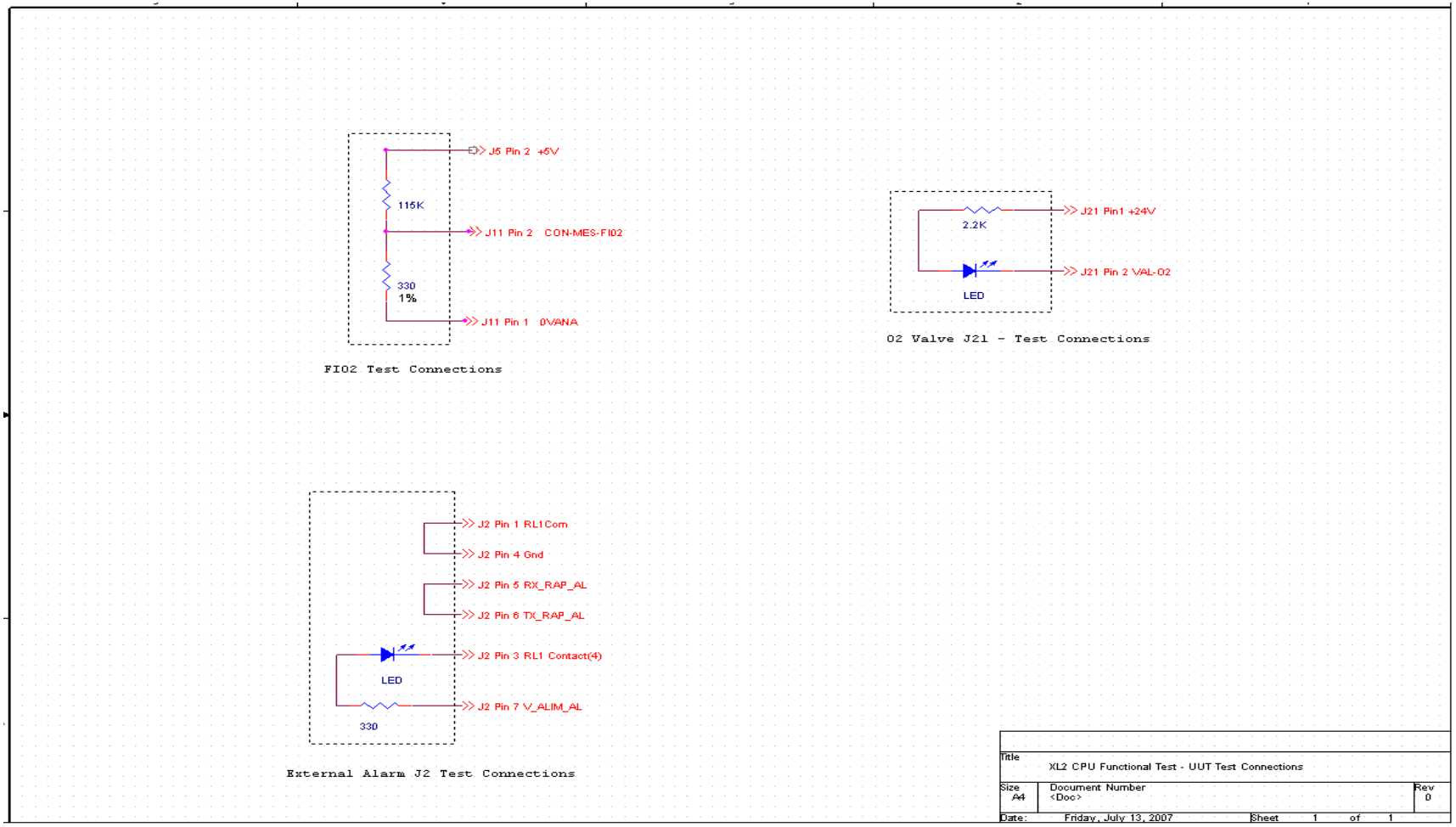
8. call TIM_ Temporize() to delay .2 sec
9. loop while waiting for the Valid key to be un-pressed
10. call TIM_ Temporize() to delay .2 sec

(If ExitTestsUnitaires flag is not true perform the following)

- call TIM_ Temporize() to delay .2 sec
- loop while waiting for the Valid key to be un-pressed
- call TIM_ Temporize() to delay .2 sec
- call DisplayErasePage() to clear display


MAIN_First_Page () continued
- execute the selected test according to the Choix number entered:
1 LCD CONTRAST TEST
2 RAM TEST
3 RTC TEST
4 EEPROM TEST
5 MONITORING FLASH TEST
6 EVENT FLASH TEST
7 BUZZER TEST
8 02 VALVE TEST
9 VALVE TEST
10 FLOW SENSORS TEST
11 PRESSURE SENSORS TEST
12 FI02 TEST
13 POWER BOARD INTERFACE TEST
14 EXTERNAL ALARM TEST
15 TURBINE INTERFACE TEST
16 WATCHDOG TEST
17 Set ExitTestsUnitaires to True

Appendix A UUT Test Connections



Appendix B Test Bench Connections

CPU Test Bench - Test Connections