|  |
| --- |
|  |
| TEST CASES |
| PSU Control |

|  |
| --- |
| ALBERTO IBARRONDO  18/03/2015 |

INDEX

[INTRODUCTION 3](#_Toc414976787)

[INTERFACE 4](#_Toc414976788)

[HARDWARE 4](#_Toc414976789)

[MTTTY INTERFACE 4](#_Toc414976790)

[RDAC – AD5292 4](#_Toc414976791)

[NIELSEN HEURISTICS FOR WINDOWS INTERFACE 5](#_Toc414976792)

[FUNCTIONAL 5](#_Toc414976793)

[CONFIGURATION 5](#_Toc414976794)

[RDAC\_COMPLETE 5](#_Toc414976795)

[RDAC\_CTRL 6](#_Toc414976796)

[RDAC\_CONFIG 7](#_Toc414976797)

[RDAC\_FEEDBACK 8](#_Toc414976798)

[FLASH\_MEM 9](#_Toc414976799)

[COMMUNICATION 10](#_Toc414976800)

[I2C Bus 10](#_Toc414976801)

[SPI BUS 11](#_Toc414976802)

PSU Control - TESt CASES

# INTRODUCTION

This document fully describes every test needed to ensure correct system functioning. Most tests work under a “black box” philosophy, analyzing the outputs to a certain input, without further inspection. The test cases documentation follows a rigid structure (explanation goes in brackets):

|  |  |
| --- | --- |
| TC\_I\_345 (Test Case ID Number, unique for each) | TC\_I\_SOFTWARE\_USER\_INTERFACE\_START\_PROCESS (Test Case ID Name, which inherits the name from the System Requirement it belongs to {in this case it would be SR\_I\_SOFTWARE\_USER\_INTERFACE} and adds a unique ending to it) |
| **Belongs to** | SR\_I\_SOFTWARE\_USER\_INTERFACE (System Requirement the test links to). |
| **Description** | Brief explanation of the test, including its LINK TO THE PROGRAMMED TEST CODE in Test\_PSU.cpp, usually within a specific method. |
| **Required** | Hadrware & Software status at the beginning of the test |
| **Input** | Data do be introduced to the test. Includes the KEYBOARD SHORTCUT to be used in the Test Menu for initialization. |
| **Output** | Data to receive from the test &interpretation of the data.  PASSED: if the output satisfies several conditions  NOT PASSED: otherwise |
| **RESULTS** – Date when the test was passed | Successful/unsuccessful and a proof of it (screenshots or pictures) |

All the software-based tests are stored in the same file: Test\_PSU.cpp (with its associated header file Test\_CPU.h). Hardware testing is also detailed here.

All of the software testing is called inserting the **TestMain()** function into the program’s main loop.

In addition to all the test documentation, a troubleshooting guide will be included for each common error found. The following template will accommodate all this info for each epigraph:

|  |  |
| --- | --- |
| “ERROR 31: Create File” | |
| Description | An explanation why this error occurred |
| Solution | How to solve the error |

# INTERFACE

## HARDWARE

### MTTTY INTERFACE

|  |  |
| --- | --- |
| “ERROR 2: Create File” | |
| Description | The MOD5213 isn’t communicating with the PC. |
| Solution | Check the selected port (COM5 normally used) and the USB connection. If problem persists, extract the MOD5213 from the test board & insert it again. |

### RDAC – AD5292

|  |  |
| --- | --- |
| TC\_I\_001 | TC\_I\_HARDWARE\_RDAC |
| **Belongs to** | SR\_I\_HARDWARE |
| **Description** | Checking with the different software methods that the output voltage varies in the desired ranges. No code available. |
| **Required** | MOD5213 connected to a single SlotCard with I2C bus operational. The SlotCard must be connected to the System Supply. A Voltmeter to monitor the output voltage |
| **Input** | No keyboard shortcut. Requires a changing value for the RDAC Value Register using a loop to cover a desired interval. |
| **Output** | The values registered in the voltmeter. In the results table, the values are:  Vcalc = 1,25\*(20.000\*(1-RP/1023)/800) – Expected Value  Vout = measurement results – Real Value  PASSED: Values follow the mathematic expression. Small relative error  NOT PASSED: Don’t. Big relative error |
| **03/03/15** | SUCCESSFUL – Data stored in  PSUControl/Tests/ TC\_I\_HARDWARE\_RDAC – Results.xlsx |

## software

|  |  |
| --- | --- |
| TRAP in MTTTY | |
| Description | When initializing, the MOD5213 doesn’t work, printing a TRAP everytime |
| Solution | The use of printf seems to cause big trouble. The only reason for using it would be printing a FLOAT value, which now can be done with string ftos(float). |

|  |  |
| --- | --- |
| OSTask doesn’t Work | |
| Description | When using a Task as well as PIT0, the Task doesn’t execute |
| Solution | This is due to the use of PIT0 as a timer interruption, as it’s being used for OSDelay functions. Use PIT1 instead |

## COMMUNICATION

### I2C Bus

|  |  |
| --- | --- |
| TC\_I\_002 | TC\_I\_COMMUNICATION\_I2C |
| **Belongs to** | SR\_I\_ COMMUNICATION |
| **Description** | Checking the I2C bus with the oscilloscope while transmitting any message. No special code required: any I2C test from TC\_F\_COMMUNICATION\_I2C will suffice (Best used with TC\_F\_COMMUNICATION\_I2C\_SIMPLE) |
| **Required** | MOD5213 connected to a single SlotCard with I2C bus operational. The SlotCard must be connected to the System Supply. An oscilloscope with two probes: one for I2C clock bus and the other for I2C Data bus. Trigger set to rising flank in the I2C Clock Bus, with a level far superior to noise. |
| **Input** | Calling any i2C function. Best used with “w” |
| **Output** | Data seen in I2C Data Bus trace.  PASSED: Data seen in I2C Data Bus trace matches the theoretical output.  NOT PASSED: Doesn’t match. |
| **01/03/15** | SUCCESSFUL – No results stored |

### SPI Bus

|  |  |
| --- | --- |
| TC\_I\_003 | TC\_I\_COMMUNICATION\_SPI |
| **Belongs to** | SR\_I\_ COMMUNICATION |
| **Description** | Checking the SPI bus with the oscilloscope while transmitting any message. No special code required: any RDAC test from TC\_F\_CONFIGURATION\_RDAC will suffice. |
| **Required** | MOD5213 connected to a single SlotCard with I2C bus operational. The SlotCard must be connected to the System Supply. An oscilloscope with two probes: one for SPI clock bus and the other for SPI MISO Data bus. Trigger set to rising flank in the SPI Clock Bus, with a level far inferior to noise. |
| **Input** | Calling any SPI function (e.g. RDAC tests) |
| **Output** | Data seen in SPI MISO Data bus trace.  PASSED: Data seen SPI MISO Data bus trace matches the theoretical output (described in RDACLibrary.cpp epigraph BUFFER METHODS  NOT PASSED: Doesn’t match. |
| **01/03/15** | SUCCESSFUL – No results stored |

## NIELSEN HEURISTICS FOR WINDOWS INTERFACE

---PENDING---

# FUNCTIONAL

## MONITORING

|  |  |
| --- | --- |
| TC\_F\_001 | TC\_F\_MONITORING\_SCAN\_VOLTAGE\_SINGLE |
| **Belongs to** | SR\_F\_MONITORING\_SCAN\_VOLTAGE |
| **Description** | Reads a single value from the ADC pin and scales it to voltage. |
| **Required** | MOD5213 connected to a single SlotCard with the ADC pin connected. The SlotCard must be connected to the System Supply. |
| **Input** | “z” – Keyboard shortcut from the Test Menu. |
| **Output** | A voltage Value  PASSED: Console-printed voltage matches to the input voltage  NOT PASSED: Doesn’t match |
|  | NOT YET DEFINED |

|  |  |
| --- | --- |
| TC\_F\_002 | TC\_F\_MONITORING\_SCAN\_CURRENT\_SINGLE |
| **Belongs to** | SR\_F\_MONITORING\_SCAN\_CURRENT |
| **Description** | Reads a single value from the ADC pin and scales it to current. |
| **Required** | MOD5213 connected to a single SlotCard with the ADC pin connected. The SlotCard must be connected to the System Supply. |
| **Input** | “x” – Keyboard shortcut from the Test Menu. |
| **Output** | A current Value  PASSED: Console-printed current matches to the input current  NOT PASSED: Doesn’t match |
|  | NOT YET DEFINED |

## CONFIGURATION

### RDAC\_COMPLETE

|  |  |
| --- | --- |
| TC\_F\_200 | TC\_F\_CONFIGURATION\_RDAC\_BATTERY\_TEST |
| **Belongs to** | SR\_F\_CONFIGURATION\_RDAC\_CTRL  SR\_F\_CONFIGURATION\_RDAC\_CONFIG  SR\_F\_CONFIGURATION\_RDAC\_FEEDBACK |
| **Description** | Battery of tests to check all the RDAC functions (RDACLibrary.cpp). Includes the option to measure output values when a change is executed. Code is in function **BATTERY\_TEST\_RDAC**() in Test\_PSU.cpp |
| **Required** | MOD5213 connected to a single SlotCard with I2C bus operational. The SlotCard must be connected to the System Supply. |
| **Input** | “u” – Keyboard shortcut from the Test Menu.  “1”/”2” – Select the slave to be tested (Upper o Lower).  “y”/”n” – Check the output voltage during the test (with a voltmeter)  A new value for the output voltage. |
| **Output** | A detailed list of the results for each function.  PASSED: Self-explanatory  NOT PASSED: Self-explanatory |
| **13/03/15** | SUCCESSFUL – Complete Screenshots in PSUControl/Tests/TC\_F\_CONFIGURATION\_RDAC – Results.docx |

### RDAC\_CTRL

|  |  |
| --- | --- |
| TC\_F\_201 | TC\_F\_CONFIGURATION\_RDAC\_CTRL\_SIMPLE |
| **Belongs to** | SR\_F\_CONFIGURATION\_RDAC\_CTRL |
| **Description** | Test for **setCtrlRDAC()** function, sending a “1” to allow RDAC value updating, then a “0” to prevent it (alternatively) |
| **Required** | MOD5213 connected to a single SlotCard with I2C bus operational. The SlotCard must be connected to the System Supply. |
| **Input** | “3” – Keyboard shortcut from the Test Menu. |
| **Output** | NONE – Should be used with Keyboard shortcut “4” to check for changes. In this case:  PASSED: “Register programming” switches from “Disabled” to “Enabled” in the console output. A 2 is received in the last 4 bits of the “Data Received” when “Enabled”. A 0 is received in the last 4 bits of the “Data Received” when “Disabled”  NOT PASSED: Doesn’t change. |
| **11/03/15** | SUCCESSFUL – Complete Screenshots in PSUControl/Tests/TC\_F\_CONFIGURATION\_RDAC – Results.docx |

### RDAC\_VALUE

|  |  |
| --- | --- |
| TC\_F\_202 | TC\_F\_CONFIGURATION\_RDAC\_VALUE\_SIMPLE |
| **Belongs to** | SR\_F\_CONFIGURATION\_RDAC\_VALUE |
| **Description** | Test for **setValRDAC()** function, sending a new value from 0 to 1023 to the RDAC. It also uses the **scanFloatValue**() function to scan the desired value in Volts (In a range from 1.25 to 32), prompting the user to write a value. |
| **Required** | MOD5213 connected to a single SlotCard with I2C bus operational. The SlotCard must be connected to the System Supply. |
| **Input** | “1” – Keyboard shortcut from the Test Menu. A new value in volts when asked. |
| **Output** | NONE – Should be used with Keyboard shortcut “2” to check for changes, or with a voltmeter to monitor the output voltage. In this case:  PASSED: Values change. Data Received = Hex value we programmed (840=0x348)  NOT PASSED: Don’t change. |
| **11/03/15** | SUCCESSFUL – Complete Screenshots in PSUControl/Tests/TC\_F\_CONFIGURATION\_RDAC – Results.docx |

|  |  |
| --- | --- |
| TC\_F\_203 | TC\_F\_CONFIGURATION\_RDAC\_VALUE\_RESET |
| **Belongs to** | SR\_F\_CONFIGURATION\_RDAC\_ VALUE |
| **Description** | Test for **resetRDAC()** function, setting the value 0x200 (midscale, 16V) to the RDAC. |
| **Required** | MOD5213 connected to a single SlotCard with I2C bus operational. The SlotCard must be connected to the System Supply. |
| **Input** | “5” – Keyboard shortcut from the Test Menu |
| **Output** | NONE – Should be used with Keyboard shortcut “2” to check for changes, or with a voltmeter to monitor the output voltage. In this case:  PASSED: Voltage changes to ~16.8V. Data Received = 0x200  NOT PASSED: Other. |
| **11/03/15** | SUCCESSFUL – Complete Screenshots in PSUControl/Tests/TC\_F\_CONFIGURATION\_RDAC – Results.docx  Voltage output was checked with a voltmeter to be 16.8 V |

|  |  |
| --- | --- |
| Values don’t change | |
| Description | The RDAC values don’t change, but the SPI connection is working. |
| Solution | ¿Have you configured Control Register to allow value updating? Do so. If it still doesn’t work, try a different set of SPI channel configuration values, lowering the frequency (the RDAC might not be reading the orders due to a way-too-fast SPI channel)  If your value is always 0, check the upcoming section: RDAC\_FEEDBACK |

### RDAC\_FEEDBACK

|  |  |
| --- | --- |
| TC\_F\_204 | TC\_F\_CONFIGURATION\_RDAC\_FEEDBACK\_CONTROL |
| **Belongs to** | SR\_F\_CONFIGURATION\_RDAC\_FEEDBACK |
| **Description** | Checking the **getCtrlRDAC()** function, reading the Control Register from the RDAC. |
| **Required** | MOD5213 connected to a single SlotCard with I2C bus operational. The SlotCard must be connected to the System Supply. |
| **Input** | “4” – Keyboard shortcut from the Test Menu |
| **Output** | PASSED: “Register programming” switches from “Disabled” to “Enabled” in the console output. A 2 is received in the last 4 bits of the “Data Received” when “Enabled”. A 0 is received in the last 4 bits of the “Data Received” when “Disabled”. Best used with command “3” to guarantee proper working.  NOT PASSED: Other Values. Everything set to 0. |
| **11/03/15** | SUCCESSFUL – Complete Screenshots in PSUControl/Tests/TC\_F\_CONFIGURATION\_RDAC – Results.docx |

|  |  |
| --- | --- |
| Received Values are always 0 | |
| Description | The RDAC values change, but no answer is received |
| Solution | The MISO channel isn’t working properly. Check if there is a pull up resistor (2K2), as there should be one. ¿Have you set the other slave to HIGH IMPEDANCE with the function **highImpRDAC()**? To receive an answer from one slave, the other must be set in high impedance mode. |

|  |  |
| --- | --- |
| TC\_F\_205 | TC\_F\_CONFIGURATION\_RDAC\_FEEDBACK\_VALUE |
| **Belongs to** | SR\_F\_CONFIGURATION\_RDAC\_FEEDBACK |
| **Description** | Checking the **getValRDAC()** function, reading the Value Register from the RDAC. Best used by setting a concrete value first (or resetting to value 0x200). |
| **Required** | MOD5213 connected to a single SlotCard with I2C bus operational. The SlotCard must be connected to the System Supply. |
| **Input** | “2” – Keyboard shortcut from the Test Menu. Best used with previous command “1” |
| **Output** | PASSED: Value matches the programmed one.  NOT PASSED: Other Values. Everything set to 0. |
| **11/03/15** | SUCCESSFUL – Complete Screenshots in PSUControl/Tests/TC\_F\_CONFIGURATION\_RDAC – Results.docx |

|  |  |
| --- | --- |
| TC\_F\_206 | TC\_F\_CONFIGURATION\_RDAC\_20TP\_PROGRAMMING |
| **Belongs to** |  |
| **Description** |  |
| **Required** | MOD5213 connected to a single SlotCard with I2C bus operational. The SlotCard must be connected to the System Supply. |
| **Input** |  |
| **Output** |  |
| **27/04/15** | SUCCESSFUL – Complete Screenshots in PSUControl/Tests/TC\_F\_CONFIGURATION\_RDAC – Results.docx |

### PSU\_TYPE

|  |  |
| --- | --- |
| SR\_F\_206 TC\_F\_CONFIGURATION\_PSU\_TYPE\_PRINT | |
| **Belongs to** | SR\_ F\_CONFIGURATION\_PSU\_TYPE |
| **Description** | Test for the struct PSU\_TYPE that will be used in the rest of the system. Prints all the values for a PSU. No code defined. |
| **Required** | MOD5213. |
| **Input** | - |
| **Output** | Console output  PASSED: Complete & functional structure. Values change; objects can be accessed, as well as their parameters.  NOT PASSED: Otherwise |
| **24/02/15** | SUCCESSFUL – Complete Screenshots in PSUControl/Tests/ TC\_F\_CONFIGURATION\_PSU\_TYPE - Results |

### FLASH\_MEM

|  |  |
| --- | --- |
| SR\_F\_207 TC\_F\_CONFIGURATION\_FLASH\_MEM\_BATTERY | |
| **Belongs to** | SR\_F\_CONFIGURATION\_FLASH\_MEM |
| **Description** | Test for flash memory saving & recovering functions from PSULibrary.cpp (**LoadFlashValuesPSU, saveInFlashValuesPSU, readFlashValuesPSU).** Also checks the auxiliary methods to set default values and print values for PSU\_TYPE objects. (**defaultValuesPSU, printValuesPSU).** Battery test implemented in **BATTERY\_TEST\_FLASH\_MEM** (). |
| **Required** | MOD5213. |
| **Input** | “y” – Keyboard shortcut from the Test Menu |
| **Output** | A detailed list of results for all functionalities.  PASSED: Self-explanatory  NOT PASSED: Self-explanatory |
| **24/03/15** | SUCCESSFUL – Complete Screenshots in PSUControl/Tests/TC\_F\_CONFIGURATION\_RDAC – Results.docx |

|  |  |
| --- | --- |
| Printf doesn’t print values | |
| Description | When invoked, output method printf() displays correctly all the text, but generates a trap when sending a value. It might be due to a failure in the FIFO piles of the MCF. |
| Solution | Use the newly created function ftos() to convert a float to string, and print it with iprintf() in a %s parameter. |

## COMMUNICATION

### I2C Bus

I2C bus communication can be indirectly checked by running any of the RDAC tests (as they require I2C communication to work). Some simple tests for I2C have been defined below, as well as I2C addressing test for I2CtoSPIBridge.

|  |  |
| --- | --- |
| SR\_F\_007 TC\_F\_COMMUNICATION\_I2C\_SIMPLE | |
| **Belongs to** | TC\_F\_COMMUNICATION\_I2C |
| **Description** | Test for I2C communication, selecting a I2C address destination and sending a short message that is prompted by the user. |
| **Required** | MOD5213 connected to a single SlotCard with I2C bus operational. |
| **Input** | “w” – Keyboard shortcut from the Test Menu |
| **Output** | The result of I2C communication  PASSED: message “I2C OK”  NOT PASSED: message “I2C ERROR [1-9]” |
|  | NOT PASSED |

|  |  |
| --- | --- |
| SR\_F\_301 TC\_F\_COMMUNICATION\_I2C\_SPI\_CONFIG | |
| **Belongs to** | TC\_F\_COMMUNICATION\_I2C |
| **Description** | Shortest test for I2C communication, sending an I2C predefined message. |
| **Required** | MOD5213 connected to a single SlotCard with I2C bus operational. |
| **Input** | “q” – Keyboard shortcut from the Test Menu |
| **Output** | The result of I2C communication  PASSED: message “I2C OK”  NOT PASSED: message “I2C ERROR [1-9]” |
| **24/03/15** | SUCCESSFUL – Complete Screenshots in PSUControl/Tests/TC\_F\_COMMUNICATION\_I2C – Results.docx |

|  |  |
| --- | --- |
| SR\_F\_302 TC\_F\_COMMUNICATION\_I2C\_ADDRESSING | |
| **Belongs to** | TC\_F\_COMMUNICATION\_I2C |
| **Description** | Test for I2C communication with the I2CtoSPIBridge, changing its address from 0x28 to 0x2F using the 3 switches and rebooting the system. Battery test implemented in **BATTERY\_TEST\_I2C** (). |
| **Required** | MOD5213 connected to a single SlotCard with I2C bus operational. The SlotCard must be connected to the System Supply. |
| **Input** | “t” – Keyboard shortcut from the Test Menu |
| **Output** | PASSED: Self-explanatory  NOT PASSED: Self-explanatory |
| **24/03/15** | SUCCESSFUL – Complete Screenshots in PSUControl/Tests/TC\_F\_COMMUNICATION\_I2C – Results.docx |

### SPI BUS

Any RDAC function can be used to check for SPI communication. SPI bus works if the “Receive Buffer” is not empty.

|  |  |
| --- | --- |
| SR\_F\_303 TC\_F\_COMMUNICATION\_SPI\_SIMPLE | |
| **Belongs to** | TC\_F\_COMMUNICATION\_SPI |
| **Description** | Test for SPI communication. Doesn’t have code. It’s indirectly passed by using any of the RDAC tests |
| **Required** | MOD5213 connected to a single SlotCard with I2C bus operational. |
| **Input** | “[1-5]” – Keyboard shortcut from the Test Menu |
| **Output** | The result of SPI communication, saved in the I2CtoSPIBridge buffer  PASSED: the “receive Buffer” is not empty.  NOT PASSED: Empty always. |
| **24/03/15** | SUCCESSFUL |