Developing the Regressive Test Suite for DS1977 Device (iButton)

Varsha Basalingappa Doshetty[1], Praveen Kumar[2]
[1]*M.Tech, VLSI and Embedded system, SCEM, Mangalore*
[2]*Asst. Prof. Department of ECE, SCEM, Mangalore*

Abstract **- An iButton is a password protected data storage, which has wide range of applications in security systems and it also acts as an authentication device. Each DS1977 device have unique factory lasered ID which cannot be changed. The aim of this paper is to develop a regressive test suite for DS1977 device which checks the operation of the device and functionalities of each command supported by it. By this proposed automated and regressive test suite it allows faster validation of DS1977 or any such similar devices designed as per the iButton/one-wire communication protocol.**

Index Terms- **iButton, memory commands, functionality check, test suite.**

I. INTRODUCTION

DS1977 device is a 32KB EEPROM data storage in which access to its memory is password protected with Read access and Full access passwords. It has greater importance in maintenance/inspection data storage, medical data carrier and audit data carrier. It also acts as an authentication device since each device has a unique ID which cannot be tampered with. To implement all the functionalities it has a set of commands for internal memory and internal ROM access. Initialization to the single wire communication always begins with master sending a Reset pulse for 480-640µs duration [1] and releasing the line. As soon as rising edge of the data line is found, slave device will acknowledge its presence on the line by sending Presence pulse for 60 to 240µs [1]. Write zero and Write one are the basic building blocks for each of the commands transmitted; Read 0 and Read 1 are the basic building blocks for reading data. Write 0, write 1, Read 0/1 are differentiated among themselves by the duration for which the one wire communication line is kept low during transmission and reception.

DS1977 supports several ROM commands and Memory Access commands. Complete communication through 1-wire protocol happens in 3 levels. In the first level slave device should acknowledge its presence to the bus master. In the second level the slave devices should execute any one of the ROM commands. After a ROM command gets executed successfully, the memory and control function of the slave devices become accessible which is third level. At this level slave device should execute any one among six memory commands.

Write Scratchpad, Read Scratchpad, Copy Scratchpad, Read Memory, Verify Password and Read Version.

II. IBUTTON FUNCTIONAL TEST SUITE

To write data to the DS1977 memory, an intermediate storage scratchpad has to be used. The flow in which data is to be written to the memory is explained below. Initially the master should issue write scratchpad command by providing target address and data, the data will be written to the 64-byte scratchpad. To verify the data written to the scratchpad, read scratchpad command is used. Then the scratchpad content has to be copied to the memory by using copy scratchpad command. While giving copy scratchpad command master has to send the address to which data to be copied, offset register and acceptable password. To read back the data from DS1977 memory, master should send read memory command along with read access password. The device having a password control register. If 0xaa pattern is written to the register then only password gets enabled. Any pattern other than 0xaa, disables the password. If the passwords are not enabled, dummy passwords are to be provided for proper operation of all the commands.

The following section describes the tests for the various commands supported by the DS1977 device.

A. Write scratchpad

WSP1: Writing the scratchpad from the starting address of the page and writing all the 64 bytes with different types of data patterns and reading the CRC bytes. The aim of this test is to verify that the CRC is being sent by the iButton at the end of every page read starting from the beginning of the page and it is correct or not.

WSP2: Writing the scratchpad from the middle of the page with different types of data patterns and reading the CRC bytes. The aim of this test is to verify that the CRC is being sent by the iButton at the end of every page read starting from the middle of the page and it is correct or not.

WSP3: Writing the scratchpad from the middle of the page and not reaching till the end of the page. This test is to ensure that the CRC will be generated only if the end of the page is reached. So for this test no CRC should be generated.

WSP4: Writing the scratchpad with data being 2 bytes and 3 bits and checking whether partial flag in ES is set or not. This test proves writing incomplete byte sets the partial flag in ES and it writes only 2 bytes.

WSP5: Doing null bytes writing to the scratchpad and checking the value in offset register. The aim of this test is to verify that the offset register contains 0 as offset when the number of locations parameter is set to 0 for write scratchpad command.

WSP6: Writing scratchpad at the address somewhere in between 0x7FD1 to 0x7FFF and writing fewer bytes (10 bytes). This test aims to check what happens if data is written in the range where there is no user access.

In figure.1 test result of one of the write scratchpad tests is shown. Written target address, data and CRC bytes are compared against read values.

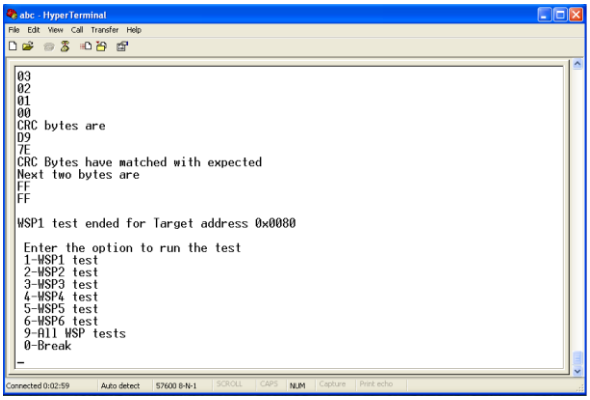

Fig.1. WSP1 test results

B. Copy scratchpad

CSP1: Writing scratchpad till the end of page and reading CRC bytes. Doing read scratchpad and then copy scratchpad. Then doing read memory for that page. This test is to verify that the CRC is being sent by the iButton at the end of every page read starting from the beginning of the page and it is correct or not.

CSP2: Writing scratchpad from the middle of the page and reading CRC bytes. Doing read scratchpad and copy scratchpad. Then read memory for that page. This test is to verify that the CRC is being sent by the iButton at the end of every page read starting from the middle of the page and it is correct or not.

CSP3: Writing scratchpad from the middle of the page and not reaching till the end of the page. Doing read scratchpad and copy scratchpad. Then read memory for that page. This test ensures that CRC cannot be generated if end of page write is not reached.

CSP4: Doing write scratchpad with sequential data pattern, writing fewer bytes followed by read scratchpad. Then while copy scratchpad, incorrect TA1 and correct TA2 are given. 0xff pattern is observed because of wrong TA1 entered. Then doing read memory for that page and verify the data in the memory. Similar tests are done with incorrect TA2 and incorrect password. The aim of this test is to verify that for unsuccessful copy scratchpad operation the read memory command should result in sending the previous correct stored in the memory and the new data entered with unsuccessful copy scratchpad operation should not be written in the memory.

CSP5:Writing the scratchpad with data 2 bytes and 3 bits only, followed by read scratchpad command and checking whether partial flag in ES is set or not. Writing incomplete byte to the scratchpad sets the partial flag in offset register. The pattern generated while copy scratchpad is checked.

CSP6: Writing the scratchpad at the address above 0x7fff with fewer bytes data. After copy scratchpad, data from the memory are read and checked. This test proves that one cannot write data at address 0x7fff because there is no user access.

In the integrated test suite, for write scratchpad and copy scratchpad user can choose between default available data or can provide user data. Figure 1 shows the menu for which the user data is chosen to carry out the write scratchpad test.

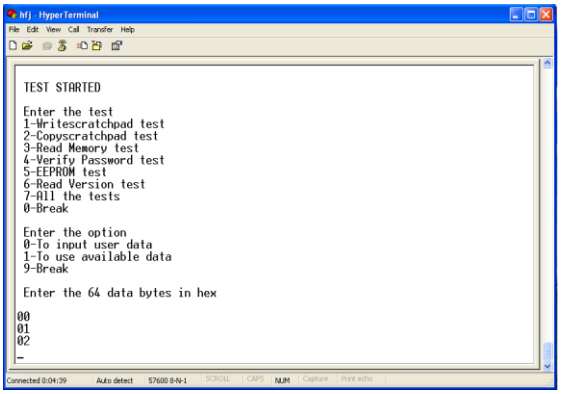

Fig.1. Write scratchpad test for user data

C. Read memory

RM1: Writing all the pages of the memory with data pattern 0000_0000_0000_0000 for page0, 0000_0000_0000_0001 for page 1 and so on till page 510 considering 16 bits at a time with correct Read access password and Full access password. Then copy data to the memory using copy scratchpad and reading memory at page5. This test is to check whether one can read from any page of the memory or not and writing to one page does not modify the content of any other page.

RM2: Reading the memory continuously from page0 to 510 with correct Read access password and Full access password. The read data from the memory are compared with the expected values. Again reading memory continuously from random page numbers till 510 pages.

RM3: Reading from page 511 to next 10 pages to check the response beyond page 511with correct Read access password and Full access password. Since there are only 511 pages in the memory, reading memory beyond it should give 0xff data.

RM4: RM1, RM2 tests are done by giving wrong Read access password and Full access password.

D. Verify Password

VP1: Read access password and Full access passwords are programmed to the address 0x7fc0 and 0x7fc8. Verify password command is checked by providing correct Read access password with Read access password address.

VP2: The above test is carried out by providing incorrect Read access password with Read access password address one at a time, correct full access password with its address check and incorrect full access password with its address check. Success or failure of the tests is checked as per the specification [1].

VP3: EEPROM memory is programmed from 0x7fc0 to 0x7fcf with data pattern being counter using write scratchpad, read scratchpad and copy scratchpad. Verify password command is checked by providing Read access password address and the counter data.

VP4: EEPROM memory is programmed from 0x7fc0 to 0x7fcf with all the data being 0x00, 0xff, 0x55 and 0xaa one at a time. Verify password command is checked by providing Full access password.

E. 32KB EEPROM

EEPROM1: Writing each page of the memory with continuous bytes 0x00, password being enabled and correct full access password is provided.

EEPROM2: Writing each page of the memory with continuous bytes 0xff, password being enabled and correct full access password is provided.

EEPROM3: Writing each page of the memory with continuous bytes 0xaa, password being enabled and correct full access password is provided.

EEPROM4: Writing each page of the memory with continuous bytes 0x55, password being enabled and correct full access password is provided.

III. IBUTTON TEST SUITE USER INTERFACE

The above tests are done by the following test setup The C code is written to all the tests which include complete functionality and is compiled. The code generates the SREC file using gcc based tool chain. Then the SREC file is loaded to the hardware platform via UART interface using HyperTerminal on PC. Response of the device is checked by generating log for the tests.

All the tests are combined into one to generate an integrated test suite which when loaded and executed validates the entire functionality of each of the commands and generates a log of the tests done and its responses. The test suite has to do the exhaustive testing of all the functionalities so that the device can be certified at one go as working or non-working.

The automated test suite designed provides the menu based user interface option for the user to enable selective or full run of the tests planned in one go. It also includes the sub menu which provides option to select the test cases under each of the tests.

In the menu, one more option is provided either to take default data pattern or to give user data pattern for write scratchpad and copy scratchpad tests.

The menu designed to provide user interface option to run the test as per the wish is shown in Figure 3.

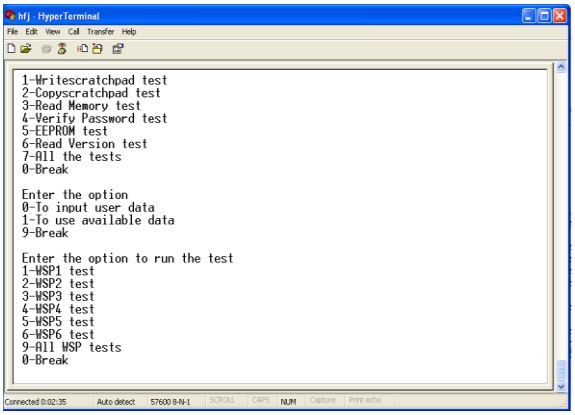

Fig.3. Menu that provides user to selectively run the test

IV. CONCLUSION

By developing the regressive test suite for iButton one can test the complete functionality of device for all the conditions at one go and can be deployed as per the requirement. The test suite does the exhaustive testing of DS1977 or any such similar devices designed as per the iButton/one-wire communication protocol.

REFERENCES

[1] DS1977 user manual by Dallas Semiconductors.
[2] C Handbook.
[3] Tool Chain Manual for compiling the code.
[4] Using CRC checks with Dallas iButton- Dallas app note.
[5] One-wire Design Guide