Design Document for Test Hook

Table of Contents

[1. Test Hooks 2](#_Toc103608940)

[1.1 Reference 2](#_Toc103608941)

[1.2 Abbreviations/Glossary 2](#_Toc103608942)

[1.3 Scope 2](#_Toc103608943)

[1.4 Overview 2](#_Toc103608944)

[1.5 Test Manager State Machine 3](#_Toc103608945)

[1.5.1 Idle state 3](#_Toc103608946)

[1.5.2 Activate Test-Mode state 3](#_Toc103608947)

[1.5.2.1 Calculating the key 3](#_Toc103608948)

[1.5.3 Test Mode 4](#_Toc103608949)

[1.6 Source code specifications 6](#_Toc103608950)

[1.6.1 Constraints/Assumption 6](#_Toc103608951)

[1.6.2 File Names 6](#_Toc103608952)

[1.6.3 Enumerations 6](#_Toc103608953)

[1.6.4 Public function Interface 6](#_Toc103608954)

# Test Hooks

## Reference

* R0055200 Gen II Communications Protocol
* RE00332057  Signia Rearchitecture Communications Protocol

## Abbreviations/Glossary

* MCU – Microcontroller Unit (Freescale K20)
* Test Mode – Operation mode of signia handle where in various error/operation conditions can be triggered via external commands (via USB/WIFI)

## Scope

The document outlines the concept of Test Hooks in the Signia Rearch software. The document also covers the entry and exit of Signia handle from Test Mode and the modules involved in the operations.

The document lists the possible test scenarios for the implementing Test Hook.

## Overview

The purpose of Test Hooks is to induce test specific behavior of the software under test example error and/or boundary conditions etc...

The Handle software shall enter Test Mode to execute a test. Entry to test mode shall be via a command ( SerialCMD\_TEST ) over USB/WIFI.

Console Manager upon receiving the command sequence SerialCMD\_TEST triggers an event to Test-Manager along with the received key and Test id that needed to be executed.

Test-Manager is Responsible for handling entry to Test mode, executing the testcase based on the request, and exiting the test mode

Figure Software design

Console Manager

Test Manager (Active Object)

Tester Event with data

TM\_Hook()

Test case 1

Test case 2

Test case n

Module 1

TM State Machine

Module 2

Module n

Activate Test

Modules shall call the TM\_Hook() interface function at points of fault injection. Upon activation of the test mode, based on the test id received the test manager executes the test case on the required Module.

If needed TestManager can be removed from the release build. One way this can be done is by having a weak function for the TM\_Hook() function, other way would be encapsulate the calls and TestManager with compiler switches.

## Test Manager State Machine

SERIALCMD\_TEST

Timeout/ invalid key

Activation success

Timeout

Figure Test Manager State Machine

## Idle state

Default state for Test Manager when Handle is in normal operation. In this state no action is done from Test Manager and waits for test mode command.

## Activate Test-Mode state

In this state the Test Manager checks for the conditions to activate Test mode.

Create a new command, SERIALCMD\_TEST. MCP sends the command SERIALCMD\_TEST, with a two byte Key, test id to be activated and data (if any based on test id). The key is calculated based on the handle’s serial number. Handle upon receiving the test command, should calculate the key and if the calculated key matches with the received key then the test mode is activated and echoes the command back to MCP.

### Calculating the key

Read the Serial number from handle by using the command SERIALCMD\_GET\_SERIALNUM. Sum the received serial number (word wise), take the inverse of it, and XOR that value with the special serial interface key (ex: 0x53 0x69 0x67 0x6E 0x69 0x61, which is “Signia”).

Example:

Handle Serial number : C17AAF0061 (0x43 0x31 0x37 0x41 0x41 0x46 0x30 0x30 0x36 0x31)

Special serial key : Signia (0x53 0x69, 0x67 0x6E, 0x69 0x61)

Step1 (calculating the sum): 0x4331

+ 0x3741

+ 0x4146

+ 0x3030

+ 0x3631

= 0x12219

= 0x2219 (ignore the overflow)

Step2 (Inverse of Sum): = 0xDDE6

Step 3 (Xor with Serial key):

= 0xDDE6 ^ 0x5369 = 0x8E8F

0x8E8F ^ 0x676E = 0xE9E1

0xE9E1 ^ 0x6961 = 0x8080

Final Key value = 0x8080

The goal is to ensure deliberate activation of the test hook without the complexity of a complex encryption. That way, the test driver on the PC side needs to know something about the handle before initiating.

Command = SERIALCMD\_TEST

Parameter = Key (2 bytes)

Parameter = Test mode

Value = data values if any

Command = SERIALCMD\_GET\_SERIALNUM

Data = Handle Serial number (Null terminated string)

Command = SERIALCMD\_GET\_SERIALNUM

MCP

Handle

Run Algorithm to Calculate Key

If received key matches with calculated key then Test Mode Active.

Each test mode shall timeout in specified time

Echo Command if Test mode successfully activated

Run Algorithm to Calculate Key

Figure Test mode activation by key

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Start (1 byte)** | **Length (2 bytes)** | **Command**  **(1 byte)** | **Key (2byte)** | **Parameter (Test hook id – 1byte)** | **Value**  **(number of bytes depending on Parameter)** | **CRC (2 bytes)** |
| **0xAA** | **0x0009 + number of bytes in Value** | **SERIALCMD\_TEST** | **0xA585** | **0x01 to 0xFF** | **0x00 0x00 …** | **<CRC>** |

Table MCP to Handle: Using Key to activate Test Mode

## Test Mode

In this state the requested Test case is executed by the Test Manager.

Depending upon the Test case Test Manager shall do one of the following to induce the error. (The list is not exhaustive and shall be extended/modified as the per need)

1. Run a Test stub
2. To update Cyclic data, the Modules shall call the TM\_Hook() interface to update the data .
3. For test cases that run only during startup, NoInitRam shall hold the request for testmode and testcase number and upon reset the testcase shall be executed. Subsequent reset shall bring the handle out of test mode. (Note: NoInitRam variables shall be cleared after first reset, this is to avoid getting into test mode in subsequent resets)

Once in test mode, any subsequent SERIALCMD\_TEST command should be ignored. Software shall come out of test mode either after timeout and perform a soft reset, or upon a HardReset.

As of now Test cases Identified for which Test Hooks shall be implemented are listed below . Each test case shall be assigned with a test hook id.

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Test Category** | **Test Case** | **Possible Test Method** |
| 1 | OS Task ERRORS |  | Run a Test stub to produce errors |
| 2 | Task Monitor | 1. Normal operation - watchdog reset test for 250ms |  |
| 1. Faults - Stack freespace < 10% | Run a Test stub to produce errors |
| 1. Faults - CPU utilization > 90% | Run a Test stub to produce errors |
| 1. Faults - Task check in failure | Run a Test stub to produce errors |
| 3 | SD Card Read / Write performance |  |  |
| 4 | USB communication Performance |  |  |
| 5 | GPIO Expander faults |  |  |
| 6 | Memory Faults | 1. Integrity of Program code |  |
| 2. Integrity of Internal SRAM |  |
| 3. Integrity of External SRAM |  |
| 7 | File system error simulation (ST\_ERR\_FILE\_SYS ) |  |  |
| 8 | Watchdog Initialization failure |  |  |
| 9 | OLED self-test failure |  |  |
| 10 | One wire Master chip com failure |  |  |
| 11 | 1-wire device write test (failure condition simulation Read failure, write failure) | 1. Handle during startup |  |
| 2. Other devices during connection detection |  |
| 12 | 1 wire authentication fail |  |  |
| 13 | Simulating Battery Parameters |  |  |
| 14 | Simulate SMBus failures | 1. ST\_ERR\_BATT\_COM: Battery DF write fails |  |
| 15 | Adapter | 1. Communication test fail |  |
| 2. Read and write hooks for Adapter parameters |  |
| 16 | Key Strokes | Up key toggle, Down key toggle,  Left Up key toggle, Left Down Key toggle,  Right Up key toggle, Right Down Key toggle,  Left safety key toggle, Right safety key toggle, |  |
| 17 | Display screen | Display screen with screen id |  |
| 18 | Play Tone | Play Tone with Tone id |  |
| 19 | Motor speed Test | Move motor with 25RPM <= SetSpeed <= 1800 |  |
| 20 | Motor Performance Test | The HANDLE software shall control the MOTOR speed within 10% of the setpoint, within 200 ms of the setpoint command, while operating at 25 RPM <= MOTOR speed <= 1800 RPM. |  |

Table Test Case

The result of each test case shall be logged on to MCP / Event Log with a tag “Test:” before every message.

## Source code specifications

### Constraints/Assumption

### File Names

|  |  |
| --- | --- |
| **Item** | **Details** |
| TestManager.c | Test Manager Source code (part of Utils folder) |
| TestManager.h | Test Manager Header file |

### Enumerations

| **Item** | **Details** |
| --- | --- |
| typedef enum {  CHARGERMANAGER,  ONEWIRE,  ADAPTERUART,  …,  …  } MODULEID; | Modules that shall use the TM\_Hook interface |
| typedef enum{  TC\_TASKMONITOR\_STACKFAULT,  TC\_TASKMONITOR\_CPUUTILIFAULT,  TC\_TASKMONITOR\_TASKCHKFAIL,  TC\_ADAPTER\_COMFAIL,  TC\_SIM\_BATTPARAM\_TS1,  TC\_SIM\_BATTPARAM\_TS2,  TC\_SIM\_BATTPARAM\_VOLTAGE,  TC\_SIM\_BATTPARAM\_TEMP,  ...,  ...  } TESTCASE; | Enumeration for Testcase, this should correspond with the parameter bitfield received over USB. |

### Public function Interface

| **Function** | **Details** |
| --- | --- |
| void **TM\_Hook**(MODULEID id, void \*pData) | Test hook function shall be called from each module from appropriate place to update the cyclic data.  Based on the Module ID and the activated test hook the function shall typecast the pData to appropriate user datatype and update the data |