

IPMI/IPMB Satellite Controller Test Procedure

On the pages that follow, you will find the test procedure for the IPMI/IPMB Satellite Controller used in CompactPCI power supplies. The firmware was "developed by Philips Semiconductors using their P89C662HBA/00 controller. Detailed" information on this controller, including features, complete specifications, application notes, parametrics, support and tools, can be found at <http://www.semiconductors.philips.com/pip/p89c662hba/00>.

Philips Semiconductors

IPMI/IPMB Satellite Controller Test Procedure

*James P. Earle
Market Applications Engineer
Philips Semiconductors
2140 Lake Park Blvd., Suite 200
Richardson, Texas 75230
972-705-2485
jim.earle@philips.com*

IPMI/IPMB Satellite Controller Functional Test

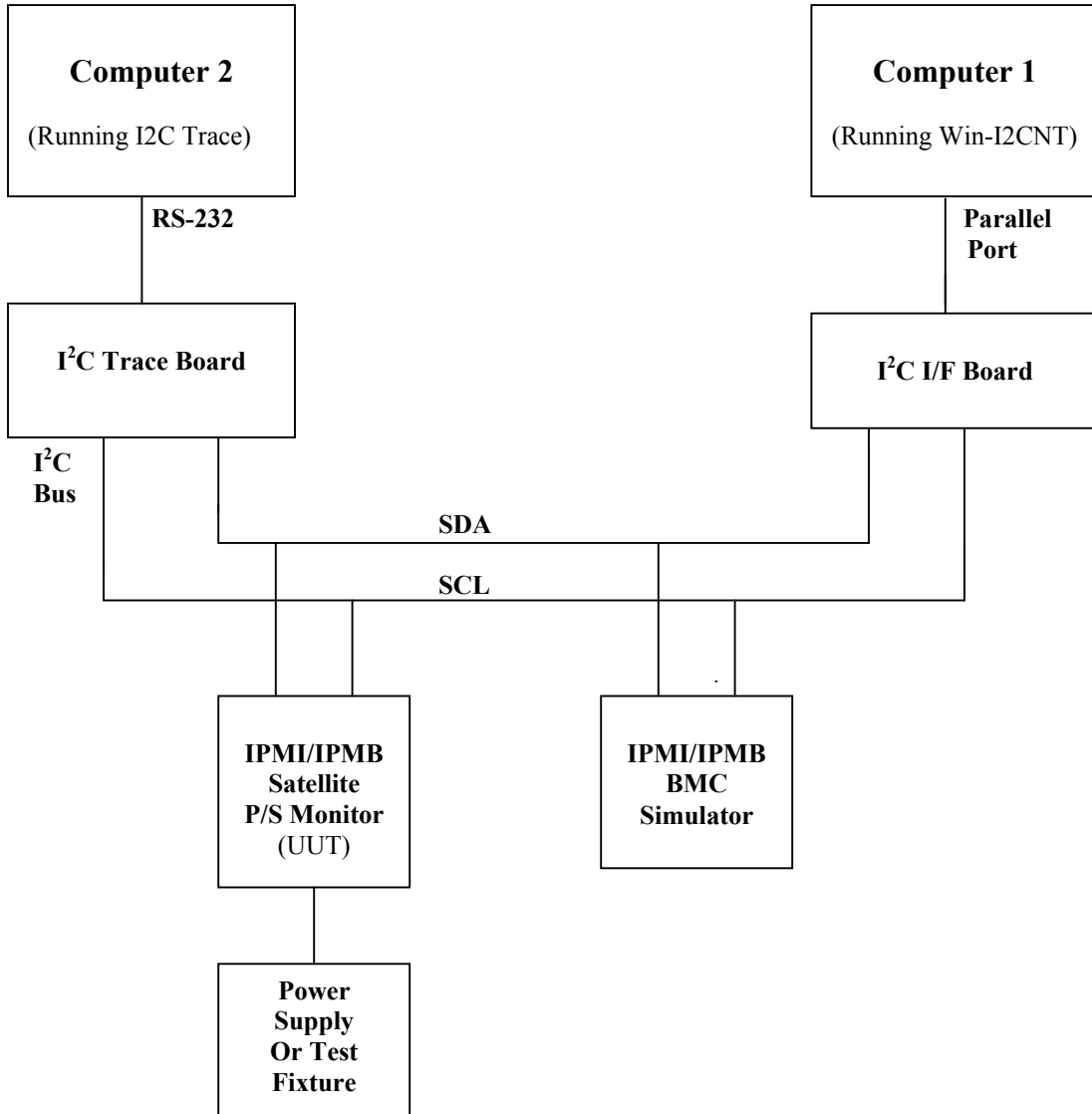


Figure 1. Test Set up

Test Description

The purpose of this test procedure is to functionally test and evaluate the IPMI/IPMB interface for a power supply in an IPMI system. The IPMI protocol is a request / response protocol. The IPMI/IPMB satellite controller being tested will typically communicate with the baseboard master controller since it manages all of the sensor data records and event logs. IPMI/IPMB protocol provides for other devices to also be able to interrogate satellite controllers, event data logs, etc. However, the test described here, will only simulate communication with the baseboard master controller (BMC). Two computers are not required. However, using two computers facilitates the testing.

Set-Up description:

The test set-up consists of the following: There are two computers, an I²C trace board, an RS-232 interface, an I²C interface board with a parallel port interface, a BMC simulator, the IPMI/IPMB satellite controller unit under test (UUT), a power supply or test fixture to connect to the UUT, and the necessary cables and power supplies to interconnect the various pieces.

Computer #1, running *WinI2CNT* along with the I²C interface board, simulates the BMC issuing commands to the UUT. The UUT then responds to the command that was embedded in the request issued by the BMC. The I²C requires that for each byte transmitted, an acknowledge bit will be transmitted back by the receiver. Computer #1 with the I²C interface board, does not have the capability to provide the acknowledge back to the transmitter. The BMC simulator board provides the acknowledge function of the BMC. There is also the case of the satellite controller issuing an event message. An event message is generated if one of the sensors scanned by the satellite controller exceeds an assertion threshold, or comes back within the deassertion threshold, assuming the proper enables are set. This is the only case where the satellite controller issues the request. The BMC simulator also functions to provide a response to event messages generated by the satellite controller. As a result the entire request/ response process can be tested under realistic conditions. Computer #1, running I2C Trace, allows monitoring of the bus so both the requests and the responses can be observed and compared with the expected responses detailed in the specification.

This test procedure has been organized to follow the same order as the specification, for the most part. This should facilitate using the specification as a verification aid in interpreting the responses to the commands issued. Most of the command requests in the test files are the same as those shown in the request-response figure examples in the specification.

The procedure categories and the associated test files are listed below:

- Application Commands
 - app_cms_test_1.mem
 - prog_id_fw.mem

- Sensor/Event Commands
 - s_e_test_1.mem
 - s_e_test_2.mem
 - s_e_test_3.mem
 - noise_test_a0.mem
 - noise_test_a1.mem
 - sdr_test
 - sdr_rec_00.mem
 - sdr_rec_01.mem
 - sdr_rec_02.mem
 - sdr_rec_03.mem
 - sdr_rec_04.mem
 - sdr_rec_05.mem
 - sdr_rec_06.mem
 - sdr_rec_07.mem
 - sdr_rec_08.mem
 - sdr_rec_09.mem

- Firmware Update Commands
 - fw_test.mem (do not use yet)

- OEM Commands
 - cont_test_1.mem
 - dev_test_1.mem
 - led_test_1.mem
 - prog_id_fw.mem
 - prog_sn_ab.mem

The folders containing the test files are organized in a similar way as shown below.

- test_files
 - a_apps (Application Commands)
 - b_sensor_event (Sensor/Event Commands)
 - c_firmware (Firmware Update Commands)
 - d_oem (OEM Commands)

Procedures

1. Set-up

Configure the test equipment and UUT as shown in **Figure 1**. The tests described here assume the BMC Simulator is an 89C66x device loaded with 66bmc_im2.hex firmware. Devices with this software are marked B-SIM.

Computer #2 should be running *I2C Trace* from The Boardshop (www.demoboard.com), or similar I2C trace utility. It is assumed computer #1 is running *Win-I2CNT* from The Boardshop. Choose “Device”, then “Universal” from the pull down menus. The files referred to in the following instructions can be loaded from the “File” pull-down menu.

2. UUT initialization

Initialization is no longer required for the satellite controller. The pre-set initialization values for thresholds are as follows:

Lower non-critical = nominal - 5%	(Voltage sensors only)
Lower critical = nominal - 10%	(Voltage sensors only)
Lower non-recoverable = nominal - 15%	(Voltage sensors only)
Upper non-critical = nominal + 5%	
Upper critical = nominal + 10%	
Upper non-recoverable = nominal + 15%	

With a 4.096 V reference, nominal = BBh.

Positive going hysteresis = 04h
Negative going hysteresis = 04h

These can be changed with the Set Sensor Thresholds/Hysteresis commands.

3. Application Commands

Open the file “app_cmd_test_1.mem”. The following commands can be issued and the responses compared with those expected in the specification.

- 1- Get Device ID
- 2- BC Get Device ID
- 3- Cold Reset *
- 4- Warm Reset *
- 5- Get Self Test Results (55h = passed)

* Since these commands reset the processor, a complete response is not possible before the reset takes place.

Prior to the “Device ID” information being programmed into the device, the memory locations will read “FFh”.

To program this location load the file “prog_id_fw.mem”. The following commands can be issued and the responses compared with those expected in the specification.

- 1- Program Device ID
- 2- Get Device ID
- 3- Program F/W Creation Date
- 4- Get F/W Creation Date
- 5- Blank

4. Sensor/Event Commands

4.1 Sensor Data Record (SDR) Tests

Open the file “\sdr_rec_00.mem”. The following commands can be issued and the responses compared with those expected in the specification.

- 1- Get Device SDR Information – Record 00
- 2- Get Device SDR – Record 00, no offset
- 3- Get Device SDR – Record 00, 16 byte offset
- 4- Get Device SDR – Record 00, 32 byte offset
- 5- Get Device SDR Repository

Similar commands for Record 01 through Record 09 are contained in this directory, so all records can be accessed.

4.2 Event Receiver/ Hysteresis test

Open the file “s_e_test_1.mem”. The following commands can be issued and the responses compared with those expected in the specification.

- 1 – Set Event Receiver
- 2 – Get Event Receiver
- 3 – Set Sensor Hysteresis
- 4 – Get Sensor Hysteresis
- 5 – blank

4.3 Thresholds/Event Enable Test

Open the file “s_e_test_2.mem”. The following commands can be issued and the responses compared with those expected in the specification.

- 1 – Set Sensor Thresholds
- 2 – Get Sensor Thresholds
- 3 – Set Sensor Event Enables
- 4 – Get Sensor Event Enables
- 5 – Get Sensor Reading

4.4 Event Status Test

Open the file “s_e_test_3.mem”. The following commands can be issued and the responses compared with those expected in the specification.

- 1 – Get Sensor Event Status
- 2 – Blank
- 3 – Blank
- 4 – Blank
- 5 – Blank

4.5 Sensor/Mux Test

Open the file “mux_test_1.mem”

This test allows reading of each sensor to verify the sensor and input mux are operational.

- 1 – Get Sensor 0 Reading
- 2 – Get Sensor 1 Reading
- 3 – Get Sensor 2 Reading
- 4 – Get Sensor 3 Reading
- 5 – Get Sensor 4 Reading

Open the file “mux_test_2.mem”

- 1 – Get Sensor 5 Reading
- 2 – Get Sensor 6 Reading
- 3 – Get Sensor 7 Reading
- 4 – Get Sensor 8 Reading (Discrete Sensor – reports 00h)
- 5 – Get Sensor 0 Reading

The sensor numbers correspond to the following inputs:

- Sensor 0 = 1VL (5V)
- Sensor 1 = 3VL (3.3)
- Sensor 2 = 4VL (12V)
- Sensor 3 = 5VL (-12V)
- Sensor 4 = -IS1-IS2
- Sensor 5 = -IS3

Sensor 6 = -IS4
Sensor 7 = IS5
Sensor 8 = Thermal Switch

If a test fixture is available that has the capability to set and adjust each input level to the subassembly, the following test can be run.

4.6 Event Threshold Test

Using the same sensor-mux commands (mux_test_1.mem, mux_test_2.mem) as section 4.4, set each of the inputs to the nominal level (BBh). Then, by monitoring the I2C trace output and adjusting the inputs to each sensor, the threshold that generate events are easily verified. The difference between the de-assertion threshold reported and the actual reading (this information is contained in the response), is due to the hysteresis settings. If the hysteresis is set to '0' via the Set Sensor Hysteresis command, there is still an internal hysteresis of '1', since the same reading cannot generate an assertion event and a de-assertion event at the same time.

The detail file for the individual sensors (for example file "sens_0.mem") provides a good way to monitor the event status of the sensor and change settings as required.

Note: If you change settings in the command request in the Win-I2CNT software, the checksum must be re-calculated and changed. Otherwise, a checksum error (error code 'CCh') will be reported in the response.

- 1 – Set Sensor 0 Hysteresis
- 2 – Set Sensor 0 Thresholds
- 3 – Set Sensor 0 Enables
- 4 – Get Sensor 0 Event Status
- 5 – Get Sensor 0 Reading

4.7 Analog noise performance test

Open the file "test_proc\sensor_test\misc_tests\noise_test_a1.mem". In the Sequencer box, hit "single" The following commands are contained in this file.

- 1- Set Sensor 0 Hysteresis
- 2- Set Sensor 0 Thresholds
- 3- Set Sensor 0 Event Enables
- 4- Get Sensor 0 Event Status
- 5- Get Sensor 0 Reading

This test gives an indication of the magnitude of the peak noise present at the input of the a/d converter. This test requires that the input can be finely adjusted to give the nominal sensor output reading (BBh) from the A/D converter using the "get sensor reading" (#5) command. If the thresholds are set very close to this level and if the hysteresis values are

very small, the event generation capabilities provide a good way to detect noise spikes. In the file "noise_test_a1.mem", the thresholds and hysteresis are set as follows:

Lower non-critical = B9h
Lower critical = B8h
Lower non-recoverable = B7h
Upper non-critical = BDh
Upper critical = BEh
Upper non-recoverable = BFh

Positive going hysteresis = 01h
Negative going hysteresis = 01h

Adjust the input to give a nominal value of BBh while monitoring the trace output when the UUT receives a "get sensor reading" command. With the thresholds set as above, a noise spike that is +/- 2 lsb's in amplitude, will cause a non-critical event to be generated. A noise spike that is +/- 3 lsb's in amplitude, will cause a critical event to be generated. A noise spike that is +/- 4 lsb's in amplitude, will cause a non-recoverable event to be generated. (1 lsb = 16mV, with Vref = 4.096V)

If the settings above did not result in any event messages, try loading the file: "noise_test_a0.mem". The threshold and hysteresis settings provided in this file are as follows:

Lower non-critical = BAh
Lower critical = B9h
Lower non-recoverable = B8h
Upper non-critical = BCh
Upper critical = BDh
Upper non-recoverable = BEh

Positive going hysteresis = 00h
Negative going hysteresis = 00h

Adjust the input to give a nominal value of BBh while monitoring the trace output when the UUT receives a "get sensor reading" command. With the thresholds set as above, a noise spike that is +/- 1 lsb's in amplitude, will cause a non-critical event to be generated. A noise spike that is +/- 2 lsb's in amplitude, will cause a critical event to be generated. A noise spike that is +/- 3 lsb's in amplitude, will cause a non-recoverable event to be generated. Events will typically be generated during this test, since any variation from the nominal value of BBh will cause an event to be generated. If these events are not very frequent, the system is generating very little noise at the inputs of the UUT's A/D converter.

5. OEM Commands

5.1 LED1 / Power OK test

Open the file “led_test_1.mem”. The Requests generated are listed as follows:

- 1- Set LED1 State (Set to 0x01-“ON”)
- 2- Get LED1 State
- 3- Set LED1 State (Set to 0x00-“OFF”)
- 4- Set LED1 State (Set to 0xFF-“Auto”)
- 5- Get LED1 State

When the LED1 state is set to “auto”, then the LED will reflect the self-test status of the sub-assembly. LED1 asserted (‘0’ for this application), indicates that all monitored power supply outputs are within the set thresholds. An LED1 “ON” or “OFF” command will override the “Power OK” status.

5.2 Cont1 / Inhibit test

Open the file “cont_test_1.mem”. The Requests generated are listed as follows:

- 1- Set Control 1 State (Set to 0x01-“ON”)
- 2- Get Control 1 State
- 3- Set Control 1 State (Set to 0x00-“OFF”)
- 4- Set Control 1 State (Set to 0xFF-“Auto”)
- 5- Get Control 1 State

When the CONT1 state is set to “auto”, then the CONT1 OUTPUT will reflect the state of the inhibit sense input. CONT1 asserted (‘0’ for this application), indicates the state of the inhibit sense input. A CONT1 “ON” or “OFF” command will override the “Inhibit sense” status.

5.3 Device status, serial number, f/w creation date test

Open the file “test_proc\command_test\oem_commands\dev_test_1.mem”. The requests generated are listed as follows:

- 1 - Get Device Status
- 2 – Get Firmware Creation Date
- 3 – Get Serial Number
- 4 – n/a
- 5 – n/a

5.4 Vendor Only OEM Commands (to program device id, s/n, f/w date)

Open the file “prog_id_fw.mem”. The requests generated are listed as follows:

- 1 – Program Device ID
- 2 – Get Device ID
- 3 – Program f/w Creation Date
- 4 – Get F/W creation date
- 5 – n/a

Open the file “prog_sn_ab.mem”. The requests generated are listed as follows:

- 1 – Program S/N – part A
- 2 – Get Serial Number
- 3 – Program S/N – part B
- 4 – Get Serial Number
- 5 – Lock Program Data

The serial number programming was broken into two separate commands to allow it's programming using the Win-I2CNT software. Do not send command request #5, Lock Program Data, until all other information is programmed in