PORSCHE

BBridge: Software Requirements and Test Cases

PEG-GX

Porsche Engineering

driving technologies

Version: 1.1
Status: Released

Version History

Date	Version	Change	Author / Role
31.03.2016	0.1	created	Janni Dürr
31.03.2016	1.0	Reviewed	J. Dürr / U. Schlieben
04.04.2016	1.1	Added MISRA C section, improved Test Cases section	A. Perico

Open points in this document

Open Point	Section	Tbd by

Review

Role	Name	Department	Date	Version	Signature

Table of contents

1 Introduction

1.1 Purpose of this document

The BBridge software is managing an over the air communication network between µ-controller host systems via Bluetooth LE.
It provides an software application layer for the µ-Controller and is managing the network handling mechanism.

This document describes the BBridge software requirements, test cases which will be used as reference for the black box testing and unit testing of the software.

1.2 Referenced Documents

Document	Description // SVN URL
BBridge_Architecture_Interfaces_Protocol.pdf	Documentation of the BBridge Architecture, software and hardware interfaces, communication protocol and API. PES document

Table 1 – Referenced Documents

1.3 Definitions, Acronyms and Abbreviations

Definition, Acronym, Abbreviaton	Desciption
Panther	µController host system that uses the BBridge software for networking
BBridge	Application that manages the networking over Bluetooth LE
Bluetooth LE	Bluetooth Low Energy , low range wireless personal area network technology
CC2540	µController subsystem for Bluetooth LE
PES	Porsche Engineering Services GmbH
TI	Texas Instruments
SPI	Serial Peripheral Interface
SPI Ready Line	Hardware line between µC host and subsystem
ACK	Acknowledge

Table 2: Definitions, Acronyms and Abbreviations

2 Software Requirements

The BBridge is configurable to operate in three different modes: IDLE, SCANNER and CONNECTABLE. In each mode, specific functionalities shall or shall not be available as described below.

2.1 IDLE Mode

In Idle mode:

- the WakeUP line is set to LOW.
- BBridge is not broadcasting.
- BBridge is not scanning for connectable devices.
- BBridge shall accept SPI commands to read and write specific configurations

2.1.1 List of Requirements by ID

ID	Description
BB171	The BBridge application shall set the WakeUP line to LOW.
BB173	It shall be possible to read a byte value from the NVM.
BB174	It shall be possible to store a byte value in the NVM.
BB107	It shall be possible to change the operating mode to Scanner over the SPI interface.
BB159	It shall be possible to change the operating mode to Connectable over the SPI interface.
BB207	It shall be possible to read the current operating mode over the SPI interface.
BB210	It shall be possible to read the "Broadcast message" over the SPI interface.
BB211	It shall be possible to set the "Broadcast message" over the SPI interface.
BB212	It shall be possible to read the "Filter message" over the SPI interface.
BB213	It shall be possible to set the "Filter message" over the SPI interface.
BB214	It shall be possible to read the "Advertisement interval" over the SPI interface.
BB215	It shall be possible to set the "Advertisement Interval" over the SPI interface.

Table 3: List of Requirements for IDLE Mode

2.2 SCANNER Mode

In this mode the BBridge application scans for BBridge CONNECTABLE devices within range and tries to validate a connection with up to three devices. As long as there is a validated connection active, the BBridge shall set the WakeUp Line to HIGH. if no BBridge is found, the WakeUP line is set to LOW.

2.2.1 List of Requirements by ID

ID	Description
BB170	In Scanner mode, the BBridge application shall set the WakeUP line to LOW when there is no validated connection active.
BB169	In Scanner mode, the BBridge application shall set the WakeUP line to HIGH when there is at least one validated connection active
BB115	The BBridge application shall be able to scan the area filtering for BBridge devices based on the Filter value.
BB116	The BBridge application shall be able to establish up to 3 validated connections with filtered BBridge devices.
BB118	It shall be possible to read the "Filter message" over the SPI interface.
BB219	It shall be possible to set the "Filter message" over the SPI interface.
BB119	It shall be possible to disconnect from a specific validated connection over the SPI interface.
BB160	When connected, it shall be possible to send data to the BBridge over the SPI interface which is then forwarded to a specific validated connection over Bluetooth .
BB161	When the BBridge receives data from a validated connection over Bluetooth, it shall be forwarded to the Panther device over the SPI interface.
BB131	It shall be possible to exit the "Scanner mode" over the SPI interface.
BB208	It shall be possible to read the current operating mode over the SPI interface.
BB176	It shall be possible to read the RSSI value from a specific validated connection
BB177	It shall be possible to read the Bluetooth Address of a specific validated connection
BB182	It shall be possible to read the broadcasted message from a specific validated connection
BB181	It shall be possible to read the possible connection states (validated or not)

Table 4: List of Requirements for SCANNER Mode

2.3 CONNECTABLE Mode

In this operating mode the BBridge device broadcasts its broadcast message. This mode waits for a BBridge scanner device to initiate a Bluetooth connection. The WakeUP line is set to HIGH as long as there is a validated connection active.

2.3.1 List of Requirements by ID

ID	Description
BB249	The BBridge application shall be able ablethe time
BB250	BBridge shall notify Panther when a validated connection has been lost.
BB251	In Connectable mode, the BBridge application shall set the WakeUP line to LOW when the validated connection is lost.
BB252	In Connectable mode, the BBridge application shall set the WakeUP line to HIGH when a validated connection is established.
BB253	BBridge shall notify Panther with an event message when a validated connection has been established.
BB254	In Connectable mode, when the BBridge application has a validated connection, it shall stop broadcasting the Broadcast message.
BB255	It shall be possible to read the "Broadcast message" over the SPI interface.
BB256	It shall be possible to set the broadcast message over the SPI interface
BB257	It shall be possible to disconnect from the connected device over the SPI interface
BB258	When connected, it shall be possible to send data to the BBridge over the SPI interface which is then forwarded to the validated connection over Bluetooth.
BB259	Once the connection is validated, the BBridge shall forward the received data from the validated connection to the Panther device over SPI.
BB261	It shall be possible to exit the "Connectable mode" over the SPI interface.
BB262	It shall be possible to read the current operating mode over the SPI interface.
BB263	It shall be possible to read the Bluetooth Address of a validated connection
BB264	It shall be possible to read the RSSI value from a validated connection

Table 5: List of Requirements for CONNECTABLE Mode

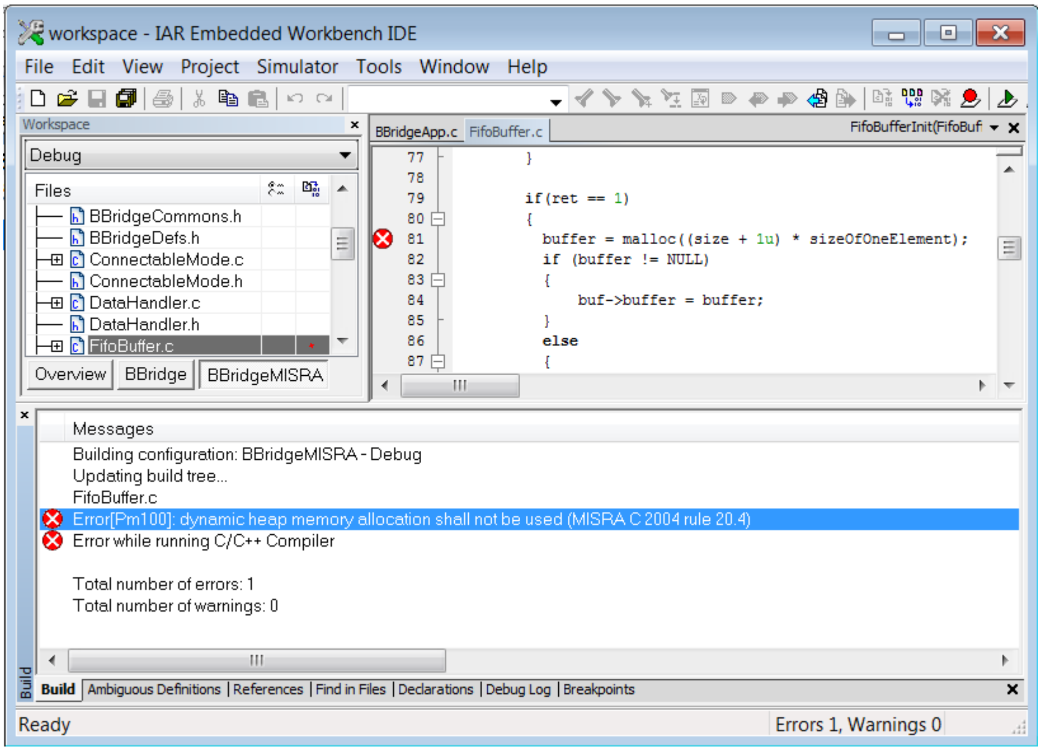
2.4 MISRA C: 2004

The BBridge modules are implemented following the MISRA C: 2004 guidelines. A dedicated “build” has been added to the project workspace so the hardware-independent BBridge modules can be easily checked against the guidelines.

2.4.1 FifoBuffer exception

The FifoBuffer is a module that allows to create and reuse this buffer implementation across different Porsche Engineering projects. The only point it is not fully compliant with MISRA C: 2004 is shown in Figure

1. The dynamic memory allocation is used only once during initialization of the modules and will not result in unpredictable behavior since no “Free” calls are issued in any place of the system.


workspace - IAR Embedded Workbench IDE
File Edit View Project Simulator Tools Window Help
Workspace
Debug
Files
BBridgeCommons.h
BBridgeDefs.h
ConnectableMode.c
ConnectableMode.h
DataHandler.c
DataHandler.h
FifoBuffer.c
Overview BBridge BBridgeMISRA
BBridgeApp.c FifoBuffer.c
FifoBufferInit(FifoBuf
Build Ambiguous Definitions References Find in Files Declarations Debug Log Breakpoints
Ready
Errors 1, Warnings 0

```
        }

        if(ret == 1)
        {
          buffer = malloc((size + 1u) * sizeOfOneElement);
          if (buffer != NULL)
          {
              buf->buffer = buffer;
          }
          else
          {
```

```
Messages
Building configuration: BBridgeMISRA - Debug
Updating build tree...
FifoBuffer.c
Error[Pm100]: dynamic heap memory allocation shall not be used (MISRA C 2004 rule 20.4)
Error while running C/C++ Compiler

Total number of errors: 1
Total number of warnings: 0
```

Figure 1: Checking BBridge modules against MISRA C (2004) guidelines

3 Test Cases

In order to perform black box testing of the software against the requirements described in section 2, test cases for each requirement are defined. In addition, unit tests are described.

Black box testing are executed on a Porsche Engineering test hardware and within a Porsche Engineering testing environment. Unit Tests are performed within a Porsche Engineering testing environment.

3.1 List of Black Box Test Cases and Description.

Req. ID	Test Case ID	Test prerequisite	Expected Test Result
BB171	TC001	1) BBridge is in IDLE Mode	WakeUP Line is LOW
BB173	TC043	1) BBridge is in IDLE Mode 2) Panther sends BBCMD_GET_NVM_BYTE command with the index as parameter.	Panther receives Acknowledge to the command BBCMD_GET_NVM_BYTE containing the Byte value.
BB174	TC044	1) BBridge is in IDLE Mode 2) Panther sends BBCMD_SET_NVM_BYTE command with the index 4 and data byte 0x55 as parameter. 3) read NVM Byte at index 4 (TC043)	Panther receives Acknowledge to the command BBCMD_SET_NVM_BYTE. TC0043 returns 0x55.
BB107	TC002	1) BBridge is in IDLE Mode 2) Panther sends BBCMD_SET_OP_MODE command, with parameter BB_OP_SCANNER.	Panther receives Acknowledge to the BBCMD_SET_OP_MODE command.
BB159	TC003	1) BBridge is in IDLE Mode 2) Panther sends BBCMD_SET_OP_MODE command, with parameter BB_OP_CONNECTABLE.	Panther receives Acknowledge to the BBCMD_SET_OP_MODE command.
BB207	TC004	1) BBridge is in IDLE Mode 2) Panther sends BBCMD_GET_OP_MODE command.	Panther receives Acknowledge to the command BBCMD_GET_OP_MODE containing the parameter BB_OP_IDLE.
BB210	TC005	1) BBridge is in IDLE Mode 2) Panther sends BBCMD_GET_BROADCAST_MSG command.	Panther receives Acknowledge to the command BBCMD_GET_BROADCAST_MSG containing the message.
BB211	TC006	1) BBridge is in IDLE Mode 2) Broadcast message is not equal "TestMessage123" 3) Panther sends BBCMD_SET_BROADCAST_MSG command with parameter "TestMessage123". 4) read Broadcast message (TC005)	Panther receives Acknowledge to the command BBCMD_SET_BROADCAST_MSG. TC005 returns "TestMessage123"
BB212	TC007	1) BBridge is in IDLE Mode 2) Panther sends BBCMD_GET_FILTER_MSG command.	Panther receives Acknowledge to the command BBCMD_GET_FILTER_MSG containing the message.

BB213	TC008	1) BBridge is in IDLE Mode 2) Filter message is not equal "TestMessage123" 3) Panther sends BBCMD_SET_FILTER_MSG command with parameter "TestMessage123". 4) read Filter message (TC007)	Panther receives Acknowledge to the command BBCMD_SET_FILTER_MSG. TC007 returns "TestMessage123"
BB214	TC009	1) BBridge is in IDLE Mode 2) Panther sends BBCMD_GET_ADV_INTERVAL command.	Panther receives Acknowledge to the command BBCMD_GET_ADV_INTERVAL containing the Advertise Interval time configuration.
BB215	TC010	1) BBridge is in IDLE Mode 2) Advertisement time is not equal 1000ms 3) Panther sends BBCMD_SET_ADV_INTERVAL command with parameter 1000ms. 4) read Advertise Interval (TC009)	Panther receives Acknowledge to the command *BCMD_SET_ADV_INTERVAL*. TC009 returns 1000ms
BB170	TC013	1) BBridge is in SCANNER Mode 2) One Validated Connection is active 3) Panther changes operating mode of second BBridge from CONNECTABLE to SCANNER Mode	WakeUp Line changes from HIGH to LOW
BB169	TC014	1) BBridge is in SCANNER Mode 2) No Validated Connection is active 3) Change Operating mode of second BBridge with matching broadcast message to CONNECTABLE mode	WakeUp Line changes from LOW to HIGH
BB115	TC015	1) BBridge is in SCANNER Mode 2) Second BBridge in CONNECTABLE mode is active with broadcast message matching filter message of first BBridge. 3)Third BBridge is active in CONNECTABLE mode with broadcast message not matching filter message of first device. 4) Read Connection States(TC017)	TC017 returns one connection
BB116	TC016	1) BBridge is in SCANNER Mode 2) Three other BBridges in CONNECTABLE mode are active with broadcast message matching filter message of first device. 3)Read Connection States (TC017)	TC017 returns three connections
BB118	TC018	1) BBridge is in SCANNER Mode 2) Panther sends BBCMD_GET_FILTER_MSG command.	Panther receives Acknowledge to the command BBCMD_GET_FILTER_MSG containing the message.
BB219	TC019	1) BBridge is in SCANNER Mode 2) Filter message is not equal "TestMessage123" 3) Panther sends BBCMD_SET_FILTER_MSG command with parameter "TestMessage123". 4) read Filter message (TC018)	Panther receives Acknowledge to the command BBCMD_SET_FILTER_MSG. TC018 returns "TestMessage123"

BB119	TC020	1) BBridge is in SCANNER Mode 2) One Validated Connection is active 3) Panther changes filter message 4) Read number of connections 5) Panther sends BBCMD_DISCONNECT command with device index as parameter 6) Read number of connections	Step 4) returns one connetion Step 6) returns no connection
BB160	TC021	1) BBridge is in SCANNER Mode 2) One Validated connection is active 2) Panther sends BBCMD_SEND_DATA containing the index of the target connection and the data as parameter.	Panther with BBridge in SCANNER mode receives Acknowledge to the command BBCMD_SEND_DATA. BBridge in CONNECTABLE forwards data to Panther.
BB161	TC022	1) BBridge is in SCANNER Mode 2) One Validated connection is active 3) Other BBridge in CONNECTABLE Mode sends data	Panther an SCANNER side reads BB_EVT_RX_MSG with correct data as parameter on SPI.
BB131	TC024	1) BBridge is in SCANNER Mode 2) Panther sends BBCMD_SET_OP_MODE command with parameter BB_OP_IDLE.	Panther receives Acknowledge to the command BBCMD_SET_OP_MODE.
BB208	TC025	1) BBridge is in SCANNER Mode 2) Panther sends BBCMD_GET_OP_MODE command	Panther receives Acknowledge to the command BBCMD_GET_OP_MODE containing the parameter BB_OP_SCANNER.
BB176	TC026	1) BBridge is in SCANNER Mode 2) One Validated connection is active 3) Panther sends BBCMD_GET_CONN_RSSI command with the connection index as parameter.	Panther receives Acknowledge to the command BBCMD_GET_CONN_RSSI containing RSSI of the connected device.
BB177	TC027	1) BBridge is in SCANNER Mode 2) One Validated connection is active 3) Panther sends BBCMD_GET_CONN_ADDR command with the connection index as parameter.	Panther receives Acknowledge to the command BBCMD_GET_CONN_ADDR containing the bluetooth address of the connected device.
BB182	TC028	1) BBridge is in SCANNER Mode 2) One Validated connection is active 3) Panther sends BBCMD_GET_CONN_BRM command with the connection index as parameter.	Panther receives Acknowledge to the command BBCMD_GET_CONN_BRM containing the message which was broadcasted by the connected device.
BB181	TC017	1) BBridge is in SCANNER Mode 2) Panther sends BBCMD_GET_CONN_STATES command.	Panther receives Acknowledge to the command BBCMD_GET_CONN_STATES with the states of the possible connections
BB249	TC029	1) BBridge is in IDLE Mode 2) Two other BBridges are in SCANNER mode with matching filter messages. 3) Panther sets operating mode of first BBridge to CONNECTABLE Mode	WakeUp Line of CONNECTABLE BBridge and one of the SCANNER BBridges changes from LOW to HIGH. WakeUp Line of third BBridge in SCANNER mode stays LOW.

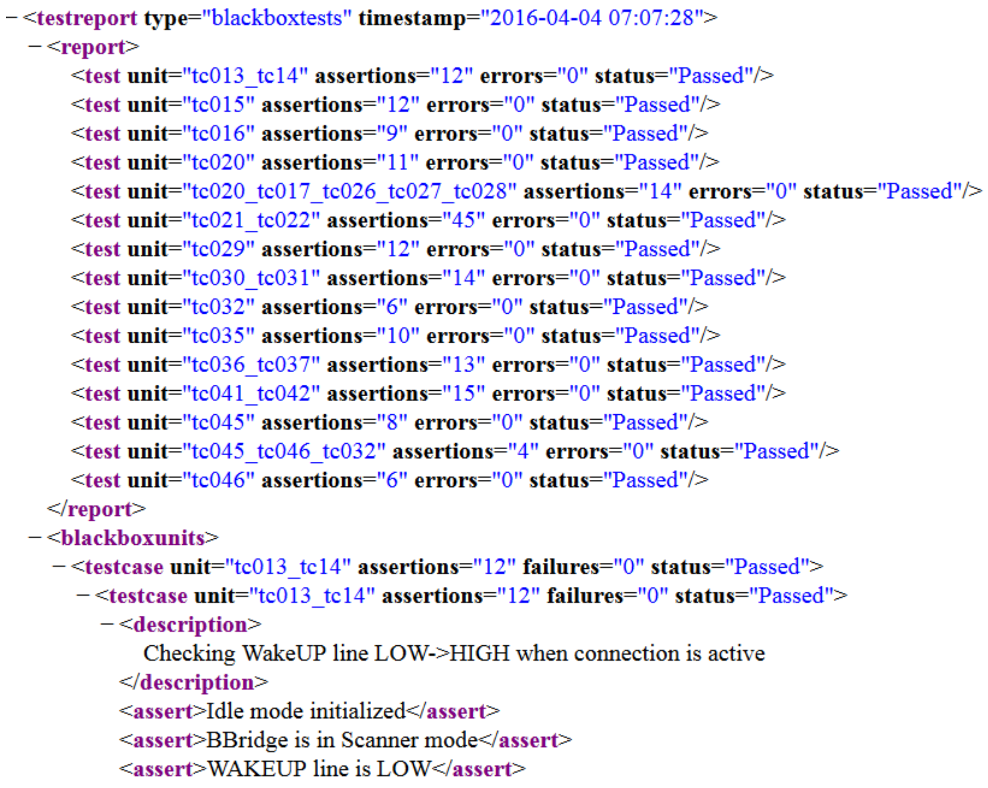
BB250	TC045	1) BBridge is in CONNECTABLE Mode 2) A Validated Connection is active 3) Turn off power supply of second BBridge device	Panther receives a Connection Lost Event from CONNECTABLE Device
BB251	TC030	1) BBridge is in CONNECTABLE Mode 2) A Validated Connection is active 3) Panther changes operating mode of second BBridge from SCANNER Mode to IDLE Mode	WakeUp Line changes from HIGH to LOW
BB252	TC031	1) BBridge is in CONNECTABLE Mode 2) No Validated Connection is active 3) Panther changes operating mode of second BBridge with matching filter message to SCANNER Mode	WakeUp Line changes from LOW to HIGH
BB253	TC046	1) BBridge is in CONNECTABLE Mode 2) No Validated Connection is active 3) Panther changes operating mode of second BBridge with matching filter message to SCANNER Mode	Panther receives a Connection Established Event from CONNECTABLE Device
BB254	TC032	1) BBridge is in CONNECTABLE Mode 2) No Validated Connection is active 3) Panther changes operating mode of second BBridge with matching filter message to SCANNER Mode	WakeUp Line changes from LOW to HIGH. Bluetooth LE Sniffing Tool shows that the CONNECTABLE device has stopped broadcasting.
BB255	TC033	1) BBridge is in CONNECTABLE Mode 2) Panther sends BBCMD_GET_BROADCAST_MSG command.	Panther receives Acknowledge to the command BBCMD_GET_BROADCAST_MSG containing the message.
BB256	TC034	1) BBridge is in CONNECTABLE Mode 2) Broadcast message is not equal "TestMessage123" 3) Panther sends BBCMD_SET_BROADCAST_MSG command with parameter "TestMessage123". 4) read Broadcast message (TC033)	Panther receives Acknowledge to the command BBCMD_SET_BROADCAST_MSG. TC033 returns "TestMessage123"
BB257	TC035	1) BBridge is in CONNECTABLE Mode 2) A Validated Connection is active 3) Panther sends BBCMD_DISCONNECT with connection index 0 as parameter.	Panther receives Acknowledge to the command BBCMD_DISCONNECT. WakeUp Line changes from HIGH to LOW
BB258	TC036	1) BBridge is in CONNECTABLE Mode 2) A Validated connection is active 2) Panther sends BBCMD_SEND_DATA containing 0 as index of the target connection and the data as parameter.	Panther with BBridge in CONNECTABLE mode receives Acknowledge to the command BBCMD_SEND_DATA. Panther on SCANNER side receives data on SPI.
BB259	TC037	1) BBridge is in CONNECTABLE Mode 2) A Validated connection is active 3) Other BBridge in SCANNER Mode sends data	Panther on CONNECTABLE side reads BB_EVT_RX_MSG with correct data as parameter on SPI.

BB261	TC039	1) BBridge is in CONNECTABLE Mode 2) Panther sends BBCMD_SET_OP_MODE command with parameter BB_OP_IDLE.	Panther receives Acknowledge to the command BBCMD_SET_OP_MODE.
BB262	TC040	1) BBridge is in CONNECTABLE Mode 2) Panther sends BBCMD_GET_OP_MODE command	Panther receives Acknowledge to the command BBCMD_GET_OP_MODE containing the parameter BB_OP_SCANNER.
BB263	TC041	1) BBridge is in CONNECTABLE Mode 2) A Validated connection is active 3) Panther sends BBCMD_GET_CONN_ADDR command with the connection index as parameter.	Panther receives Acknowledge to the command BBCMD_GET_CONN_ADDR containing the bluetooth address of the connected device.
BB264	TC042	1) BBridge is in CONNECTABLE Mode 2) A Validated connection is active 3) Panther sends BBCMD_GET_CONN_RSSI command with the connection index as parameter.	Panther receives Acknowledge to the command BBCMD_GET_CONN_RSSI containing RSSI of the connected device.

Table 6: Software technical requirements and respective test cases

3.2 Black Box Testing Results

The blackbox tests are implemented following the test cases described in section 3.1. The results provide reference for the current system status regarding its behavioral correctness and can be used to check implementation fulfillment. Testing results are provided in XML-format within the file “Tests/Results_Blackbox_Testing.xml”.



```
– <testreport type="blackboxtests" timestamp="2016-04-04 07:07:28">
  – <report>
      <test unit="tc013_tc14" assertions="12" errors="0" status="Passed"/>
      <test unit="tc015" assertions="12" errors="0" status="Passed"/>
      <test unit="tc016" assertions="9" errors="0" status="Passed"/>
      <test unit="tc020" assertions="11" errors="0" status="Passed"/>
      <test unit="tc020_tc017_tc026_tc027_tc028" assertions="14" errors="0" status="Passed"/>
      <test unit="tc021_tc022" assertions="45" errors="0" status="Passed"/>
      <test unit="tc029" assertions="12" errors="0" status="Passed"/>
      <test unit="tc030_tc031" assertions="14" errors="0" status="Passed"/>
      <test unit="tc032" assertions="6" errors="0" status="Passed"/>
      <test unit="tc035" assertions="10" errors="0" status="Passed"/>
      <test unit="tc036_tc037" assertions="13" errors="0" status="Passed"/>
      <test unit="tc041_tc042" assertions="15" errors="0" status="Passed"/>
      <test unit="tc045" assertions="8" errors="0" status="Passed"/>
      <test unit="tc045_tc046_tc032" assertions="4" errors="0" status="Passed"/>
      <test unit="tc046" assertions="6" errors="0" status="Passed"/>
    </report>
  – <blackboxunits>
    – <testcase unit="tc013_tc14" assertions="12" failures="0" status="Passed">
      – <testcase unit="tc013_tc14" assertions="12" failures="0" status="Passed">
        – <description>
            Checking WakeUP line LOW->HIGH when connection is active
          </description>
          <assert>Idle mode initialized</assert>
          <assert>BBridge is in Scanner mode</assert>
          <assert>WAKEUP line is LOW</assert>
```

Table 7: Example for blackbox test results.

3.3 Unit Tests

Every BBridge module implemented for the project has its own group of tests to ensure that the implementation meets the design and the needs of the module.

The modules* are unit-tested in two steps:

1- Unit tests of the public functions:

   a. These tests stimulate the “function under test” with parameters or internal variable changes to behave and output its expected return values.

   b. If needed, hardware-dependent functions are also stimulated to generate expected results. (see section 3.3.1)

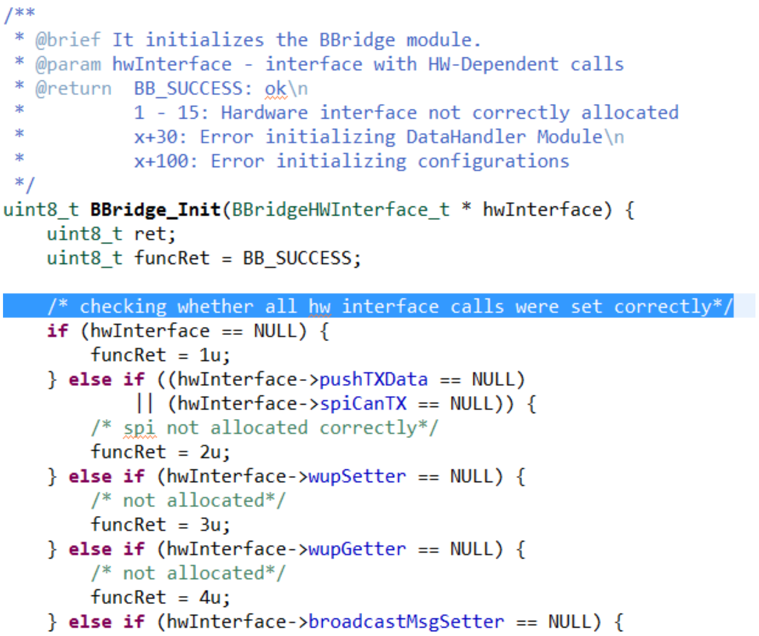
2- Unit tests for the main state machine “work function”:

   a. In these tests the behavior of the “work function” (module state machine) of each module is tested against its expected behavior by calling the work function a specific number of times and checking the current state of the state machine.

   b. A special function, available only for the test environment, is used access and verify the state of the module state machine.

3.3.1 Hardware abstraction

The modules were designed to be hardware-independent so the unit tests could be done offline - without accessing the hardware functions. Every hardware-dependent function is given to the modules over a hardware interface structure with function pointers for the hardware-dependent functions.

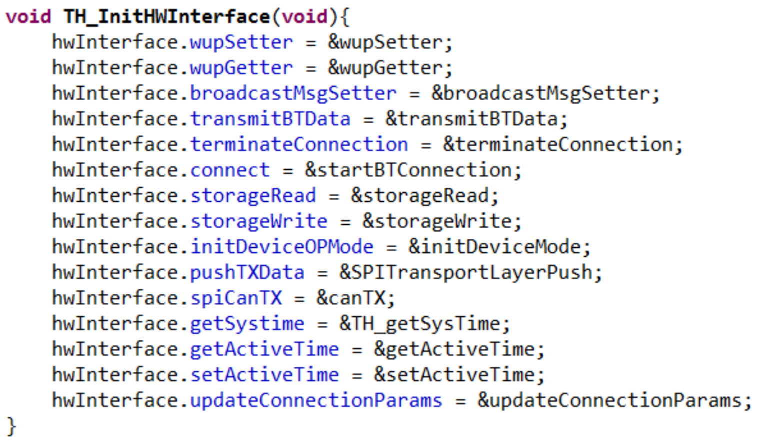


```
/**
 * @brief It initializes the BBridge module.
 * @param hwInterface - interface with HW-Dependent calls
 * @return  BB_SUCCESS: ok\n
 *          1 - 15: Hardware interface not correctly allocated
 *          x+30: Error initializing DataHandler Module\n
 *          x+100: Error initializing configurations
 */
uint8_t BBridge_Init(BBridgeHWInterface_t * hwInterface) {
    uint8_t ret;
    uint8_t funcRet = BB_SUCCESS;

    /* checking whether all hw interface calls were set correctly*/
    if (hwInterface == NULL) {
        funcRet = 1u;
    } else if ((hwInterface->pushTXData == NULL)
            || (hwInterface->spiCanTX == NULL)) {
        /* spi not allocated correctly*/
        funcRet = 2u;
    } else if (hwInterface->wupSetter == NULL) {
        /* not allocated*/
        funcRet = 3u;
    } else if (hwInterface->wupGetter == NULL) {
        /* not allocated*/
        funcRet = 4u;
    } else if (hwInterface->broadcastMsgSetter == NULL) {
```

Figure 2: Hardware Interface structure given to a BBridge module

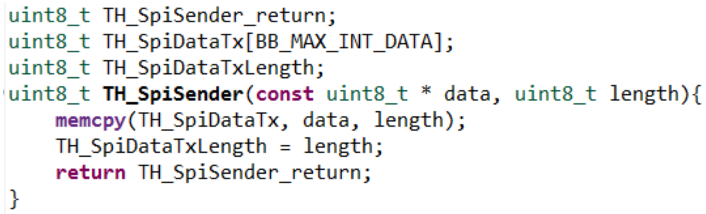
In order to simulate the hardware functionality offline, dedicated test functions are implemented according to the hardware-function behavior. These test functions simulate the output of the online behavior.



```
void TH_InitHWInterface(void){
    hwInterface.wupSetter = &wupSetter;
    hwInterface.wupGetter = &wupGetter;
    hwInterface.broadcastMsgSetter = &broadcastMsgSetter;
    hwInterface.transmitBTData = &transmitBTData;
    hwInterface.terminateConnection = &terminateConnection;
    hwInterface.connect = &startBTConnection;
    hwInterface.storageRead = &storageRead;
    hwInterface.storageWrite = &storageWrite;
    hwInterface.initDeviceOPMode = &initDeviceMode;
    hwInterface.pushTXData = &SPITransportLayerPush;
    hwInterface.spiCanTX = &canTX;
    hwInterface.getSystime = &TH_getSysTime;
    hwInterface.getActiveTime = &getActiveTime;
    hwInterface.setActiveTime = &setActiveTime;
    hwInterface.updateConnectionParams = &updateConnectionParams;
}
```

Figure 3: Creating an offline hardware interface for the test environment

For instance, in the test-environment, the Test Helper (TH) function **TH_SpiSender** is used to simulate the SPI transmission behavior. This function is called when the BBridge modules needs to transmit data over SPI and provides mechanisms (over its global variables) to stimulate its real behavior by influencing its return values. Furthermore it is possible to check the values that were given by the module as parameters to be transmitted.



```
uint8_t TH_SpiSender_return;
uint8_t TH_SpiDataTx[BB_MAX_INT_DATA];
uint8_t TH_SpiDataTxLength;
uint8_t TH_SpiSender(const uint8_t * data, uint8_t length){
    memcpy(TH_SpiDataTx, data, length);
    TH_SpiDataTxLength = length;
    return TH_SpiSender_return;
}
```

Figure 4: Test function structure to simulate a hardware behavior

3.4 Unit Testing Results

Testing results of the unit tests described in section 3.3 are provided in XML-format within the file “Tests/Results_Unit_Testing.xml”. Every “BBridge module” has its own group of tests as shown in Table 8.

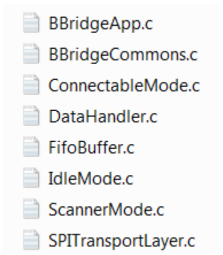
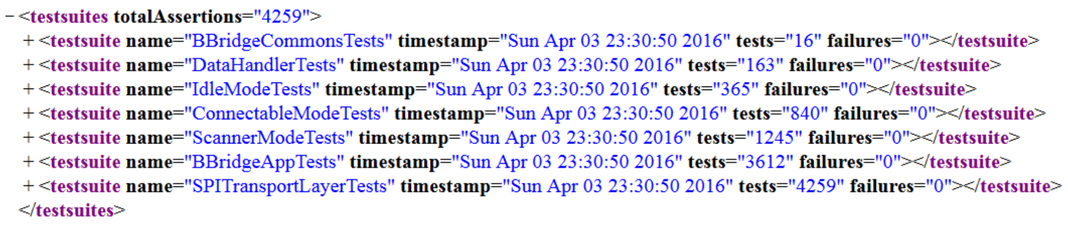
BBridge Modules	Unit test report
 BBridgeApp.c BBridgeCommons.c ConnectableMode.c DataHandler.c FifoBuffer.c IdleMode.c ScannerMode.c SPITransportLayer.c	 - <testsuites totalAssertions="4259"> + <testsuite name="BBridgeCommonsTests" timestamp="Sun Apr 03 23:30:50 2016" tests="16" failures="0"></testsuite> + <testsuite name="DataHandlerTests" timestamp="Sun Apr 03 23:30:50 2016" tests="163" failures="0"></testsuite> + <testsuite name="IdleModeTests" timestamp="Sun Apr 03 23:30:50 2016" tests="365" failures="0"></testsuite> + <testsuite name="ConnectableModeTests" timestamp="Sun Apr 03 23:30:50 2016" tests="840" failures="0"></testsuite> + <testsuite name="ScannerModeTests" timestamp="Sun Apr 03 23:30:50 2016" tests="1245" failures="0"></testsuite> + <testsuite name="BBridgeAppTests" timestamp="Sun Apr 03 23:30:50 2016" tests="3612" failures="0"></testsuite> + <testsuite name="SPITransportLayerTests" timestamp="Sun Apr 03 23:30:50 2016" tests="4259" failures="0"></testsuite> </testsuites>

Table 8: Example of test report with corresponding BBridge module

Note: FifoBuffer is a module already tested in production previous projects and the results are not compatible with this environment for test results.
Note II: Hardware-dependent modules with direct calls from the BLE Stack and TI implementation is not included in the scope of this test report.