



# DESIGN AND IMPLEMENTATION OF THE BOOTLOADER

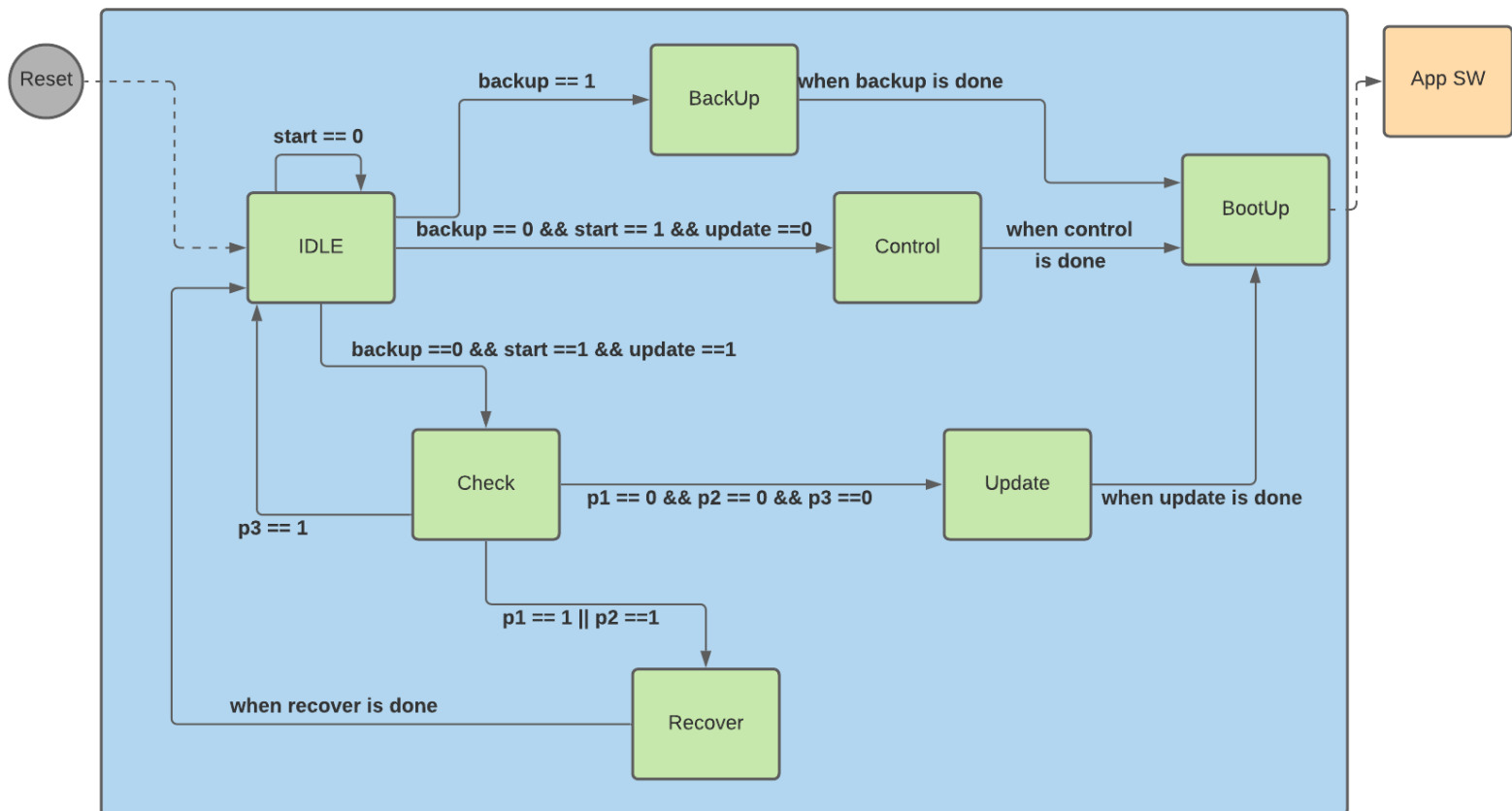
EFE ÖZTABAN  
KAYA GÖKALP

## Requirements List for the Bootloader

1. Bootloader shall be able to jump to the application software.
2. Bootloader shall be able to update the application software with the received image when there is a firmware update.
3. Bootloader shall be able to communicate through RMAP.
4. Bootloader shall have states as IDLE, Bootup, Check, Recover, BackUp and Update which can be switched between.
5. Bootloader status shall be visible to the master (in flight model as CPM, in simulation model EGSE) using status register.
6. The current state of the bootloader shall be able to be controlled by the master (in flight model as CPM, in simulation model EGSE) using the control registers.
7. Bootloader shall be able to read and write to the non-volatile memory which application software stands in.
8. Bootloader shall have a check system on the receiving image for verifying communication channel (integrity of the receiving image).
9. Bootloader shall have a check system on the incoming data correctness (checking if it is valid firmware).
10. Bootloader shall have a check system regarding the capability of writing and reading the non-volatile memory.
11. Bootloader shall know the portion in the non-volatile memory which belongs to the application software (address of the start and end point).
12. Bootloader shall have a robustness sequence for a crash during firmware update which should recover the old application software.
13. Non-volatile memory used by bootloader shall have enough space for three application images. Two of these memory regions shall be changeable with update, whereas the third one shall be unchanged.
14. Bootloader shall overwrite the firmware update to the oldest changeable application image and it should jump to the newest changeable application image.
15. The third application image shall be used as the last backup when the master (in flight model as CPM, in simulation model EGSE) changes the related control register.
16. Bootloader shall have a check system for the software images in the non-volatile memory while booting without updating. This system shall choose which software image to boot.

## State Diagram

### BOOTLOADER



P1 = Incoming data correctness problem (means receiving data is not a SW image)

P2 = Problem with integrity of the receiving image (CRC check problem in the receiving data)

P3 = Write/Read error on the non-volatile memory

## Registers

Control	Status
start	P1
update	P2
bootup_backup	P3
	state_info
	success

System memory (in non-volatile memory)
select_image (old/new/back_up)
write_ability

## Memory

PROM
Bootloader
BackUp App SW Image
EEPROM
App SW Image 1
App SW Image 2

Unchangeable memory region
Changeable memory region

# Implementation details of the Bootloader

1. Bootloader shall be able to jump to the application software.
  - a. There will be three pointers (reset vectors of the software image) which point to the start of the memory locations which are reserved for the three application images. One of these pointers will be chosen by the bootloader according to the `select_image` (this keeps the information about which software image to boot). With the selected pointer, the bootloader will jump to the application software by sending the program counter to the pointed location in the memory.
  - b. `select_image` will be determined before the BootUp state as;
    - i. if there is no update, it is already set.
    - ii. if there is an update, it will be set in Update state.
    - iii. if the last backup image will be used, it will be set in BackUp state.
  - c. Jumping to the application image will be done in BootUp state.
2. Bootloader shall be able to update the application software with the received image when there is a firmware update.
  - a. If there is an update for the application image, master will write this new update to the volatile memory of the AIB by RMAP.
  - b. Master will inform the BL by changing the control register byte called update.
  - c. When writing the update, the master will change the control register byte called start and update.
  - d. BL will be waiting in the IDLE state by default. And it will be constantly checking start and update bits to change the state. After start and update registers are changed, BL will start the check process.
  - e. If there is no problem, BL will go into Update state and will write the update to the non-volatile memory. The oldest copy (the image that is not pointed by `select_image`) of the application image will be replaced by the update. Also `select_image` will be changed to show the location of the newest application image.
  - f. If there is a problem with the update or communication, BL will change the status register byte related to the specific problem (p1, p2, p3). After sending the error message with status registers, BL will go to IDLE state and wait for the master.
3. Bootloader shall be able to communicate through RMAP.
  - a. RMAP communication with two of the SpaceWire nodes (which are master (in flight model as CPM, in simulation model EGSE) and BL) will be initialized in the IDLE state.
  - b. Master will write to control registers by RMAP and read the status registers by RMAP.
  - c. Master will write the update by RMAP.

- d. For the software update, RMAP communication with acknowledgment will be used and after receiving acknowledgement, master will change the control register called start.
- 4. Bootloader shall have states as IDLE, Bootup, Check, Recover, Control, BackUp and Update which can be switched between.
  - a. The states and switching condition for the states is defined in the state diagram.
  - b. State of the BL will be reset to IDLE after it is started. With the control register inputs states will be switched between according to the state diagram.
  - c. While leaving the IDLE state, status registers related to errors called p1, p2 and p3 will be reset. Because if there is an error in the previous iteration, BL needs to try the sequence and do the checks again.
- 5. Bootloader status shall be visible to the master (in flight model as CPM, in simulation model EGSE) using status register.
  - a. With the status register called state\_info, the state which BL is currently on will be shown to the master.
  - b. This register will be reset to the IDLE after reset. For every change of the states, this register will be updated.
- 6. The current state of the bootloader shall be able to be controlled by the master (in flight model as CPM, in simulation model EGSE) using the control registers.
  - a. Master will change the control registers called start, update and bootup\_backup. By looking at the control registers, the state of the BL will be changed according to the state diagram.
- 7. Bootloader shall be able to read and write to the non-volatile memory which application software stands in.
  - a. BL should be able to read the non-volatile memory reserved for instructions in AIB.
  - b. This capability will be checked in the check sequence with write\_ability.
  - c. Before going into BootUp, the selected software image will be checked and will be changed with backup if it is necessary.
- 8. Bootloader shall have a check system on the receiving image for verifying communication channel (integrity of the receiving image).
  - a. After data is received to the volatile memory and BL gets into check state, data will be checked with CRC.
  - b. If there is no problem, the system will do the other checks and go into Update state.

- c. If there is a problem, a related status register called p2 will be set and BL will wait in IDLE state.
- 9. Bootloader shall have a check system on the incoming data correctness (checking if it is valid firmware).
  - a. After data is received to the volatile memory and BL gets into check state, data will be checked to control if it is valid data by looking at the specific parts (bytes) of the data.
  - b. If there is no problem, the system will do the other checks and go into Update state.
  - c. If there is a problem, a related status register called p1 will be set and BL will wait in IDLE state.
- 10. Bootloader shall have a check system regarding the capability of writing and reading the non-volatile memory.
  - a. There will be a small region in the non-volatile memory for that purpose called write\_ability.
  - b. For the check system, a bunch of dummy bytes will be written to that location by the BL. Then BL will read this region and check the correctness. If there is no problem, that means there is no problem with writing and reading capability.
  - c. After the check process, this region called write\_ability will be reset for later checks.
  - d. If there is a problem in this process, the related status register with this problem called p3 will be set and BL will wait in the IDLE state.
- 11. Bootloader shall know the portion in the non-volatile memory which belongs to the application software (address of the start and end point).
  - a. There will be three memory regions in the non-volatile memory reserved for application images (two changeable by update, one unchangeable). The start address of these regions will be saved with three pointers (reset vectors of the software image) for BL. Size of each memory region will be 3 MB.
  - b. The information about the oldest and newest software image will be kept by the select\_image.
  - c. With the select\_image, the appropriate pointer to the memory location will be chosen and boot to that software image.
- 12. Bootloader shall have a robustness sequence for a crash during firmware update which should recover the old application software.
  - a. If a problem is found during the check process, a related status register will be set.
  - b. Received data will not be written into the non-volatile memory and select\_image will be changed the same as the previous state because there won't be a change in the software images in the non-volatile memory.

13. Non-volatile memory used by bootloader shall have enough space for three application images. Two of these memory regions shall be changeable with update, whereas the third one shall be unchanged.
- a. There will be three memory regions in the non-volatile memory reserved for application images (two changeable by updates, one unchangeable). The start address of these regions will be saved with three pointers (reset vectors of the software image) for BL. Size of each memory region will be 3 MB.
  - b. Two of the changeable memory regions will be used in normal mode. When there is an update without an error, the new software image will be overwritten to the oldest software image (check from `select_image`). After that change, `select_image` will be changed accordingly.
  - c. The third unchangeable memory will be written only before the flight. This memory region shall not be changed with BL anytime. This memory region will be used for the last backup. If the related control register called `boot_backup` set by the master, BL will boot to that software image. This software image will be an unchangeable copy of the very initial version of the software image.
14. Bootloader shall overwrite the firmware update to the oldest changeable application image and it should jump to the newest changeable application image.
- a. Two of the changeable memory regions will be used in normal mode. When there is an update without an error, the new software image will be overwritten to the oldest software image (check from `select_image`). After that change, `select_image` will be changed accordingly.
  - b. Oldest version will be kept as a backup. In normal mode the newest version will be used.
15. The third application image shall be used as the last backup when the master (in flight model as CPM, in simulation model EGSE) changes the related control register.
- a. The third unchangeable memory will be written only before the flight. This memory region shall not be changed with BL anytime. This memory region will be used for the last backup. If the related control register called `boot_backup` set by the master, BL will boot to that software image. This software image will be an unchangeable copy of the very initial version of the software image.
16. Bootloader shall have a check system for the software images in the non-volatile memory while booting without updating. This system shall choose which software image to boot.
- a. When booting process starts without an update, BL will go into Control state. In this state BL will do checks on the newest software image in the non-volatile memory.
  - b. If there is no problem is found during the checks, `select_image` will stay the same and the newest software image will be boot.



- C. If a problem is found during the checks, `select_image` will be changed to use the previous version of the software image. Then this older version of the software image will be boot.

# Verification details for the Requirements of the Bootloader

1. Bootloader shall be able to jump to the application software.
  - a. For four modes (boot without update/boot with update/boot with backup/boot with last backup), the system will boot. If application software starts for all four modes, this requirement is verified.
2. Bootloader shall be able to update the application software with the received image when there is a firmware update.
  - a. The update mode (start=1 && update=1) will be activated and a new software will be sent to BL. After BL boots the application, if the application will work as the new version, this requirement will be verified.
3. Bootloader shall be able to communicate through RMAP.
  - a. Status registers will be read by the master. Control registers and software updates will be written by the master. Writing to control registers and reading from the status registers will be performed. Also, a software update will be performed. If there is no problem with these processes, this requirement will be verified.
4. Bootloader shall have states as IDLE, Bootup, Check, Recover, BackUp and Update which can be switched between.
  - a. Status register called state\_info will be read by the master. If the status of the BL is changing with the appropriate inputs (change of the control register) is changing, this requirement will be verified.
5. Bootloader status shall be visible to the master (in flight model as CPM, in simulation model EGSE) using status register.
  - a. Status register called state\_info will be read by the master. If the status can be read and changed correctly, this requirement will be verified.
6. The current state of the bootloader shall be able to be controlled by the master (in flight model as CPM, in simulation model EGSE) using the control registers.
  - a. Related control registers will be set by the master to change to every state. Then the state of the BL will be checked with the status register called state\_info. If every state can be switched correctly, this requirement will be verified.

7. Bootloader shall be able to read and write to the non-volatile memory which application software stands in.
  - a. A software update will be performed and if there is no p3 error after this process, this requirement will be verified.
8. Bootloader shall have a check system on the receiving image for verifying communication channel (integrity of the receiving image).
  - a. Software updates will be performed with correct and erroneous data. After the update with correct and erroneous (p2 check should not give an error) data, p1 will be checked. If for both cases, BL works correctly if;
    - i. p2 = 0 and boot to software application for correct data
    - ii. p2 = 1 and get into IDLE without booting with the erroneous datathis requirement will be verified.
9. Bootloader shall have a check system on the incoming data correctness (checking if it is valid firmware).
  - a. Software updates will be performed with correct and dummy data. After the update with correct and erroneous data, p2 will be checked. If for the both cases, BL works correctly;
    - i. p1 = 0 and boot to software application for correct data
    - ii. p1 = 1 and get into IDLE without booting with the erroneous datathis requirement will be verified.
10. Bootloader shall have a check system regarding the capability of writing and reading the non-volatile memory.
  - a. A software update will be performed and if there is no p3 error and another software update will be performed with a configuration which is not possible to write/read to the non-volatile memory. If there is a p3 error in this process, this requirement will be verified.
11. Bootloader shall know the portion in the non-volatile memory which belongs to the application software (address of the start and end point).
  - a. Three memory regions for software images will be written with different application images. Then booting to different three applications will be performed. If all of them work correctly then this requirement will be verified.
12. Bootloader shall have a robustness sequence for a crash during firmware update which should recover the old application software.
  - a. A software update with erroneous data will be performed. In the next iteration, booting to two changeable software images will be performed. If both applications work correctly like before the update, this requirement will be verified.

13. Non-volatile memory used by bootloader shall have enough space for three application images. Two of these memory regions shall be changeable with update, whereas the third one shall be unchanged.
  - a. Both changeable software images will be updated with the biggest limit for the software images (3 MB). Also, consecutive two updates will be performed with correct data. If there is no problem with these processes, this requirement will be verified.
14. Bootloader shall overwrite the firmware update to the oldest changeable application image and it should jump to the newest changeable application image.
  - a. A software update will be performed. Then for the two changeable software images, the booting process will be performed. For both application starts correctly, and oldest/newest applications change correctly after update, this requirement will be verified.
15. The third application image shall be used as the last backup when the master (in flight model as CPM, in simulation model EGSE) changes the related control register.
  - a. A boot process with last backup (start=1 && boot\_backup=1) will be performed. If the application starts correctly, this requirement will be verified.
16. Bootloader shall have a check system for the software images in the non-volatile memory while booting without updating. This system shall choose which software image to boot.
  - a. A boot process without an update will be performed (start =1 && update = 0 && backup = 0). Then it will be checked if the newest version of the application is started. After that check, the newest version of the software image will be damaged intentionally, and it will be checked if the previous version of the application is started. If no problem occurs during these checks, this requirement will be verified.

## Responsibilities of the Master for Bootloader

**To boot without update:** When the system starts, BL will be waiting in the IDLE state to take commands from the master. To check if the BL is waiting in the IDLE state, master can check the status register bits called **state\_info**. If these bits are “001”, it means BL is at IDLE state. To start the BL, master should set control bits as **start = 1 update = 0 bootup\_backup = 0**. After these bits are changed, master should check the bits of status register called **state\_info**. When these bits change from “001”, it means that BL left the IDLE state. When BL left the IDLE state, master should reset the bits in the control register as **start = 0 update = 0 bootup\_backup = 0**. Then master should control the status register bit called **success**. If status register called **success** is 1, it means BL is finished the process correctly and master can stop checking the status register bit called **success**.

**To boot with update:** When the system starts, BL will be waiting in the IDLE state to take commands from the master. To check if the BL is waiting in the IDLE state, master can check the status register bits called **state\_info**. If these bits are “001”, it means BL is at IDLE state. While BL is waiting the IDLE state, master should write the update in the AIB’s volatile memory. When master write the update correctly (RMAP will send ack and verification for that communication), to start the boot process, master should set control bits as **start = 1 update = 1 bootup\_backup = 0**. After these bits are changed, master should check the bits of status register called **state\_info**. When these bits change from “001”, it means that BL left the IDLE state. When BL left the IDLE state, master should reset the bits in the control register as **start = 0 update = 0 bootup\_backup = 0**. Then master should control the status register bits called **p1, p2, p3** and **success**. If one of these bits indicates problem (p1, p2, p3) are set to 1, it means BL found a problem during boot process and master should act accordingly to the related problem (try again to update, boot normally or boot with backup). If status register called **success** is 1, it means BL is finished the process correctly and master can stop checking the status register bits called **p1, p2, p3** and **success**.

**To boot the last backup:** When the system starts, BL will be waiting in the IDLE state to take commands from the master. To check if the BL is waiting in the IDLE state, master can check the status register bits called **state\_info**. If these bits are “001”, it means BL is at IDLE state. To start the BL, master should set control bits as **start = 1 update = 0 bootup\_backup = 1**. After these bits are changed, master should check the bits of status register called **state\_info**. When these bits change from “001”, it means that BL left the IDLE state. When BL left the IDLE state, master should reset the bits in the control register as **start = 0 update = 0 bootup\_backup = 0**. Then master should control the status register bit called **success**. If status register called **success** is 1, it means BL is finished the process correctly and master can stop checking the status register bit called **success**.

# Detailed Explanation of the Change of Control and Status Register Bits

## Control Register

**start:** This bit will be set by master to start the process of the BL. If there is an update this bit shall be set after the new software is sent to the volatile memory of the AIB and after the update bit is set. If there is no update, it will be set to boot the application. This bit shall be set only if BL is in the IDLE state (this will be known from the state\_info). And when the BL change its states from the IDLE, this bit shall be reset by the master.

**update:** This bit will be set by master to tell the BL that there is an update. For the update this bit shall be set after the new software is sent to the volatile memory of the AIB and before the start bit is set. This bit shall be set only if BL is in the IDLE state (this will be known from the state\_info). And when the BL change its states from the IDLE, this bit shall be reset by the master.

**boot\_backup:** This bit will be set by master to tell the BL that last backup shall be boot. For the backup boot this bit shall be set before the start bit is set. This bit shall be set only if BL is in the IDLE state (this will be known from the state\_info). And when the BL change its states from the IDLE, this bit shall be reset by the master.

## Status Register

**p1:** This bit will be set by BL to tell master that there is a data correctness error in the receiving data. This bit will be set in the Check state after the related checks are done and an error is found. This bit will be reset when BL is leaving the IDLE state.

**p2:** This bit will be set by BL to tell master that there is an integrity error in the receiving data. This bit will be set in the Check state after the related checks are done and an error is found. This bit will be reset when BL is leaving the IDLE state.

**p3:** This bit will be set by BL to tell master that there is a writing/reading capability error to the non-volatile memory of AIB. This bit will be set in the Check state after the related checks are done and an error is found. This bit will be reset when BL is leaving the IDLE state.

**state\_info:** This will be a 3 bits information in the register for 7 different states of the BL. These bits will be set into the appropriate configuration when BL gets into a new state by the BL. These bits will be reset to IDLE state by BL when BL gets into the IDLE state.

001 -> IDLE	101 -> Update
010 -> Control	110 -> Recover
011 -> BootUp	111-> BackUp
100 -> Check	

**success:** This bit will be set by BL to tell master that booting with/without update is successfully done. This bit will be reset by BL in the IDLE state. This bit will be set when BL leaving the BootUp state.

## Internal Memory

This memory will be in the non-volatile memory of the AIB. It will be used for internal process of the BL.

**select\_image:** This will be 2 bits of information. These bits will keep the record of which SW image to use.

01-> First copy of the SW

10-> Second copy of the SW

11-> Last backup

This memory shall be written as 01 before the flight. When there is a successful update if;

select\_image is 01, it will be changed to 10 by BL.

select\_image is 10, it will be changed to 01 by BL.

BL will check the control register bit boot\_backup in the IDLE state. If this bit is set by the master, select\_image will be changed to 11.

**write\_ability:** This will be \* bits of information. These bits will be used to simulate writing and reading into the non-volatile memory. When BL is in IDLE, these bits will be reset to 000\*\*. Then is the Check state, a bunch of dummy bits will be written. Then these bits will be read and control if it is same with the written bits.

# Test Cases for the Verification

## Test Case #1

Description: Normal booting procedure

Aim: to check normal booting procedure, state transitions, non-volatile memory usage.

### Test Steps:

- It is required to have 3 SW image in the non-volatile memory of the AIB before this test.
- System will be reset.
- EGSE will wait until BL is in IDLE. It will be seen from the status register bits called state\_info. When state\_info is "001", EGSE will set control register bits as start=1, update=0 and bootup\_backup = 0.
- When BL leaves the IDLE state, EGSE will reset the control register bits as start=0, update=0 and bootup\_backup = 0.
- Status register bits called state\_info will be monitored during the test. It should be "001", "010" and "011" in order.
- Status register bit called success will be monitored during the test. After booting process is done, this bit should be set to 1.
- If there is no problem in monitoring state\_info and success bits and if the application starts correctly, this test will be finished successfully.

Verified Requirements: #1 (partial), 3 (partial), 4 (partial), 5 (partial), 6 (partial), 11 (partial)

## Test Case #2

Description: Booting procedure with update

Aim: to check booting procedure with update, state transitions, non-volatile memory usage, update capability.

### Test Steps:

- It is required to have 3 SW image in the non-volatile memory of the AIB before this test.
- System will be reset.
- EGSE will wait until BL is in IDLE. It will be seen from the status register bits called state\_info. When state\_info is "001", EGSE will send the update (new application SW) into the non-volatile memory of the AIB. This update should be in a correct form with no errors.
- After sending the update correctly, EGSE will set control register bits as start=1, update=1 and bootup\_backup = 0.
- When BL leaves the IDLE state, EGSE will reset the control register bits as start=0, update=0 and bootup\_backup = 0.
- Status register bits called state\_info will be monitored during the test. It should be "001", "100", "101" and "011" in order.



- Status register bit called success will be monitored during the test. After booting process is done, this bit should be set to 1.
- Status register bits called p1, p2 and p3 will be monitored during the test. These bits should be set to 0 all the time during the test.
- If there is no problem in monitoring state\_info, success, p1, p2 and p3 bits and if the application starts correctly with updated version, this test will be finished successfully.

Verified Requirements: #1 (partial), 2 (partial), 3 (partial), 4 (partial), 5 (partial), 6 (partial), 7 (fully), 8 (partial), 9 (partial), 10 (partial)

### Test Case #3

Description: Booting procedure with last backup

Aim: to check booting procedure with last backup, state transitions, non-volatile memory usage.

#### Test Steps:

- It is required to have 3 SW image in the non-volatile memory of the AIB before this test.
- System will be reset.
- EGSE will wait until BL is in IDLE. It will be seen from the status register bits called state\_info. When state\_info is "001", EGSE will set control register bits as start=1, update=0 and bootup\_backup = 1.
- When BL leaves the IDLE state, EGSE will reset the control register bits as start=0, update=0 and bootup\_backup = 0.
- Status register bits called state\_info will be monitored during the test. It should be "001", "111" and "011" in order.
- Status register bit called success will be monitored during the test. After booting process is done, this bit should be set to 1.
- If there is no problem in monitoring state\_info and success bits and if the application starts correctly with backup version, this test will be finished successfully.

Verified Requirements: #1 (partial), 3 (partial), 4 (partial), 5 (partial), 6 (partial), 11(partial), 15 (fully)

### Test Case #4

Description: Booting procedure with 2<sup>nd</sup> SW image

Aim: to check booting procedure with 2<sup>nd</sup> SW image, state transitions, non-volatile memory usage.

#### Test Steps:

- It is required to have 3 SW image in the non-volatile memory of the AIB before this test.
- The SW image which is pointer by the select\_image status register bit, should be corrupted manually before this test.
- System will be reset.

- EGSE will wait until BL is in IDLE. It will be seen from the status register bits called state\_info. When state\_info is "001", EGSE will set control register bits as start=1, update=0 and bootup\_backup = 0.
- When BL leaves the IDLE state, EGSE will reset the control register bits as start=0, update=0 and bootup\_backup = 0.
- Status register bits called state\_info will be monitored during the test. It should be "001", "010" and "011" in order.
- Status register bit called success will be monitored during the test. After booting process is done, this bit should be set to 1.
- If there is no problem in monitoring state\_info and success bits and if the application starts correctly with 2<sup>nd</sup> version, this test will be finished successfully.

Verified Requirements: #1 (partial), 3 (partial), 4 (partial), 5 (partial), 6 (partial), 11(partial), 16 (fully)

## Test Case #5

Description: Updating SW image with dummy data

Aim: to check incoming data correctness check system, state transitions.

### Test Steps:

- It is required to have 3 SW image in the non-volatile memory of the AIB before this test.
- System will be reset.
- EGSE will wait until BL is in IDLE. It will be seen from the status register bits called state\_info. When state\_info is "001", EGSE will send the update (new application SW) into the non-volatile memory of the AIB. This update should be a dummy data (not a SW update).
- After sending the update, EGSE will set control register bits as start=1, update=1 and bootup\_backup = 0.
- When BL leaves the IDLE state, EGSE will reset the control register bits as start=0, update=0 and bootup\_backup = 0.
- Status register bits called state\_info will be monitored during the test. It should be "001", "100", "110" and "001" in order.
- Status register bit called success will be monitored during the test. This bit should be set to 0 all the time during the test.
- Status register bits called p1, p2 and p3 will be monitored during the test. p1 bit should be set to 1 after the process and p2 and p3 bits should be set to 0 all the time during the test.
- If there is no problem in monitoring state\_info, success, p1, p2 and p3 bits and if the application does not start and BL waits in the IDLE state at the end of the procedure, this test will be finished successfully.

Verified Requirements: #4 (partial), 5 (partial), 9 (partial).

## **Test Case #6**

**Description:** Updating SW image with erroneous data

**Aim:** to check data integrity check system, state transitions.

### **Test Steps:**

- It is required to have 3 SW image in the non-volatile memory of the AIB before this test.
- System will be reset.
- EGSE will wait until BL is in IDLE. It will be seen from the status register bits called state\_info. When state\_info is "001", EGSE will send the update (new application SW) into the non-volatile memory of the AIB. This update should be an erroneous data (corrupted SW update).
- After sending the update, EGSE will set control register bits as start=1, update=1 and bootup\_backup = 0.
- When BL leaves the IDLE state, EGSE will reset the control register bits as start=0, update=0 and bootup\_backup = 0.
- Status register bits called state\_info will be monitored during the test. It should be "001", "100", "110" and "001" in order.
- Status register bit called success will be monitored during the test. This bit should be set to 0 all the time during the test.
- Status register bits called p1, p2 and p3 will be monitored during the test. p2 bit should be set to 1 after the process and p1 and p3 bits should be set to 0 all the time during the test.
- If there is no problem in monitoring state\_info, success, p1, p2 and p3 bits and if the application does not start and BL waits in the IDLE state at the end of the procedure, this test will be finished successfully.

**Verified Requirements:** #4 (partial), 5 (partial), 8 (partial).

## **Test Case #7**

**Description:** Updating SW image with write/read capability error

**Aim:** to check write/read capability check system, state transitions.

### **Test Steps:**

- It is required to have 3 SW image in the non-volatile memory of the AIB before this test.
- It is required to unplug the non-volatile memory which contains the 2 SW image before this test.
- System will be reset.
- EGSE will wait until BL is in IDLE. It will be seen from the status register bits called state\_info. When state\_info is "001", EGSE will send the update (new application SW) into the non-volatile memory of the AIB. This update should be an erroneous data (corrupted SW update).
- After sending the update, EGSE will set control register bits as start=1, update=1 and bootup\_backup = 0.

- When BL leaves the IDLE state, EGSE will reset the control register bits as start=0, update=0 and bootup\_backup = 0.
- Status register bits called state\_info will be monitored during the test. It should be "001", "100", "110" and "001" in order.
- Status register bit called success will be monitored during the test. This bit should be set to 0 all the time during the test.
- Status register bits called p1, p2 and p3 will be monitored during the test. p3 bit should be set to 1 after the process and p1 and p2 bits should be set to 0 all the time during the test.
- If there is no problem in monitoring state\_info, success, p1, p2 and p3 bits and if the application does not start and BL waits in the IDLE state at the end of the procedure, this test will be finished successfully.

Verified Requirements: #4 (partial), 5 (partial), 10 (partial).

## Test Case #8

Description: Updating SW image with error and boot again

Aim: to check the robustness sequence of the BL.

### Test Steps:

- It is required to have 3 SW image in the non-volatile memory of the AIB before this test.
- System will be reset.
- EGSE will wait until BL is in IDLE. It will be seen from the status register bits called state\_info. When state\_info is "001", EGSE will send the update (new application SW) into the non-volatile memory of the AIB. This update should be an erroneous data (corrupted SW update).
- After sending the update, EGSE will set control register bits as start=1, update=1 and bootup\_backup = 0.
- When BL leaves the IDLE state, EGSE will reset the control register bits as start=0, update=0 and bootup\_backup = 0.
- EGSE will wait until BL is in IDLE. It will be seen from the status register bits called state\_info. When state\_info is "001", EGSE will set control register bits as start=1, update=0 and bootup\_backup = 0.
- When BL leaves the IDLE state, EGSE will reset the control register bits as start=0, update=0 and bootup\_backup = 0.
- Status register bit called success will be monitored during the test. After booting process is done, this bit should be set to 1.
- If there is no problem success bit and if the application does not start and BL waits in the IDLE state at the end of the procedure, this test will be finished successfully.

Verified Requirements: #12 (fully).

## **Test Case #9**

**Description:** Updating SW image without error and boot the older version

**Aim:** to check the two copy of SW image switching system

### **Test Steps:**

- It is required to have 3 SW image in the non-volatile memory of the AIB before this test.
- System will be reset.
- EGSE will wait until BL is in IDLE. It will be seen from the status register bits called state\_info. When state\_info is "001", EGSE will send the update (new application SW) into the non-volatile memory of the AIB. This update should be in a correct form with no errors.
- After sending the update correctly, EGSE will set control register bits as start=1, update=1 and bootup\_backup = 0.
- When BL leaves the IDLE state, EGSE will reset the control register bits as start=0, update=0 and bootup\_backup = 0.
- After application starts, system should be stopped. The SW image which is pointer by the select\_image status register bit, should be corrupted manually.
- System will be reset.
- EGSE will wait until BL is in IDLE. It will be seen from the status register bits called state\_info. When state\_info is "001", EGSE will set control register bits as start=1, update=0 and bootup\_backup = 0.
- When BL leaves the IDLE state, EGSE will reset the control register bits as start=0, update=0 and bootup\_backup = 0.
- Status register bit called success will be monitored during the test. After booting process is done, this bit should be set to 1.
- If there is no problem in monitoring state\_info and success bits and if the application starts correctly, this test will be finished successfully.
- If there is no problem in monitoring success bit and if the application starts correctly with older version, this test will be finished successfully.

**Verified Requirements:** #14 (fully).

## **Test Case #10**

**Description:** Testing the SW image size limits

**Aim:** to check the memory size limits.

### **Test Steps:**

- It is required to have 3 SW image in the non-volatile memory of the AIB before this test.
- System will be reset.
- EGSE will wait until BL is in IDLE. It will be seen from the status register bits called state\_info. When state\_info is "001", EGSE will send the update (new application SW) into the non-

volatile memory of the AIB. This update should be in a correct form with no errors and size of 3 MB.

- After sending the update, EGSE will set control register bits as start=1, update=1 and bootup\_backup = 0.
- When BL leaves the IDLE state, EGSE will reset the control register bits as start=0, update=0 and bootup\_backup = 0.
- Status register bit called success will be monitored during the test. After booting process is done, this bit should be set to 1.
- Status register bits called p1, p2 and p3 will be monitored during the test. These bits should be set to 0 all the time during the test.
- The procedures until now should be repeated.
- System will be reset.
- EGSE will wait until BL is in IDLE. It will be seen from the status register bits called state\_info. When state\_info is "001", EGSE will send the update (new application SW) into the non-volatile memory of the AIB. This update should be in a correct form with no errors and size of 3 MB.
- After sending the update, EGSE will set control register bits as start=1, update=1 and bootup\_backup = 0.
- When BL leaves the IDLE state, EGSE will reset the control register bits as start=0, update=0 and bootup\_backup = 0.
- Status register bit called success will be monitored during the test. After booting process is done, this bit should be set to 1.
- If there is no problem in monitoring state\_info, success, p1, p2 and p3 bits and if the application starts correctly with updated versions (for both iteration), this test will be finished successfully.

Verified Requirements: #13 (fully).

# After Implementation

## Current Implementation

Currently we have the following concepts implemented:

1. Bootloader states
2. Bootloader housekeeping data
3. Means of communication between us and the bootloader for testing
4. Ability to jump to different applications

## Bootloader States

Bootloader states and their flow diagrams are implemented. Which means that the bootloader is switching to the desired states with predetermined control register changes with respect to the design document of the BL.

## Bootloader Housekeeping Data

Bootloader housekeeping data is created for debugging purposes. The bootloader sends out the housekeeping data to the tester so that state transitions and select\_image bit changes can be checked by the tester.

## Means of Communication Between Tester and the Bootloader

We have built the bootloader using the 'qnd' sample project that is provided by TJ. Which enables us to send data and read data from SpaceWire protocol. With SpaceWire we are sending some messages and changing the control register to mimic the master and test the BL.

Control registers and select\_image bit is directly sent by SpaceWire protocol. This communication method is created to mimic the master during testing and debugging. Two types of command is sent by SpaceWire:

1. The messages start with "00" send the information about the control registers which are start, update and backup. Detailed explanation is showed below:



So, this message means that boot with updating the software image. The image which will be boot is sent in another message.

2. The messages start with "01" send the information about the select\_image. In this message "01" is used for the newest software image, "02" is used for the oldest software image and "03" is used for the last backup. Detailed explanation is showed below:



So, this message means that select the oldest software image to boot.

## Ability to jump to different applications

BL can successfully jump to applications that are residing in the specific locations on the RAM. The addresses that BL is currently jumping is defined at the beginning of the code with #define statements.

Things we lack:

1. Executing from non-volatile memory.
2. Checking mechanism

## Executing from Non-Volatile Memory

Executing an image that is residing on non-volatile memory couldn't be accomplished. All the examples and tests are done on the images (also bootloader itself) that resides on RAM.

## Checking Mechanism

The basic structure for the checking mechanism is prepared but the actual checking operation (checksum) is not completed due to timing constraints.

## Addresses of Bootloader and Images

```
grmon2> batch load_app_images_and_bootloader.sh
40011000 .text          19.7kB / 19.7kB [=====>] 100%
40015EC0 .rodata        16B          [=====>] 100%
40015ED0 .data          168B         [=====>] 100%
Total size: 19.87kB (18.08Mbit/s)
Entry point 0x40011000
Image /home/bootloader/bootloader_images/helloWorld1.elf loaded
40022000 .text          19.7kB / 19.7kB [=====>] 100%
40026EC0 .rodata        16B          [=====>] 100%
40026ED0 .data          168B         [=====>] 100%
Total size: 19.87kB (23.25Mbit/s)
Entry point 0x40022000
Image /home/bootloader/bootloader_images/helloWorld2.elf loaded
40033000 .text          19.7kB / 19.7kB [=====>] 100%
40037EC0 .rodata        16B          [=====>] 100%
40037ED0 .data          168B         [=====>] 100%
Total size: 19.87kB (23.25Mbit/s)
Entry point 0x40033000
Image /home/bootloader/bootloader_images/helloWorld3.elf loaded
40044000 .text          19.7kB / 19.7kB [=====>] 100%
40048EC0 .rodata        16B          [=====>] 100%
40048ED0 .data          168B         [=====>] 100%
Total size: 19.87kB (18.08Mbit/s)
Entry point 0x40044000
Image /home/bootloader/bootloader_images/helloWorld4.elf loaded
40000000 .text          21.0kB / 21.0kB [=====>] 100%
40005400 .rodata        16B          [=====>] 100%
40005410 .data          176B         [=====>] 100%
Total size: 21.19kB (21.70Mbit/s)
Entry point 0x40000000
Image /home/bootloader/bootloader_images/bootloader.elf loaded
```

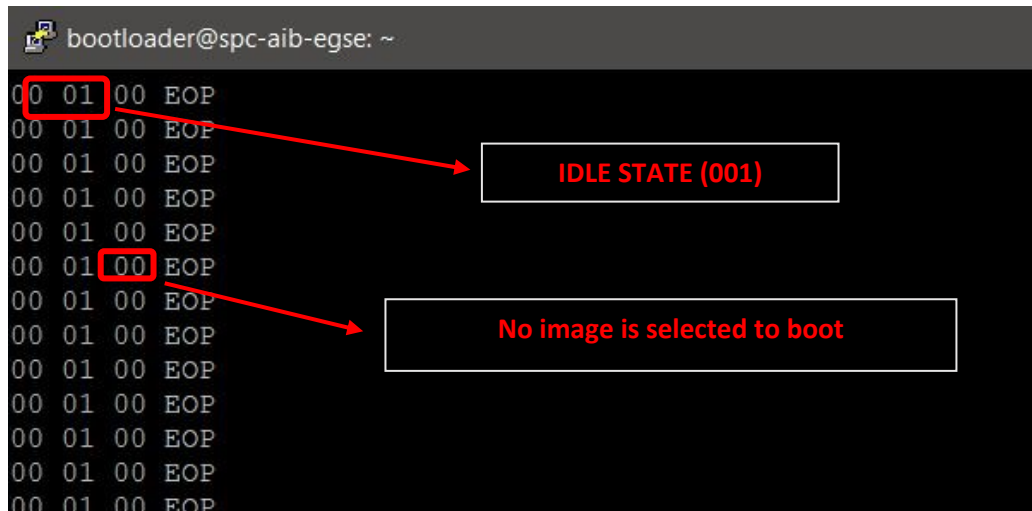
The images and the bootloader are uploaded to the shown addresses above by using GRMON. These addresses are specified in their makefile. Also these addresses are referenced with a define statement at the beginning of the code.



## Sample Runs

### 1. For Booting the First Image

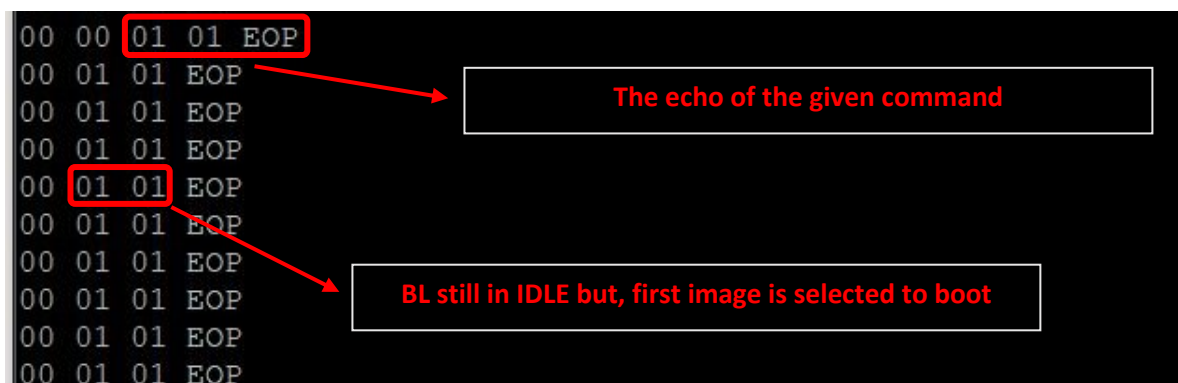
When the BL starts to run, it waits in the IDLE (001) state, and wait for the command.



```
bootloader@spc-aib-egse: ~  
00 01 00 EOP  
00 01 00 EOP  
00 01 00 EOP  
00 01 00 EOP  
00 01 00 EOP  
00 01 00 EOP  
00 01 00 EOP  
00 01 00 EOP  
00 01 00 EOP  
00 01 00 EOP  
00 01 00 EOP  
00 01 00 EOP  
00 01 00 EOP  
00 01 00 EOP
```

The terminal output shows a series of "00 01 00 EOP" lines. A red box highlights the first "00 01" and an arrow points to a text box labeled "IDLE STATE (001)". Another red box highlights the "00" in the sixth line, with an arrow pointing to a text box labeled "No image is selected to boot".

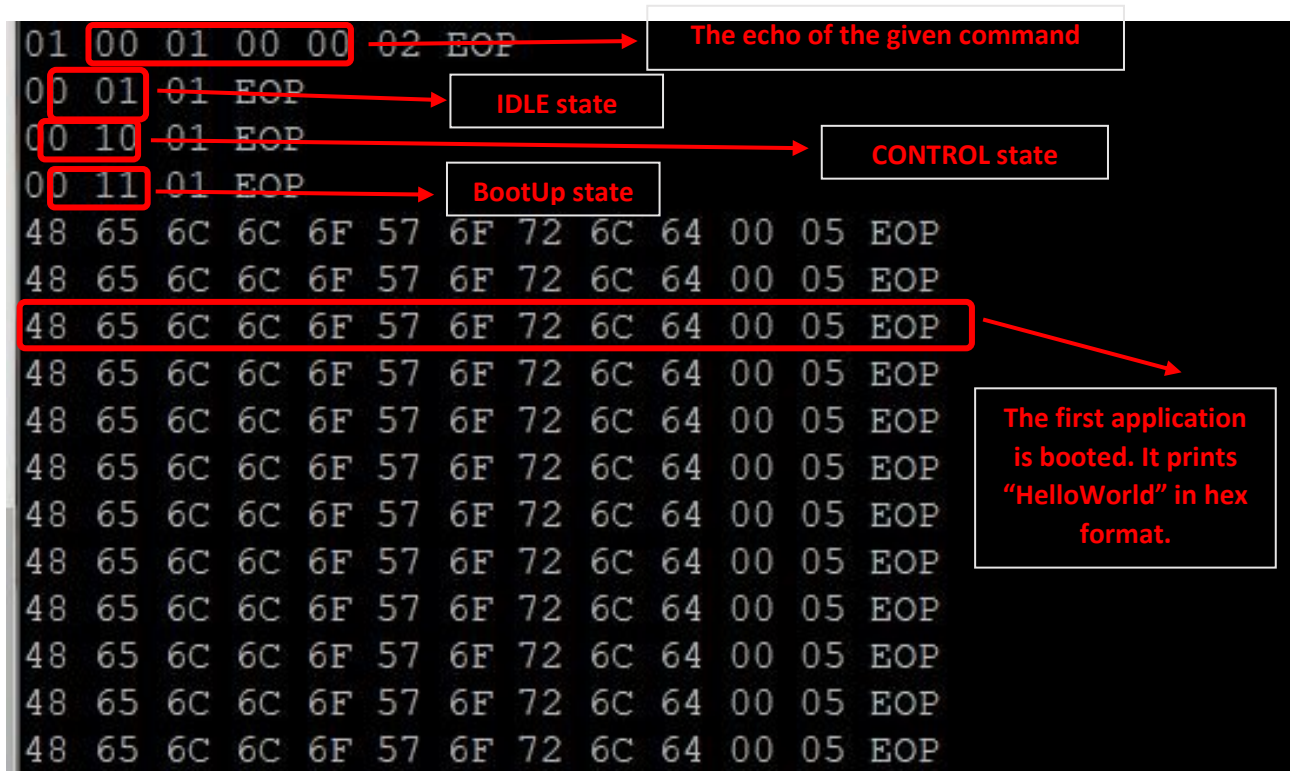
The first command is given to choose the image. For this case the first image is chosen to boot. The given command for this is "01 01 EOP".



```
00 00 01 01 EOP  
00 01 01 EOP  
00 01 01 EOP  
00 01 01 EOP  
00 01 01 EOP  
00 01 01 EOP  
00 01 01 EOP  
00 01 01 EOP  
00 01 01 EOP  
00 01 01 EOP  
00 01 01 EOP
```

The terminal output shows a series of lines. The first line "00 00 01 01 EOP" has "01 01 EOP" highlighted with a red box, and an arrow points to a text box labeled "The echo of the given command". The fifth line "00 01 01 EOP" has "01 01" highlighted with a red box, and an arrow points to a text box labeled "BL still in IDLE but, first image is selected to boot".

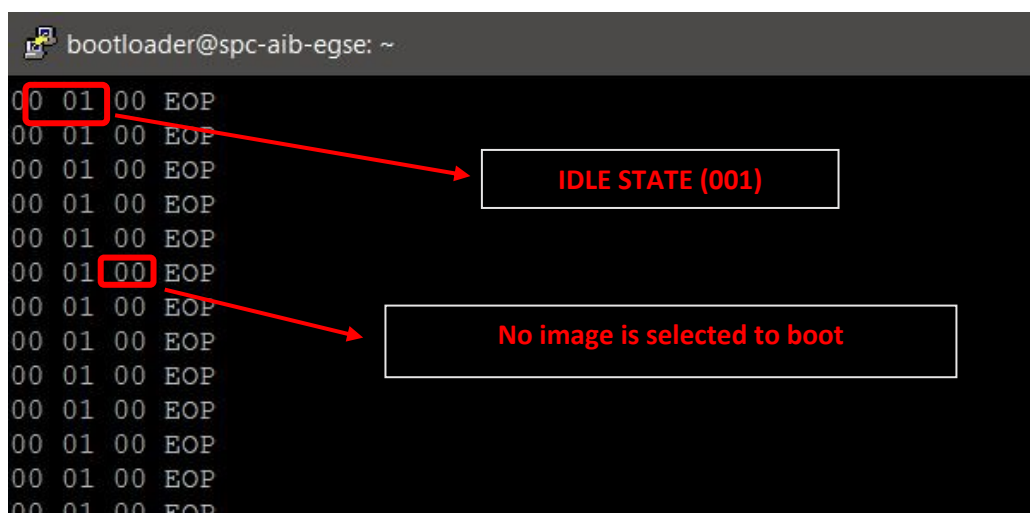
The second command is given to start the booting process. The given command for this is “**00 01 00 00 EOP**”.



After BL gets start command, it follows the state diagram. It switches between IDLE, CONTROL and BootUp states in order. After that it boots to the first application. This application is created for test purposes and it only prints “HelloWorld” message in a loop.

## 2. For Booting the Second Image

When the BL starts to run, it waits in the IDLE (001) state, and wait for the command.



The first command is given to choose the image. For this case the second image is chosen to boot. The given command for this is “ **01 02 EOP** ”.

```

00 eo00 01 02 EOP
00 01 02 EOP
00 01 02 EOP
00 01 02 EOP
00 01 02 EOP
00 01 02 EOP
00 01 02 EOP
00 01 02 EOP
00 01 02 EOP
00 01 02 EOP
00 01 02 EOP

```

The echo of the given command

BL still in IDLE but, second image is selected to boot

The second command is given to start the booting process. The given command for this is “**00 01 00 00 EOP**”.

[illegible]

After BL gets start command, it follows the state diagram. It switches between IDLE, CONTROL and BootUp states in order. After that it boots to the second application. This application is created for test purposes, and it only prints “WorldHello” message in a loop.

### 3. For Booting the BackUp Image

When the BL starts to run, it waits in the IDLE (001) state, and wait for the command.

```
bootloader@spc-aib-egse: ~  
00 01 00 EOP  
00 01 00 EOP  
00 01 00 EOP  
00 01 00 EOP  
00 01 00 EOP  
00 01 00 EOP  
00 01 00 EOP  
00 01 00 EOP  
00 01 00 EOP  
00 01 00 EOP  
00 01 00 EOP  
00 01 00 EOP  
00 01 00 EOP  
00 01 00 EOP  
00 01 00 EOP
```

Diagram annotations:

- Red box around "00 01" in the second line, arrow points to: **IDLE STATE (001)**
- Red box around "00" in the eighth line, arrow points to: **No image is selected to boot**

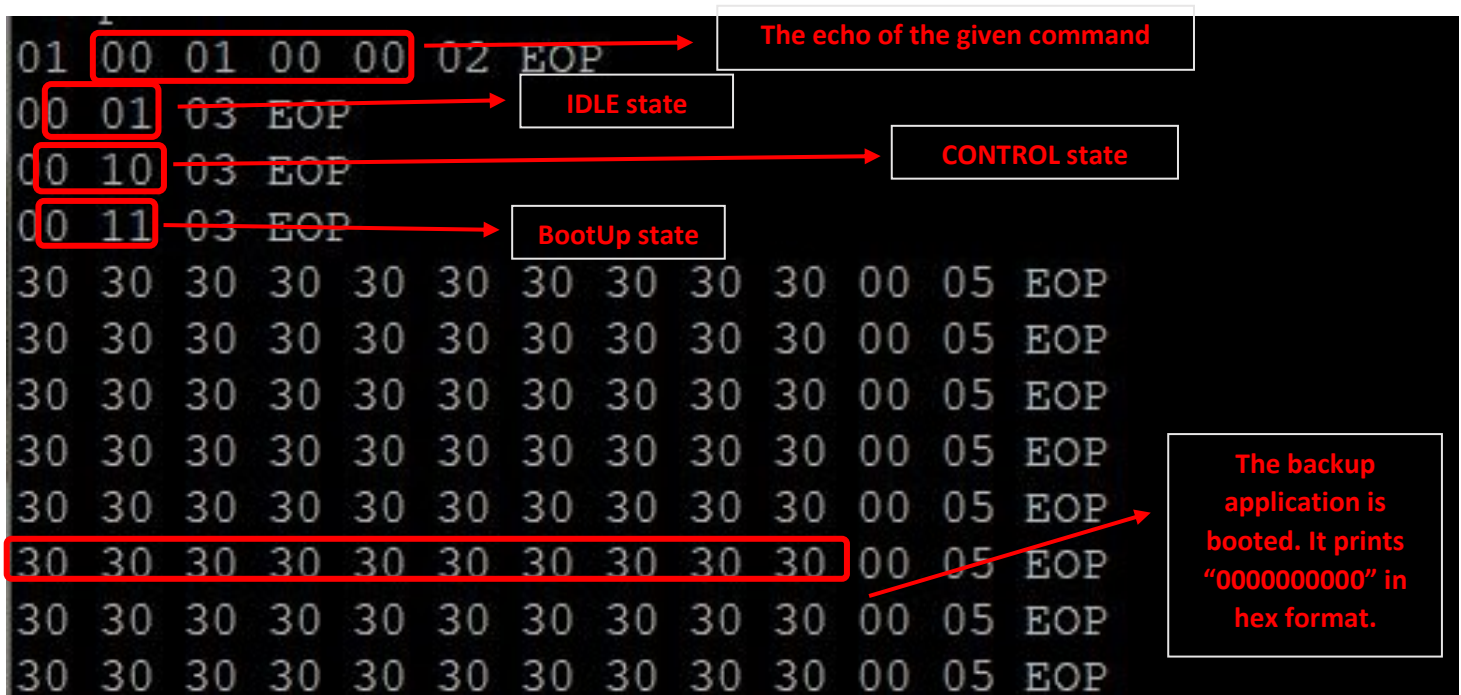
The first command is given to choose the image. For this case the backup (third) image is chosen to boot. The given command for this is “ **01 03 EOP** ”.

```
0000 01 03 EOP  
00 01 03 EOP  
00 01 03 EOP  
00 01 03 EOP  
00 01 03 EOP  
00 01 03 EOP  
00 01 03 EOP  
00 01 03 EOP  
00 01 03 EOP  
00 01 03 EOP  
00 01 03 EOP
```

Diagram annotations:

- Red box around "01 03 EOP" in the first line, arrow points to: **The echo of the given command**
- Red box around "01 03" in the seventh line, arrow points to: **BL still in IDLE but, backup image is selected to boot**

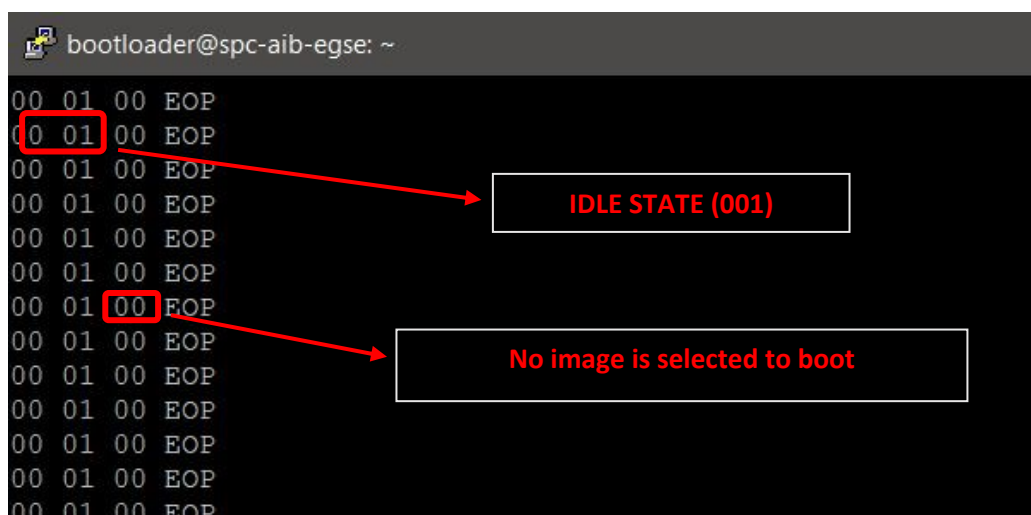
The second command is given to start the booting process. The given command for this is “00 01 00 00 EOP”.



After BL gets start command, it follows the state diagram. It switches between IDLE, CONTROL and BootUp states in order. After that it boots to the backup application. This application is created for test purposes, and it only prints “0000000000” message in a loop.

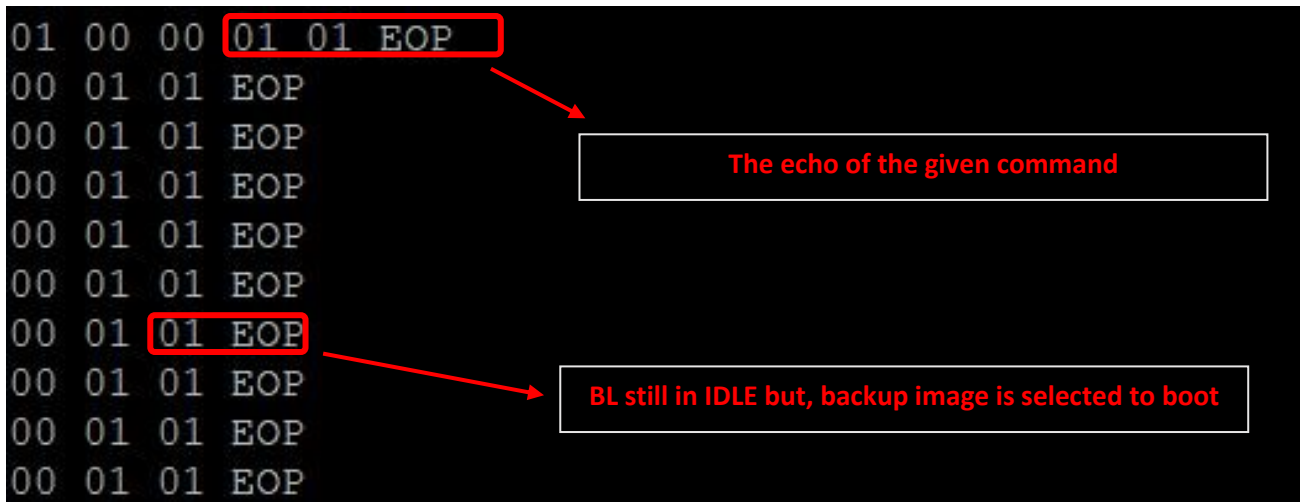
#### 4. For Booting with updating the image

When the BL starts to run, it waits in the IDLE (001) state, and wait for the command.

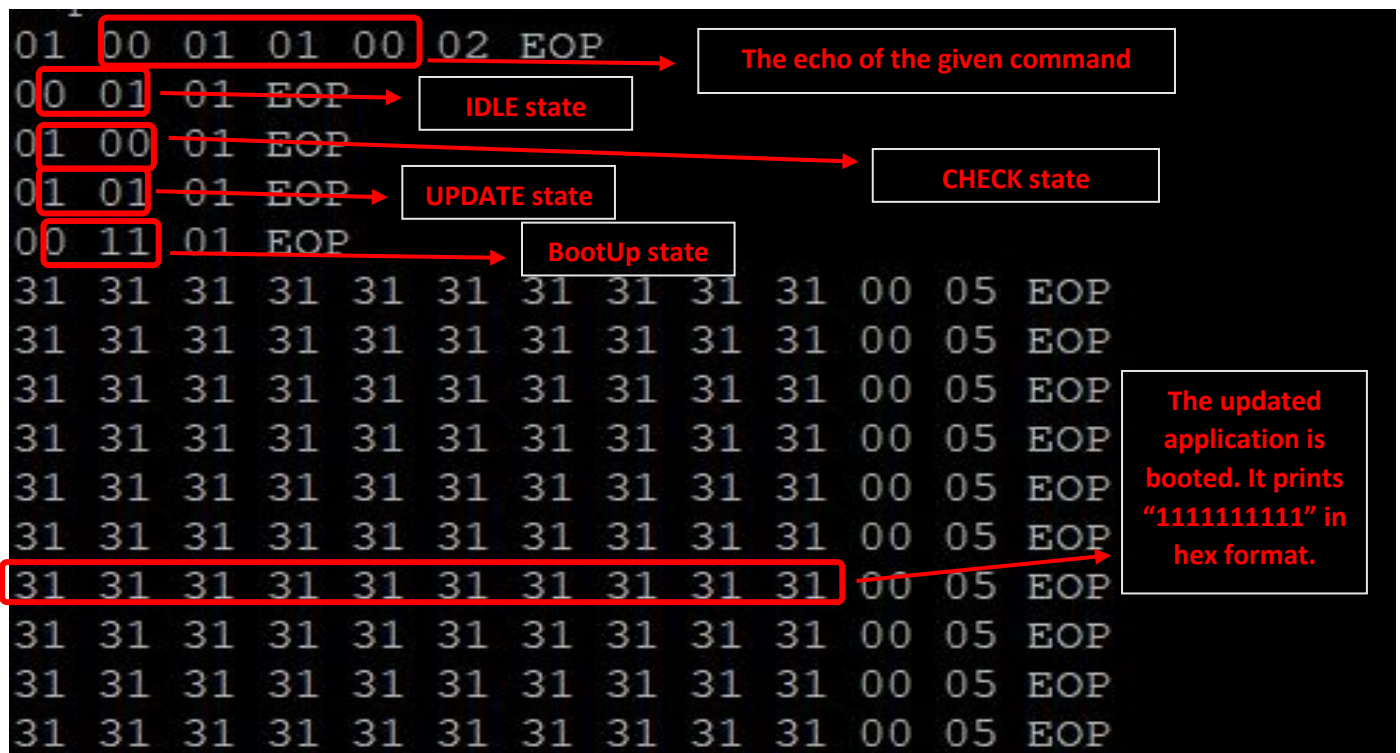




The first command is given to choose the image. For this case the first image is chosen to boot. This image will be updated first, then will be booted to the updated version. The given command for this is “ **01 01 EOP** ”.



The second command is given to start the booting process. The given command for this is “**00 01 01 00 EOP**”.



After BL gets start command, it follows the state diagram. It switches between IDLE, CHECK, UPDATE and BootUp states in order. After that it boots to the backup application. This application is created for test purposes, and it only prints “111111111” message in a loop.

# Challenges Encountered During the Implementation

## Non-Volatile Memory Related Problems

We had two different problems with non-volatile memories on the system. First problem was about the onboard flash that is shipped with the development kit from OCE Technologies. The flash is not detected by the grmon2 and we couldn't write something to it either. After some research from the schematics DT realized that command pins for the flash chip that resides on the development board are not connected. So, it is not possible to send any commands to the flash chip which is needed to write or read something from the chip since the required pins are physically non-connected. Development kit's example code for the flash is also not working because of this hardware problem.

Second problem we had was related to the EEPROMs. We have 5 EEPROM chips on the board connected to the development board by J14 connector. These EEPROM chips are selected via the chipselect pin 1 which can be checked from the J2 on the development board. In our tests we have realized that writing to an address that is between 0x10000000 to 0x1FFFFFFF triggers the chipselect 1 pin so that region is mapped to the EEPROMs. We had two sub-problems related to the EEPROMs. First problem was the fact that after writing something to the EEPROM and re-running the startup.sh was deleting the content of the EEPROMs completely. We have realized that grmon was doing some initializations that are deleting the content so with the -ni flag starting the grmon did not delete the whole content. Second problem arises at this point even though some of the data is preserved with the -ni flag some of the data is gone. We couldn't find out how & why this happens but some of the values that are residing on the EEPROM were still in the EEPROMs after the power cycle by starting grmon with -ni but some of them were gone.

Due to these problems, we could not execute an image from the non-volatile memory. Also, we had to place our bootloader on the RAM.

## Jumping to Application Problem

We had a problem that was solved regarding the jumping operation. We were trying to run the application software like a function. To do that we created a function pointer pointing to the beginning of the application image and tried to call that function. This resulted in a trap since the instruction at the required address is not fetched yet. We tried to solve this issue by writing a trap handler, but we did not succeed with that approach. After some trial and error, we tried to jump to a specific address with inline assembly and it successfully jumped to the beginning of the application image.

## Results, Comments and Future Work

During the project, BL requirements list and implementation details for these requirements are completed mostly. Also, the general structure of the BL is mostly implemented. The implemented BL can run in RAM with the capabilities of;

- booting to different software images which resides in different memory addresses in RAM
- Updating the software image with a received image into RAM

In addition to these BL implementations, some problems are detected in AIB. Because of the various hardware and software problems (explained above), there are problems with using internal flash and

external EEPROM from the board. The detection of these problems is also achieved during the project.

As a future work for the project, implementation of the BL can be improved by adding more complicated and specific capabilities such as adding an improved and more complex image check mechanism into the BL. Furthermore, the problems detected in the hardware and software should be studied and these problems should be solved with various methods. After solving these problems, BL should be modified to run in non-volatile memory as in the requirements.