UART User Bootloader Setting for Updating User Application Firmware

HB Rev. 1.09

Introduction

The user bootloader can be used to update the user application firmware in the code flash memory of the target microcontroller via UART communication between the host computer and the microcontroller. The bootloader project, application project, programming tool (E-PGM+, A-Link debugger, etc.), and UART downloader tool are required to set up the user bootloader.

This document describes how to set up the user bootloader that downloads firmware to code flash memory of the ABOV 32-bit Arm Cortex-M microcontrollers using the UART interface. This document also provides relevant examples.

Table 1 shows the list of microcontrollers to which this document applies.

Table 1. Applicable Devices

Base Product	Part Number
A31G11x	A31G112KN, A31G112GR, A31G112LU, A31G112SQ, A31G112KY, A31G112KU, A31G112CL, A31G111KU, A31G111KN, A31G111LU, A31G111GR
A31G12x	A31G123CL, A31G123MM, A31G123RM, A31G123RL, A31G123SQ, A31G123ML, A31G123RN, A31G122RM, A31G122RL, A31G122MM, A31G122ML
A31G21x	A31G213CL, A31G213KN, A31G213SQ, A31G213GR, A31G212GR, A31G212SQ, A31G212CL, A31G212KN
A31G31x	A31G313CL, A31G313CU, A31G313RL, A31G313RM, A31G313SN, A31G314CL, A31G314CU, A31G314ML, A31G314MM, A31G314RL, A31G314RM, A31G314SN, A31G316ML, A31G316MM, A31G316RL, A31G316RM
A33G52x	A33G527VQ, A33G527VL, A33G527RL, A33G526VQ, A33G526VL, A33G526RM, A33G526ML, A33G526RL, A33G526MM, A33G524RM, A33G524ML, A33G524RL, A33G524MM
A34M41x	A34M418YL, A34M418VL, A34M418RL, A34M416VL, A34M416RL, A34M414VL, A34M414RL
A31T21x	A31T214CL, A31T214RL, A31T214SN, A31T216CL, A31T216RL, A31T216SN
A33G53x	A33G539VQ, A33G539VL, A33G539MM, A33G539RL A33G538VQ, A33G538VL, A33G538MM, A33G538RL
A34M420	A34M420YL, A34M420VL, A34M420RL
A34L716	A34L716VL, A34L716RL
A34M456	A34M456VL, A34M456RL, A34M456RK

Reference Document

The documents below are available on www.abovsemi.com:

- User's manual for 32-bit microcontroller device
- Datasheet for 32-bit microcontroller device
- Example code
  - Bootloader and Application project files

Contents

Introduction 1
Reference Document 2
1. User Bootloader Implementation Sequence 6
2. Memory User Map for Bootloader 7
2.1 Bootloader Area 7
2.2 Application Area 7
3. Bootloader Programming Sequence 8
4. Bootloader Project Setting 9
4.1 startup_(DeviceName)_bootkeil.s 9
4.2 main.c 9
4.3 Bootloader Information Setting for Each Microcontroller (def.h or defboot.h) 10
4.3.1 A31G213 in defboot.h 11
4.3.2 A31G123 in defboot.h 11
4.3.3 A33G527 in defboot.h 12
4.3.4 A31M418 in defboot.h 12
4.3.5 A33G539 in defboot.h 13
4.3.6 A34M420 in defboot.h 13
4.3.7 A34L716 in defboot.h 14
4.4 Limitations 14
4.5 Bootloader Project Option Setting 15
5. Bootloader Project – Hex to Array 16
6. Application Project Setting 17
6.1 main.c 17
6.2 bootcode.c 18
6.2.1 A31G213 Application Project 19
6.3 Application Project Option Setting 20
7. Download Application Firmware via UART 22
8. Conclusion 23
Appendix – Downloader Tool Protocol Flowchart 24
Revision History 25

List of Tables

Table 1. Applicable Devices 1

List of Figures

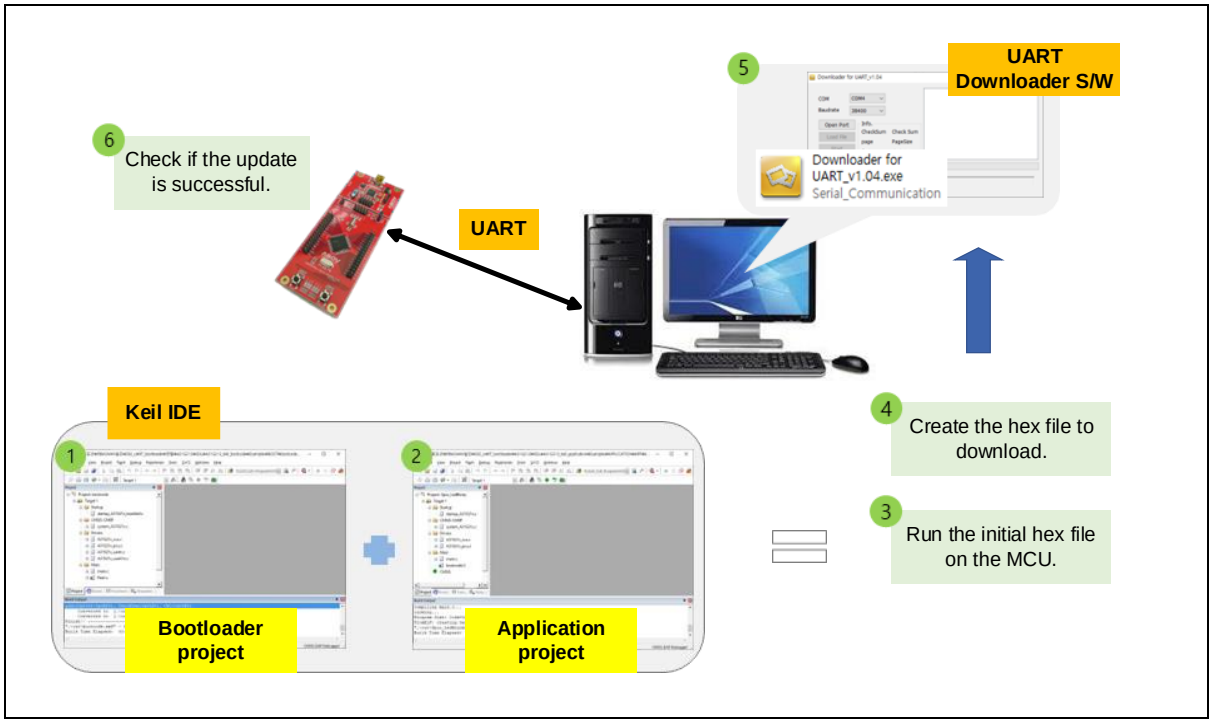
Figure 1. Implementation Procedure for User Bootloader 6
Figure 2. User Bootloader Flash Memory Map (A31G213) 7
Figure 3. A31G213 Bootloader Programming Flowchart and Programming Example 8
Figure 4. Keil Setting Example of “Option for Target” 15
Figure 5. Keil Setting Example of “Hex to Array C file conversion” 16
Figure 6. Example of bootcode.c File Copy 18
Figure 7. Example of Applying bootcode.c to Application Project 18
Figure 8. Example of Applying bootcode.c to A31G213 Project 19
Figure 9. “Options for Target” Setting Example of Application Project 20
Figure 10. bootcode.c File Option Example of Application Project 21
Figure 11. Application Firmware Download Process 22
Figure 12. Initial Protocol for Downloader Tool 24
Figure 13. Data Transmission Protocol for Downloader Tool 24

1. User Bootloader Implementation Sequence

This chapter describes the implementation sequence of the user bootloader.

Users can implement a user bootloader that updates the user application firmware by using the Bootloader project, Application project, programming tool (E-PGM+, A-Link debugger, etc.), and the UART downloader tool in the appropriate order, as shown in Figure 1.

Figure 1. Implementation Procedure for User Bootloader



1. Build the Bootloader project in the Keil environment to create a bootloader hex file.
2. Convert the bootloader hex file to a C file and add it to the Application project. Then, build the Application project in the Keil environment to create an application hex file.
3. Use a programming tool (E-PGM+, A-Link debugger) to download the application hex file to the target board for the first time.
4. If the application firmware code is modified, build the modified Application project in the Keil environment and create a new hex file to download. Of course, in this case, the modified Application project must include a bootloader file in C code format.
5. The new hex file created in step 4 can be downloaded to the target board using the UART downloader tool: Use the UART downloader tool to set up the communication serial ports and baud rate, then download the application hex file through the UART channel.
6. Verify that the application firmware on the target board operates correctly after the application hex file is completely updated.

2. Memory User Map for Bootloader

This chapter describes how to configure the flash memory area for the user bootloader.

Examples in this chapter are based on the A31G213 microcontroller.

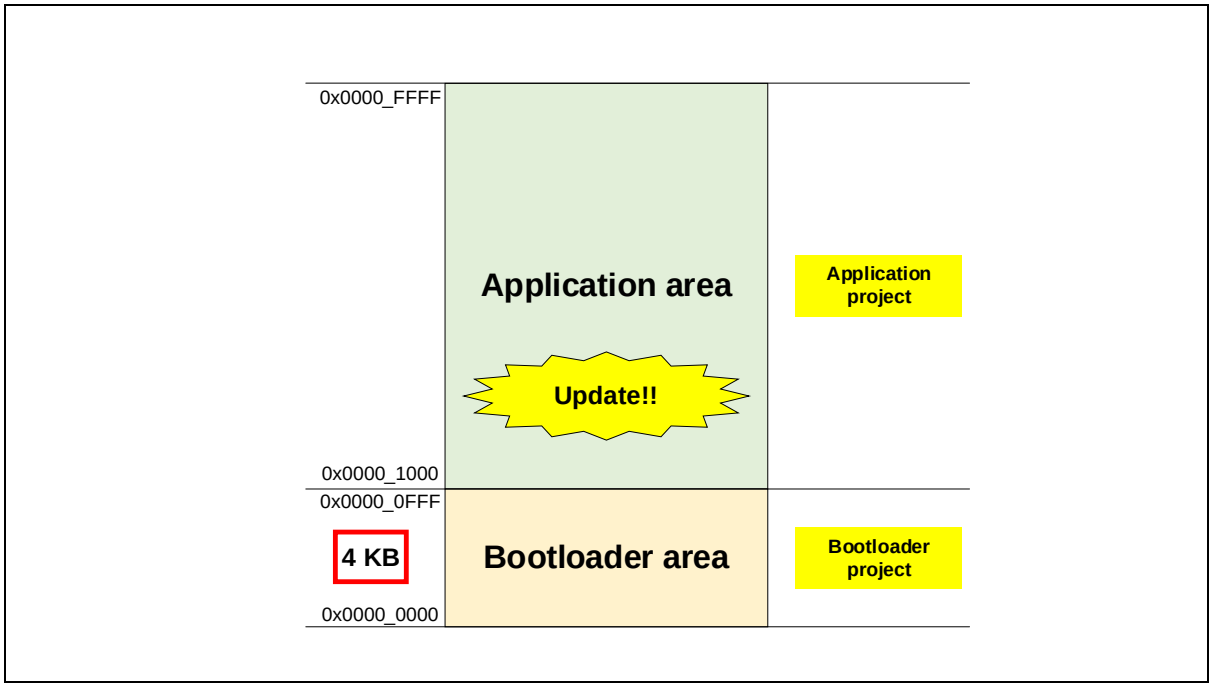
2.1 Bootloader Area

- The bootloader firmware operates in this area (for application firmware updates).
- Size: 4 KB (0x0000 ~ 0x0FFF)

2.2 Application Area

- The application firmware operates in this area.
- The application firmware is updated in this area by the user bootloader.
- Size: 60 KB (the remaining area excluding the bootloader area: 0x1000 ~ 0xFFFF)

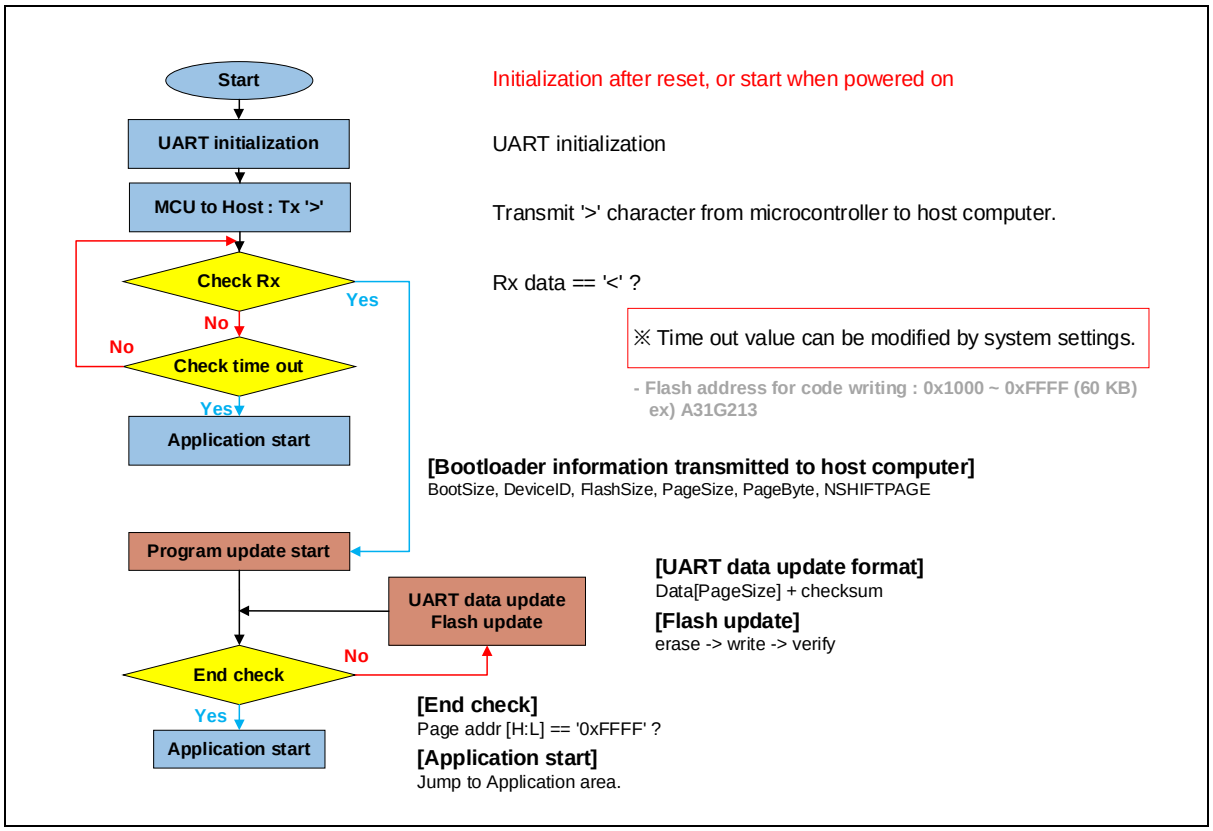
Figure 2. User Bootloader Flash Memory Map (A31G213)



3. Bootloader Programming Sequence

Figure 3 shows the sequence of bootloader operations and provides a programming example based on the A31G213 microcontroller.

Figure 3. A31G213 Bootloader Programming Flowchart and Programming Example



4. Bootloader Project Setting

This chapter describes how to set up the bootloader code that is programmed and executed in the bootloader area, assuming that the development environment is Keil.

- Project path: ~/Example/BOOT/bootcode/keil/
- Keil project: bootcode.uvprojx

4.1 startup_(DeviceName)_bootkeil.s

1. The application stack pointer and start address are defined.
   A. Application area of A31G213: 0x1000 ~ 0xFFFF
2. The _fjump function is used to jump to the specified address in the application area.

```
1               AREA |.text|, CODE, READONLY
2
3               ; jump application start
4               _fjump PROC
5               EXPORT _fjump
6
7               ldr r0, =0x1000           ; Application stack pointer
8               ldr r0, [r0]
9               mov sp, r0
10              ldr r0, =0x1004           ; Start application
11              ldr r0, [r0]
12              bx r0
13              NOP
14              NOP
15              NOP
16              NOP
17
18
19              ENDP
```

4.2 main.c

The _fjump() is defined as an extern in the main.c file.

```
1           //// Assembly jump \
2           code extern void _fjump(void);
3           …
4           _fjump();
```

4.3 Bootloader Information Setting for Each Microcontroller (def.h or defboot.h)

The bootloader transmits code flash memory information of the target microcontroller to the software used to download the application firmware via the UART channel.

The code flash memory information of the target microcontroller is defined in def.h or defboot.h file.

The code below is the code flash-specific values defined in the def.h file, and users must check the code flash memory information of the target microcontroller before setting the bootloader.

For the code flash memory information of each microcontroller, you can refer to the user's manual and datasheet of the microcontroller.

```
#define ABOV                ‘A’

#define Flash1k             ‘g’
#define Flash2k             ‘h’
#define Flash4k             ‘’'
#define Flash8k             ‘l’
#define Flash16k            'm'
#define Flash32k            ‘n’
#define Flash64k            ‘o'
#define Flash128k           ‘p’
#define Flash256k           ‘q’
#define Flash384k           ‘r’
#define Flash512k           ‘s’
#define Flash768k           ‘t’
#define Flash1024k          ‘u’

#define EEprom64            ‘.’
#define EEprom128           ‘/’
#define EEprom256           ‘0’
#define EEprom512           ‘1’
#define EEprom1024          ‘2’
#define EEprom2048          ’3’
#define EEprom4096          ‘4’

#define Boot128             ‘a'
#define Boot256             ‘b’
#define Boot512             ‘c’
#define Boot1024            ‘d'
#define Boot2048            ‘e’
#define Boot4096            ‘f’
#define Boot8192            ‘g’

#define Page32              ‘Q’
#define Page64              ‘R’
#define Page128             ‘S’
#define Page256             ‘T’
#define Page512             ‘V’
#define Page1024            ‘W’
#define Page2048            ‘X’
#define Page4096            ‘Y’
```

The sections below provide examples of applying information on some target microcontrollers.

4.3.1 A31G213 in defboot.h

- BootSize: Boot4096 ('f')
- DeviceID: ABOV (‘A‘)
- FlashSize: Flash64k (‘o’)
- PageSize: Page512 (‘V’) ← Unit size for erase and program
- PageByte: Same as the PageSize (512)
- NSHIFTPAGE: Shift number for calculating addresses (512 = 2^9)

```
1    #define BootSize        Boot4096
2    #define DeviceID        ABOV
3    #define FlashSize       Flash64k
4    #define PageSize        Page512
5    #define EEpromSize      EEprom64
6    #define PageByte        512
7    #define NSHIFTPAGE      9
```

4.3.2 A31G123 in defboot.h

- BootSize: Boot4096 ('f')
- DeviceID: ABOV (‘A‘)
- FlashSize: Flash64k (‘o’)
- PageSize: Page128 (‘S’) ← Unit size for erase and program
- PageByte: Same as the PageSize (128)
- NSHIFTPAGE: Shift number for calculating addresses (128 = 2^7)

```
1    #define BootSize        Boot4096
2    #define DeviceID        ABOV
3    #define FlashSize       Flash64k
4    #define PageSize        Page512
5    #define EEpromSize      EEprom64
6    #define PageByte        128
7    #define NSHIFTPAGE      7
```

4.3.3 A33G527 in defboot.h

- BootSize: Boot4096 (‘f’)
- DeviceID: ABOV (‘A’)
- FlashSize: Flash384k (‘r’)
- PageSize: Page1024 (‘W’) ← Unit size for erase and program
- PageByte: Same as the PageSize (1024)
- NSHIFTPAGE: Bit-shift number for calculating addresses (1024 = 2^{10})

```
1    #define BootSize        Boot4096
2    #define DeviceID        ABOV
3    #define FlashSize       Flash384k
4    #define PageSize        Page1024
5    #define EEpromSize      EEprom64
6    #define PageByte        1024
7    #define NSHIFTPAGE      10
```

4.3.4 A31M418 in defboot.h

- BootSize: Boot4096 (‘f’)
- DeviceID: ABOV (‘A’)
- FlashSize: Flash512k (‘s’)
- PageSize: Page1024 (‘W’) ← Unit size for erase and program
- PageByte: Same as the PageSize (1024)
- NSHIFTPAGE: Bit-shift number for calculating addresses (1024 = 2^{10})

```
1    #define BootSize        Boot4096
2    #define DeviceID        ABOV
3    #define FlashSize       Flash512k
4    #define PageSize        Page1028
5    #define EEpromSize      EEprom32
6    #define PageByte        1024
7    #define NSHIFTPAGE      10
```

4.3.5 A33G539 in defboot.h

- BootSize: Boot4096 (‘f’)
- DeviceID: ABOV (‘A’)
- FlashSize: Flash768k (‘t’)
- PageSize: Page2048 (‘X’) ← Unit size for erase and program
- PageByte: Same as the PageSize (2048)
- NSHIFTPAGE: Bit-shift number for calculating addresses (2048 = 2^{11})

```
#define BootSize        Boot4096
#define DeviceID        ABOV
#define FlashSize       Flash768k
#define PageSize        Page2048
#define EEpromSize      EEprom64
#define PageByte        2048
#define NSHIFTPAGE      11
```

4.3.6 A34M420 in defboot.h

- BootSize: Boot4096 ('f')
- DeviceID: ABOV (‘A‘)
- FlashSize: Flash1024k (‘u’)
- PageSize: Page2048 (‘X’) ← Unit size for erase and program
- PageByte: Same as the PageSize (2048)
- NSHIFTPAGE: Shift number for calculating addresses (2048 = 2^{11})

```
#define BootSize        Boot4096
#define DeviceID        ABOV
#define FlashSize       Flash768k
#define PageSize        Page2048
#define EEpromSize      EEprom64
#define PageByte        2048
#define NSHIFTPAGE      11
```

4.3.7 A34L716 in defboot.h

- BootSize: Boot4096 ('f')
- DeviceID: ABOV (‘A‘)
- FlashSize: Flash256k (‘Q’)
- PageSize: Page2048 (‘X’) ← Unit size for erase and program
- PageByte: Same as the PageSize (2048)
- NSHIFTPAGE: Shift number for calculating addresses (2048 = 2^{11})

```
1    #define BootSize        Boot4096
2    #define DeviceID        ABOV
3    #define FlashSize       Flash256k
4    #define PageSize        Page2048
5    #define EEpromSize      EEprom32
6    #define PageByte        2048
7    #define NSHIFTPAGE      11
```

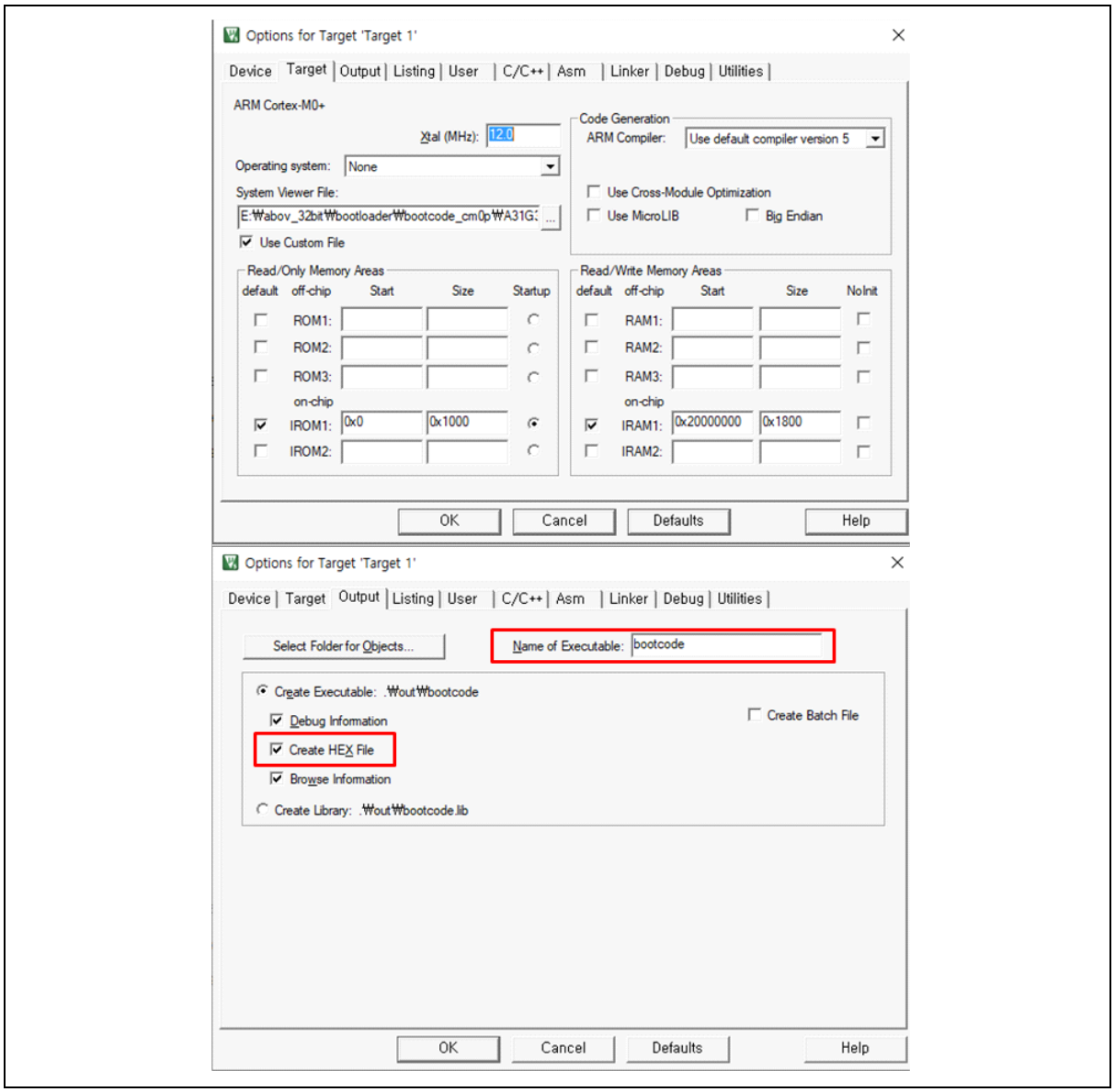
4.4 Limitations

- UART
  - Bootloader: To successfully program code flash memory using the bootloader, users must disable global interrupts and use the UART only in polling mode when transmitting and receiving data.
  - Application: UART interrupt mode is available.

4.5 Bootloader Project Option Setting

1. Enter the output file name (hex file) of the Bootloader project as “bootcode”.
2. Select the “Create HEX file” option for the Bootloader project and compile the Bootloader project. The ″bootcode.hex″ file is created after compilation.
3. Convert the “bootcode.hex” file to the ″bootcode.c″ file that is represented as array values using the “hex to array” conversion tool. (See chapter 5.)
4. Add the “bootcode.c” file to the Application project. For the convenience of development, the example in this section uses the same file name, “bootcode”, in the Bootloader and Application projects.

Figure 4. Keil Setting Example of “Option for Target”

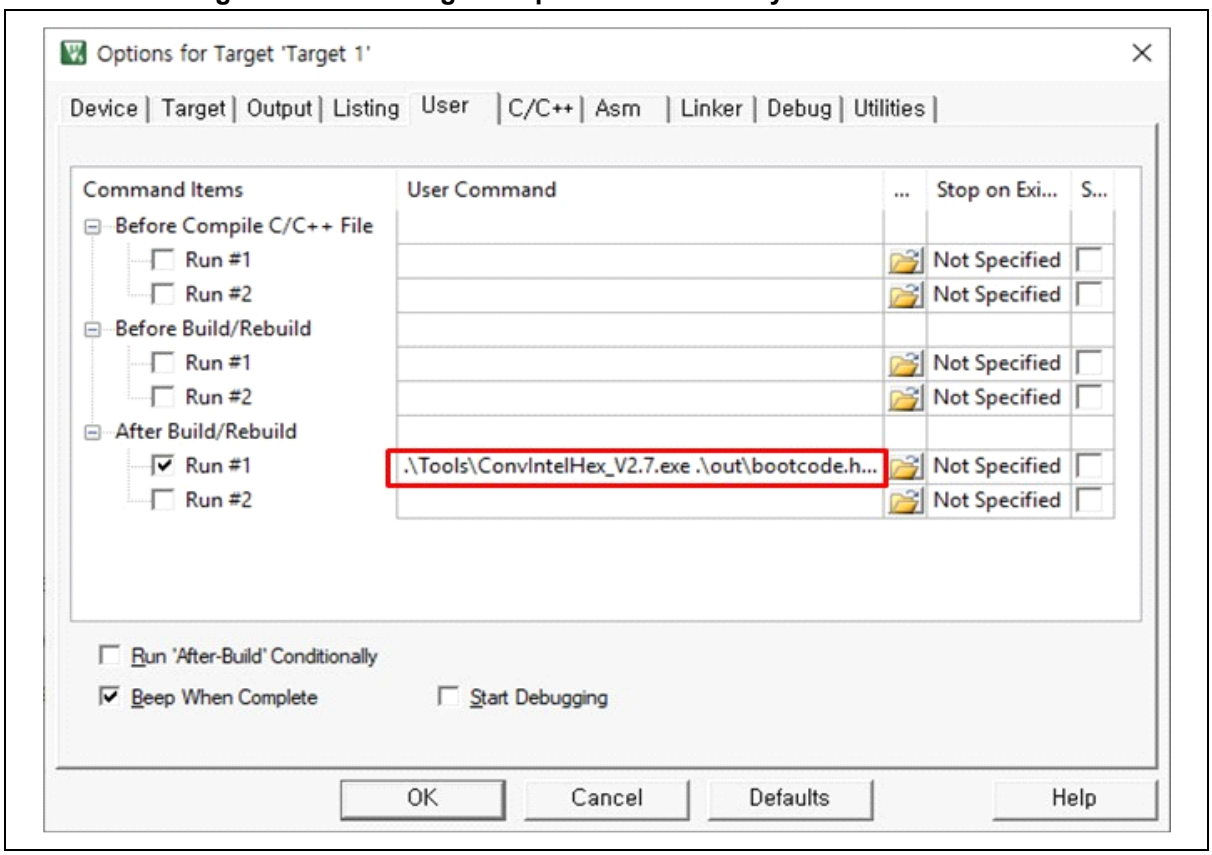


5. Bootloader Project – Hex to Array

This chapter describes how to convert the hex file compiled in the Bootloader project to the array C file format.

1. The “ConvIntelHex_V2.7.exe” file is located in the path below:
   A. DeviceName_keil_bootcode\Examples\BOOT\bootcode\Keil\Tools\ConvIntelHex_V2.7.exe
2. On the “User” tab of the “Options for Target” window shown in Figure 5, check that the “After Build/Rebuild” field displays the path for the conversion program shown below and other information, such as the location where the hex file is output after the project build and the file name, and options.
   A. Path for the conversion program: .\Tools\ConvIntelHex_V2.7.exe .\out\bootcode.hex -ed [0,1000]
3. When you build the Bootloader project, the “ConvIntelHex_V2.7.exe” file is executed to convert the hex file to the array C format file (out\bootcode.c).
4. The converted C format file must be added to the Application project. (See section 0.)

Figure 5. Keil Setting Example of “Hex to Array C file conversion”



6. Application Project Setting

This chapter describes how to set up application code, assuming that the development environment is Keil.

- Project path: ~/Example/APPLICATION/APP/Keil/
- Keil project: Gpio_LedBlinky.uvprojx (Example of Application project)

6.1 main.c

The vector table offset is redefined at the top of the main() function of the main.c file.

To redefine the VTOR, first set the global interrupt to disable and specify the starting address of the VTOR as the first address in the application area.

```
1    ...
2    __disable_irq(); //Interrupt disable
3
4    SCB->VTOR = 0x00001000; //Vector Table Offset Register
5    ...
```

6.2 bootcode.c

1. Copy and paste the “bootcode.c” file (converted from the hex file) into the folder where the main.c file of the Application project is located.
2. Add the copied “bootcode.C” file to the Application project.
3. If there are any changes to the Bootloader project, follow the procedures in chapter 4 and chapter 5 again to update the “bootcode.c” file. Then, proceed with steps 1 to 3, shown in Figure 1, and make an initial download of the modified application hex file to the target microcontroller.

Figure 6. Example of bootcode.c File Copy

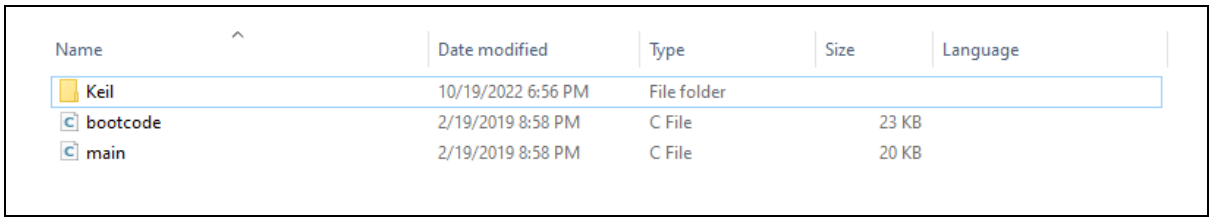
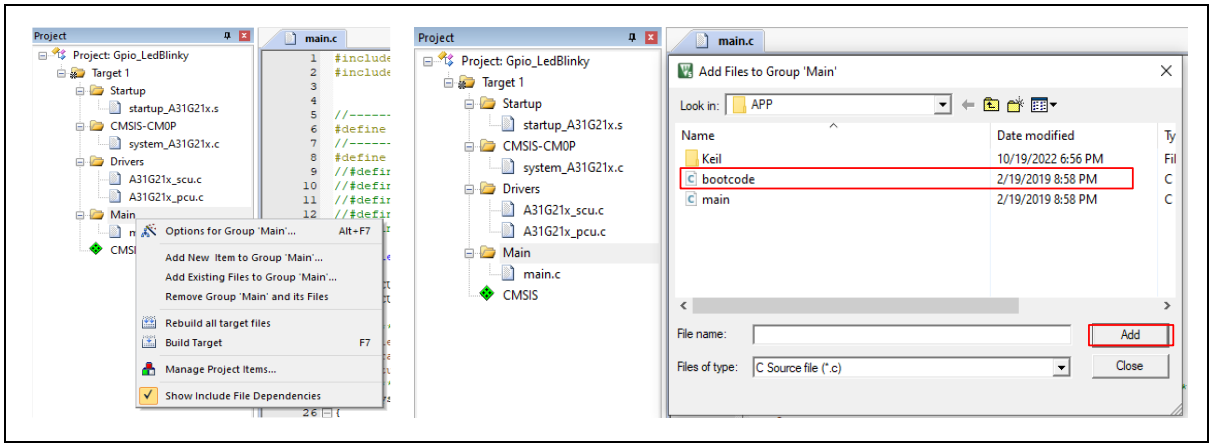
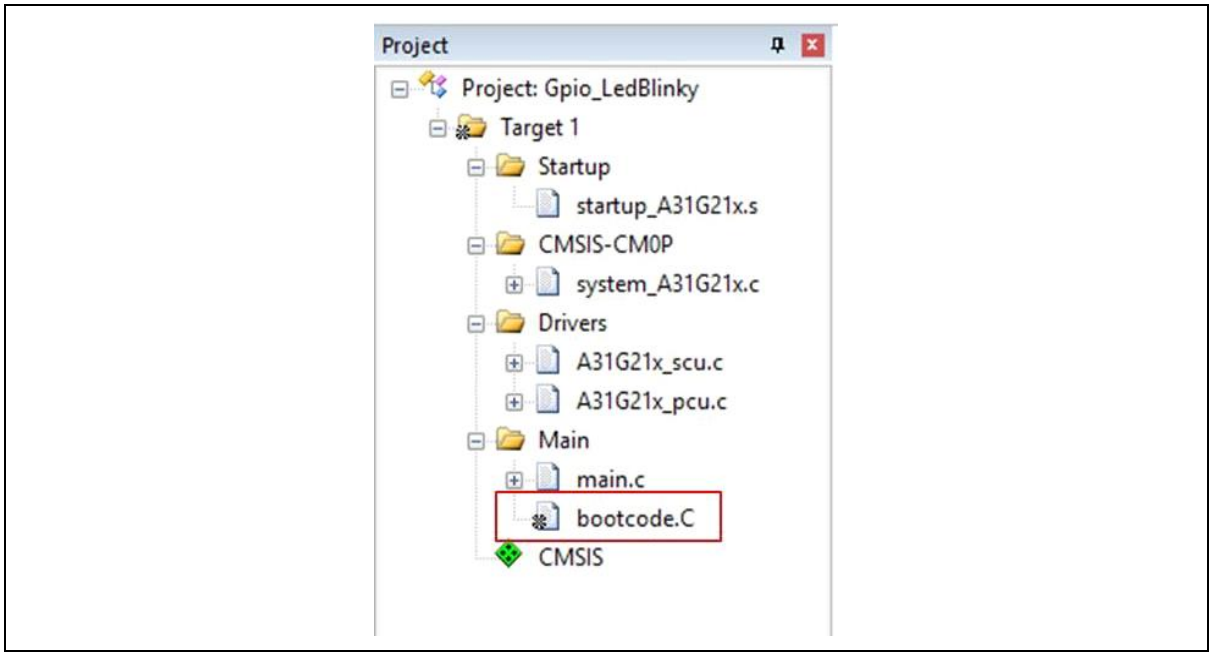


Figure 7. Example of Applying bootcode.c to Application Project



6.2.1 A31G213 Application Project

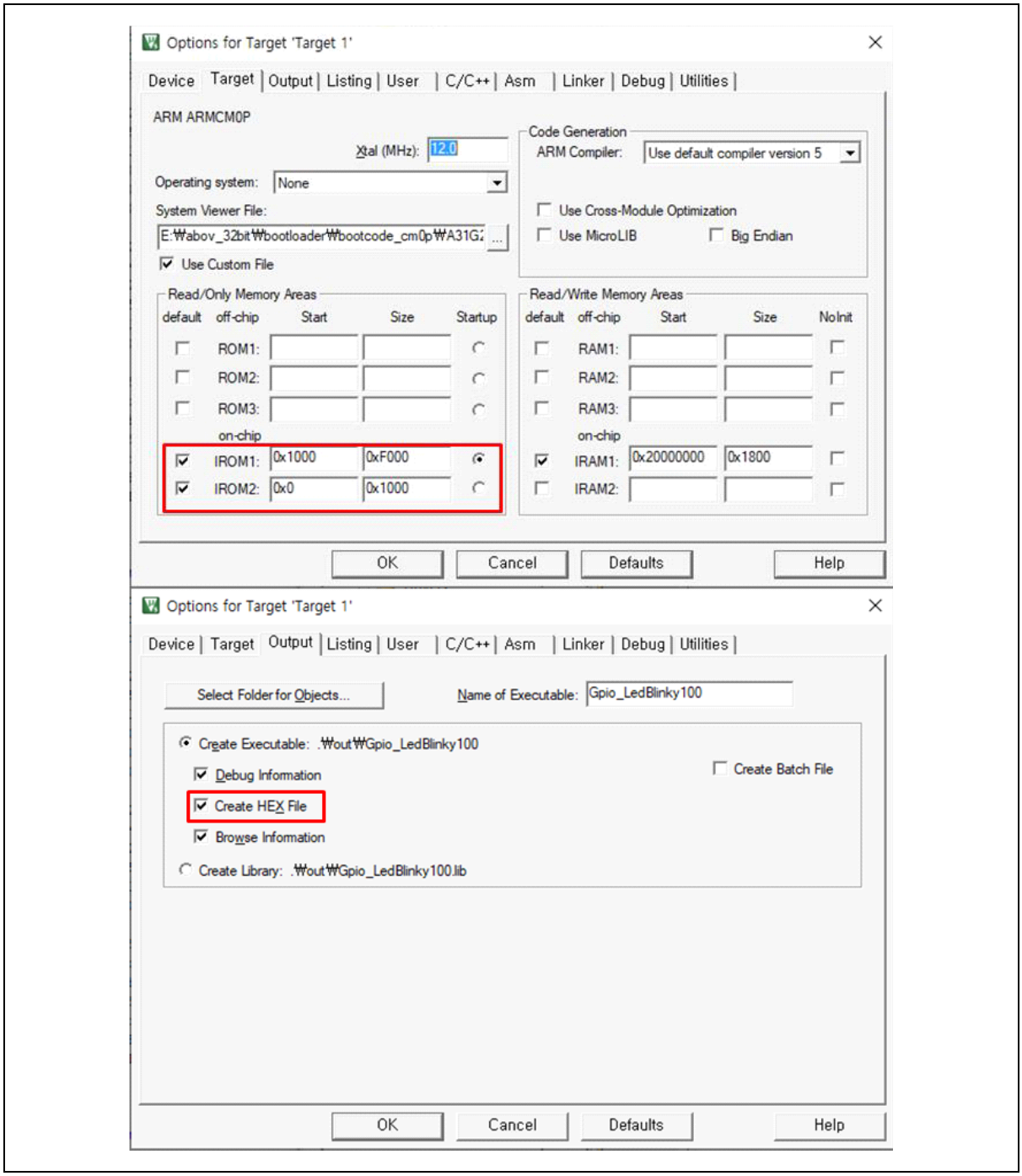
Figure 8. Example of Applying bootcode.c to A31G213 Project



6.3 Application Project Option Setting

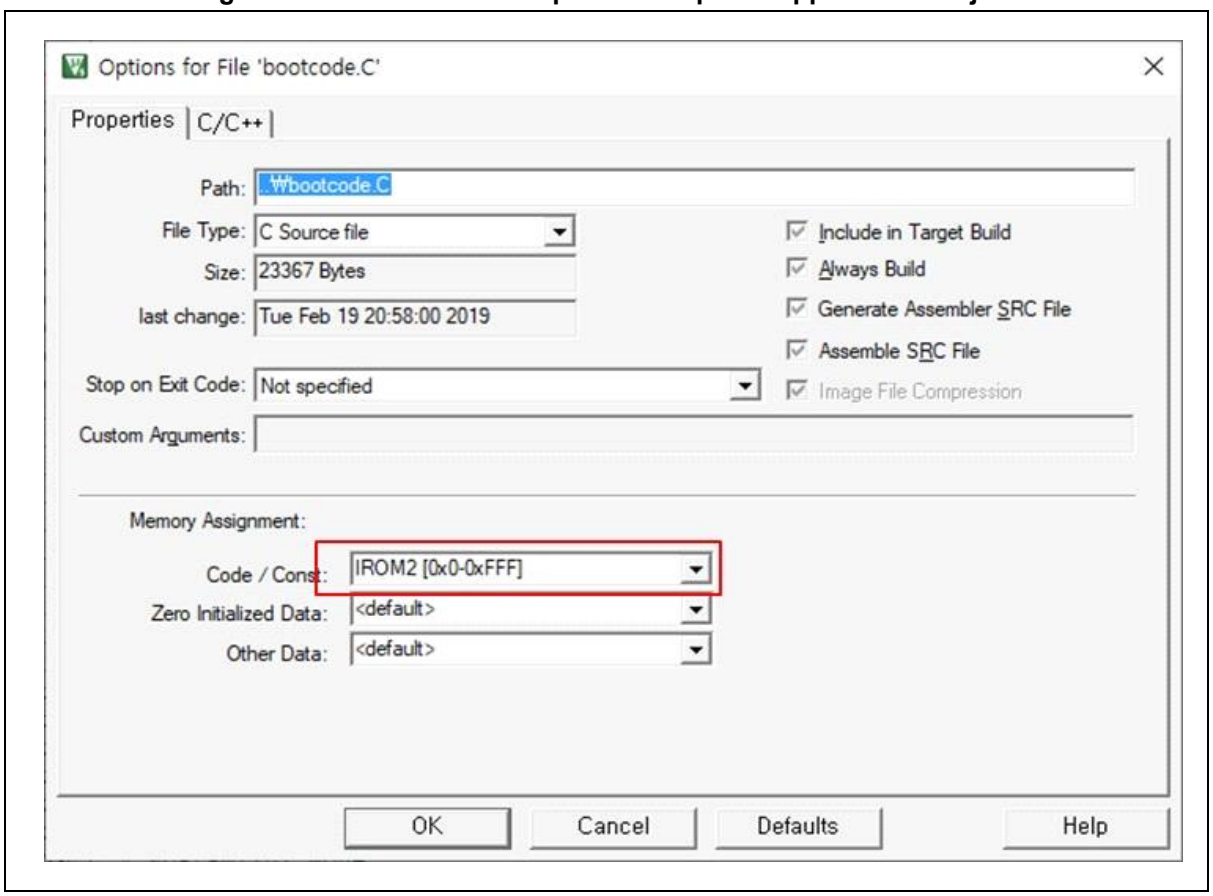
1. Set IROM1 as the first address of the application area and IROM2 as the first address of the bootloader area.
2. Select the “Create Hex file” option (to create the hex file used by the UART downloader tool).

Figure 9. “Options for Target” Setting Example of Application Project



3. On the “Properties” tab of the “Options for File” window, specify the bootloader location using the address of the IROM2 area, as shown in Figure 10.

Figure 10. bootcode.c File Option Example of Application Project



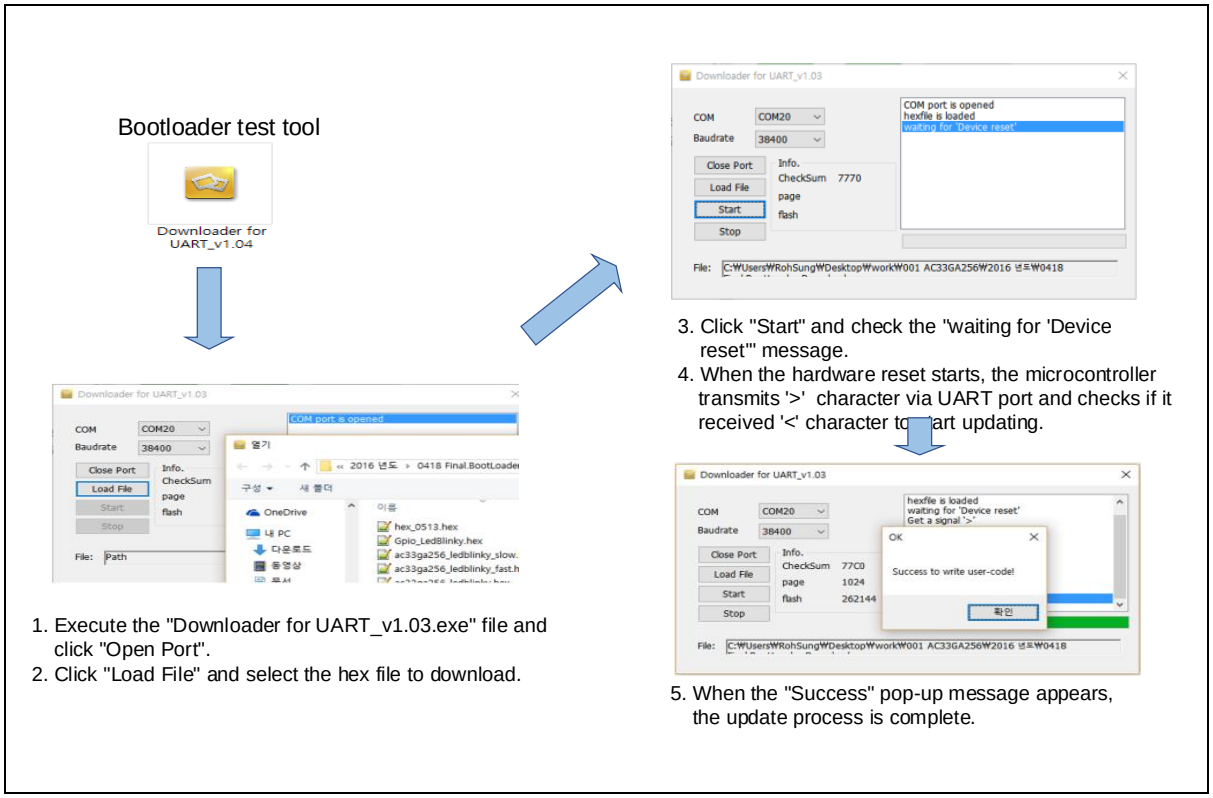
4. Build the application project to create the hex file.

7. Download Application Firmware via UART

This chapter describes the process of updating the application hex file using the UART downloader tool as follows:

1. The UART downloader tool is located in the path below:
   A. Program Path: ~/Downloader_for_UART_v1.04/Downloader_for_UART_v1.04.exe
2. Build the application project to create the hex file and make an initial download of the hex file to the target microcontroller using a programming tool (E-PGM+, A-Link debugger). This step corresponds to the step 3 in Figure 1.
3. With the bootloader code running on the target device, execute the “Downloader for UART_v1.04.exe” file to download the application hex file via UART serial communication. (See Figure 11.)
4. Once the application firmware has been completely updated, restart the target device and check if the application code is operating normally.

Figure 11. Application Firmware Download Process



8. Conclusion

This document describes how to set up the user bootloader to update the user application firmware on the target microcontroller. For this purpose, we use the Bootloader project, Application project, programming tool (E-PGM+, A-Link debugger, etc.), and UART downloader tool in this document.

It is important to check Table 1 and refer to the reference documentation, as it is necessary to know the flash memory information of the target microcontroller.

Figure 1 shows the entire process of updating the user application firmware using the user bootloader. Each chapter in this document includes examples that guide users through the correct setup of the Bootloader and Application projects.

Chapter 7 provides how to use the UART downloader tool to update the user application firmware successfully.

Appendix – Downloader Tool Protocol Flowchart

Figure 12. Initial Protocol for Downloader Tool

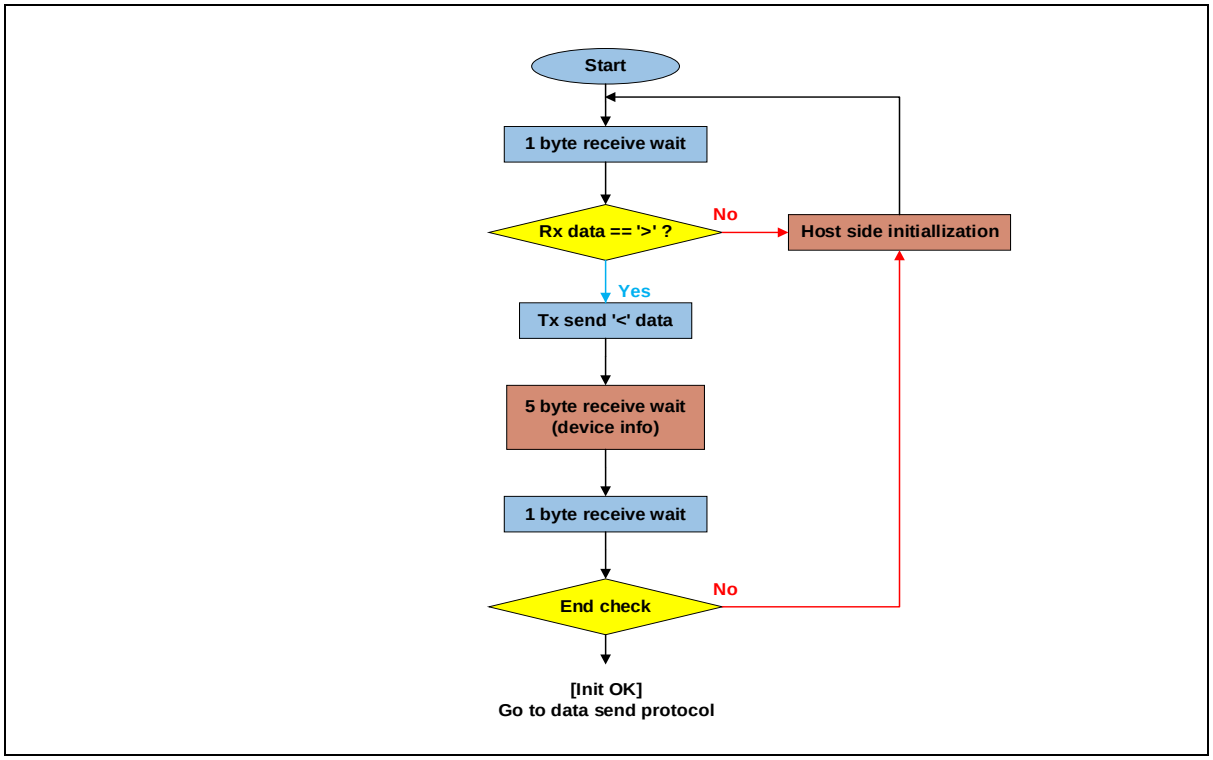
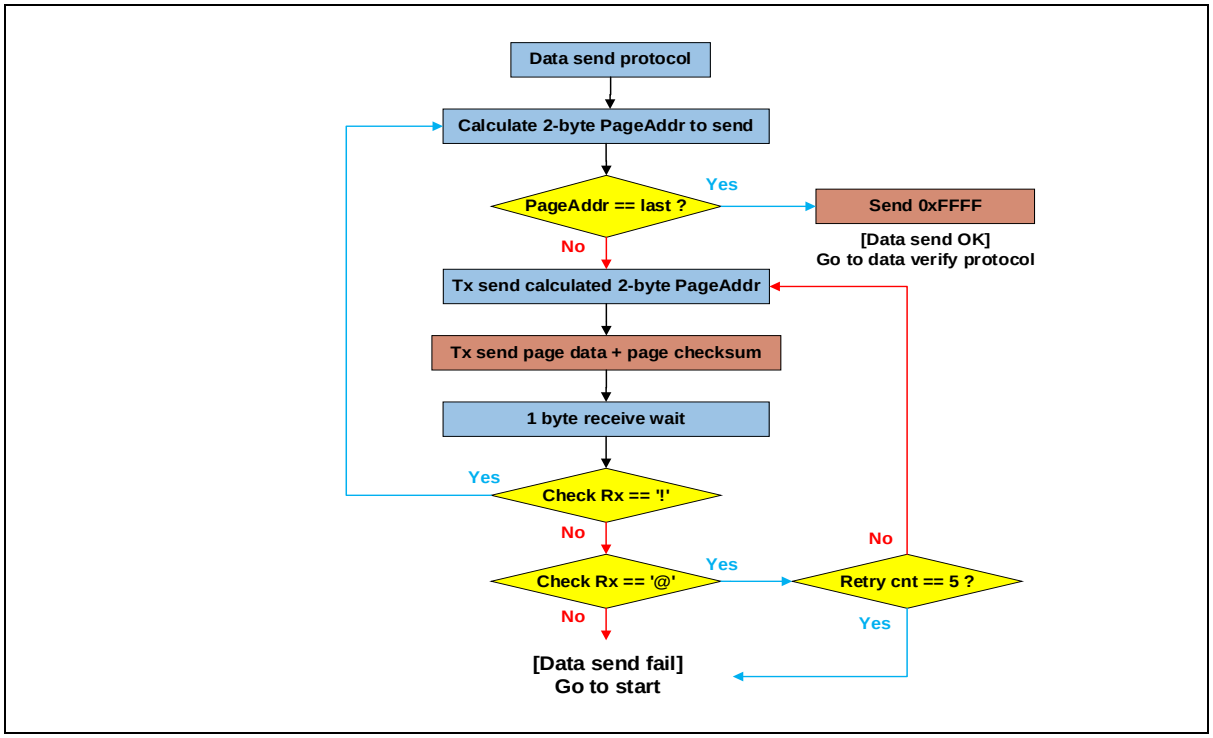


Figure 13. Data Transmission Protocol for Downloader Tool



Revision History

Revision	Date	Notes
1.00	Mar. 11, 2020	Initial release
1.01	Jan. 15, 2021	Added the target microcontroller, A31G123.
1.02	May. 11, 2021	Added a target microcontroller list. Added A34M418 and A33G527 boot info.
1.03	Aug. 3, 2021	Added the target microcontroller, A31T21x.
1.04	Nov. 7, 2022	Updated the template of this document.
1.05	Oct. 8, 2023	Added the target microcontroller, A33G53x. Added the A33G53x device to Table 1 and also added section 4.3.5.
1.06	Dec. 27, 2023	Added the target microcontroller, A34M420. Added the A34M420 device to Table 1 and also added section 0.
1.07	May. 27, 2024	Added the target microcontroller, A34L716. Added the A34L716 device to Table 1 and also added section 4.3.7.
1.08	Nov. 15, 2024	Updated the disclaimer.
1.09	Nov. 22, 2024	Added A34M456 device.

Korea

Regional Office, Seoul
R&D, Marketing & Sales
8th Fl., 330, Yeongdong-daero,
Gangnam-gu, Seoul,
06177, Korea

Tel: +82-2-2193-2200
Fax: +82-2-508-6903
www.abovsemi.com

HQ, Ochang
R&D, QA, and Test Center
37, Gangni 1-gil, Ochang-eup,
Cheongwon-gun,
Chungcheongbuk-do,28126, Korea

Tel: +82-43-219-5200
Fax: +82-43-217-3534
www.abovsemi.com

Domestic Sales Manager
Tel: +82-2-2193-2206
Fax: +82-2-508-6903
Email: sales_kr@abov.co.kr

Global Sales Manager
Tel: +82-2-2193-2281
Fax: +82-2-508-6903
Email: sales_gl@abov.co.kr

China Sales Manager
Tel: +86-755-8287-2205
Fax: +86-755-8287-2204
Email: sales_cn@abov.co.kr