BLE Application Note

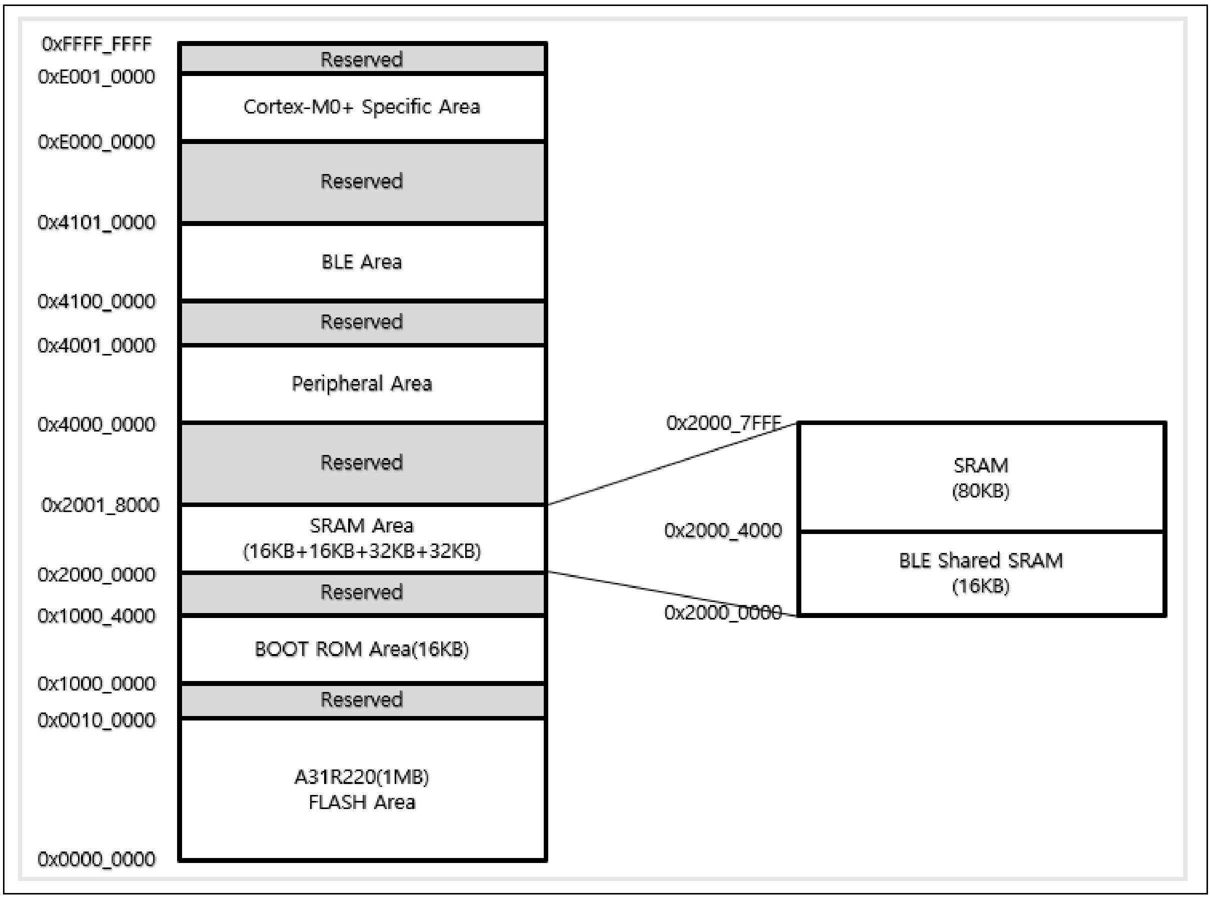
Executing Code from RAM

AN Rev. 1.01

Introduction

This application note explains how to execute user code from RAM. Executing user code in RAM instead of flash memory allows testing of user code on a device without changing the contents of flash memory.

Figure 1. Memory Map of A31R220



Contents

Introduction..1
1. RAM Image 4
    1.1 Configuring Project..4
    1.2 Executing User Code 6
2. Limitations 8
Revision History 9

List of Figures

Figure 1. Memory Map of A31R220 1
Figure 2: Target Options 4
Figure 3: Memory Layout Selection 5
Figure 4. RESET Section in the Map File 5
Figure 5: Initialization File Settings 6
Figure 6: Flash Tool Selection 7

1. RAM Image

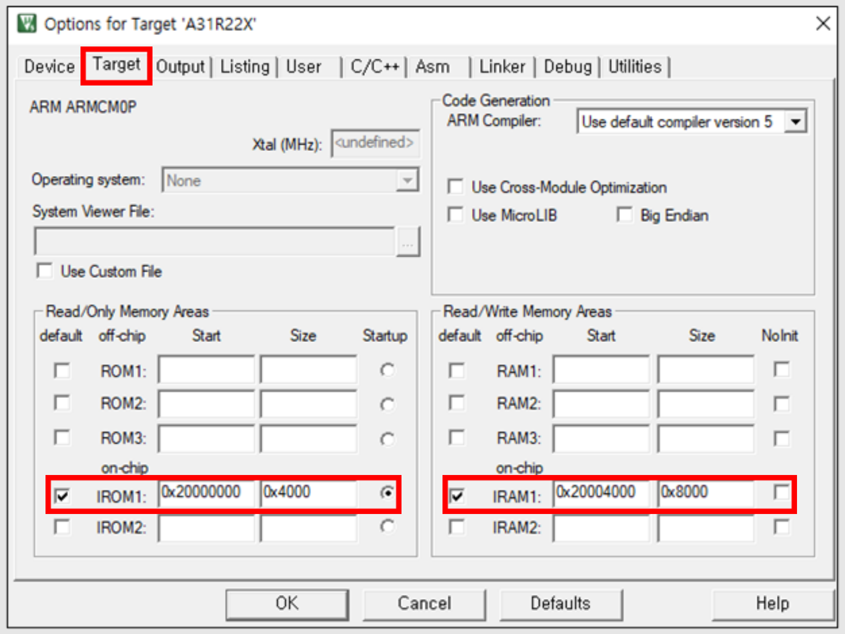
Before executing a user code from RAM, it is necessary to create an executable image of the user code for loading into RAM. In other words, you can execute the user code by making an executable image and loading it into RAM.

1.1 Configuring Project

To build a user code image to run from RAM, you must set up the area for user code and data in RAM when you create a project. To do this, follow the procedure below:

1. Open the **Options for Target** dialog box in the KEIL MDK-ARM.
2. On the **Target** tab, set the address configuration of the ROM and RAM according to the device you want to use. The following configuration is an example for A31R22x devices.
   - IROM1 Start: 0x20000000
   - IROM1 Size: 0x4000
   - IRAM1 Start: 0x20004000
   - IRAM1 Size: 0x8000

Figure 2: Target Options



Options for Target 'A31R22X'

ARM ARMCM0P

Xtal (MHz): <undefined>

Operating system: None

System Viewer File:

□ Use Custom File

Code Generation

ARM Compiler: Use default compiler version 5

□ Use Cross-Module Optimization

□ Use MicroLIB □ Big Endian

Read/Only Memory Areas

default	off-chip	Start	Size	Startup
□	ROM1:			○
□	ROM2:			○
□	ROM3:			○
	on-chip			
☑	IROM1:	0x20000000	0x4000	●
□	IROM2:			○

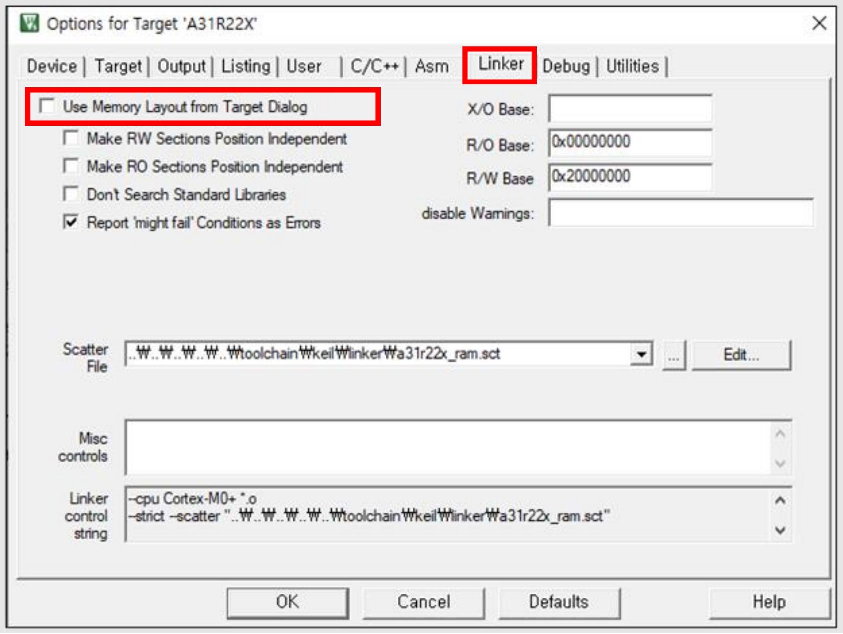
Read/Write Memory Areas

default	off-chip	Start	Size	NoInit
□	RAM1:			□
□	RAM2:			□
□	RAM3:			□
	on-chip			
☑	IRAM1:	0x20004000	0x8000	□
□	IRAM2:			□

If you want to use a scatter loading file to configure memory in detail, follow this procedure:

1. Open the **Options for Target** window in the KEIL MDK-ARM.
2. On the **Linker** tab, select the scatter loading file with the '**sct**' extension (for complex memory layouts).
   A. Uncheck the **Use Memory Layout from Target Dialog** checkbox.
   B. Select the specified scatter loading file in the **Scatter File** field.
      i. For example, if you are using the A31R220 device, select the '**a31r22x_ram.sct**' file from '**…\toolchain\keil\linker\**' folder if you use the A31R220 device.

Figure 3: Memory Layout Selection


Options for Target 'A31R22X'
Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities
OK
Cancel
Defaults
Help

□ Use Memory Layout from Target Dialog
□ Make RW Sections Position Independent
□ Make RO Sections Position Independent
□ Don't Search Standard Libraries
☑ Report 'might fail' Conditions as Errors

X/O Base:
R/O Base: 0x00000000
R/W Base 0x20000000
disable Warnings:

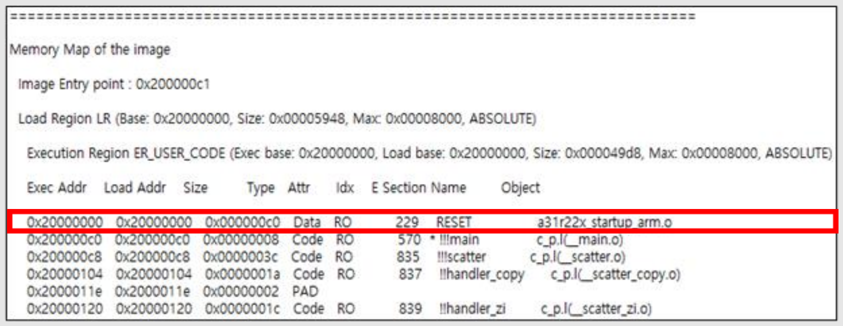
Scatter File: ..₩..₩..₩..₩..₩toolchain₩keil₩linker₩a31r22x_ram.sct ... Edit...

Misc controls:

Linker control string:
```
--cpu Cortex-M0+ *.o
--strict --scatter "..₩..₩..₩..₩..₩toolchain₩keil₩linker₩a31r22x_ram.sct"
```

3. Build the project with the settings shown above.
4. When the project building is completed, open the map file to verify that the image was built with the memory configuration. For example, the address of the RESET section must be located at 0x20000000 in RAM.

Figure 4. RESET Section in the Map File



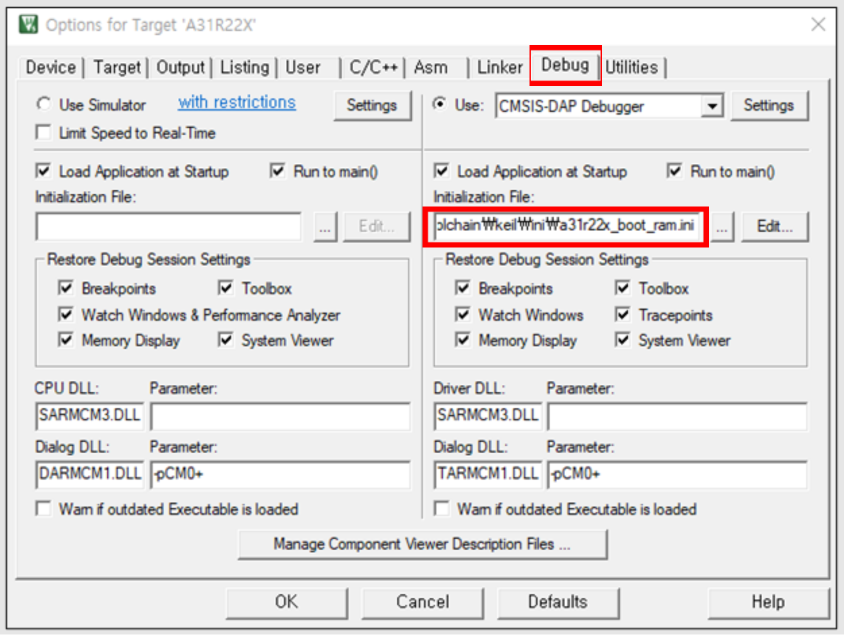
```
==============================================================================

Memory Map of the image

  Image Entry point : 0x200000c1

  Load Region LR (Base: 0x20000000, Size: 0x00005948, Max: 0x00008000, ABSOLUTE)

    Execution Region ER_USER_CODE (Exec base: 0x20000000, Load base: 0x20000000, Size: 0x000049d8, Max: 0x00008000, ABSOLUTE)

    Exec Addr    Load Addr    Size         Type   Attr      Idx    E Section Name        Object

    0x20000000   0x20000000   0x000000c0   Data   RO          229    RESET               a31r22x_startup_arm.o
    0x200000c0   0x200000c0   0x00000008   Code   RO          570  * !!!main             c_p.l(__main.o)
    0x200000c8   0x200000c8   0x0000003c   Code   RO          835    !!!scatter          c_p.l(__scatter.o)
    0x20000104   0x20000104   0x0000001a   Code   RO          837    !!handler_copy      c_p.l(__scatter_copy.o)
    0x2000011e   0x2000011e   0x00000002   PAD
    0x20000120   0x20000120   0x0000001c   Code   RO          839    !!handler_zi        c_p.l(__scatter_zi.o)
```

1.2 Executing User Code

To load and execute a user code from RAM, you must use the **Initialization File** to set the **PC** and **SP** register values to the correct addresses in RAM. To do this, follow this procedure:

1. Open the **Options for Target** dialog box in the KEIL MDK-ARM.
2. Go to the **Initialization File** field on the **Debug** tab and select the **Initialization File**.

Figure 5: Initialization File Settings



○ Use Simulator with restrictions

□ Limit Speed to Real-Time

☑ Load Application at Startup ☑ Run to main()

Initialization File:

Restore Debug Session Settings

☑ Breakpoints ☑ Toolbox

☑ Watch Windows & Performance Analyzer

☑ Memory Display ☑ System Viewer

CPU DLL: SARMCM3.DLL Parameter:

Dialog DLL: DARMCM1.DLL Parameter: -pCM0+

□ Warn if outdated Executable is loaded

● Use: CMSIS-DAP Debugger

☑ Load Application at Startup ☑ Run to main()

Initialization File: olchain₩keil₩ini₩a31r22x_boot_ram.ini

Restore Debug Session Settings

☑ Breakpoints ☑ Toolbox

☑ Watch Windows ☑ Tracepoints

☑ Memory Display ☑ System Viewer

Driver DLL: SARMCM3.DLL Parameter:

Dialog DLL: TARMCM1.DLL Parameter: -pCM0+

□ Warn if outdated Executable is loaded

The code below shows an example of an **Initialization File** named **a31r22x_boot_ram.ini**.

```
load %L

SP = _RDWORD(0x20000000);
PC = _RDWORD(0x20000004);
_WDWORD(0xE000ED08, 0x20000000);
xPSR=0xF1000000;
```

Each line in the **a31r22x_boot_ram.ini** file is described below:

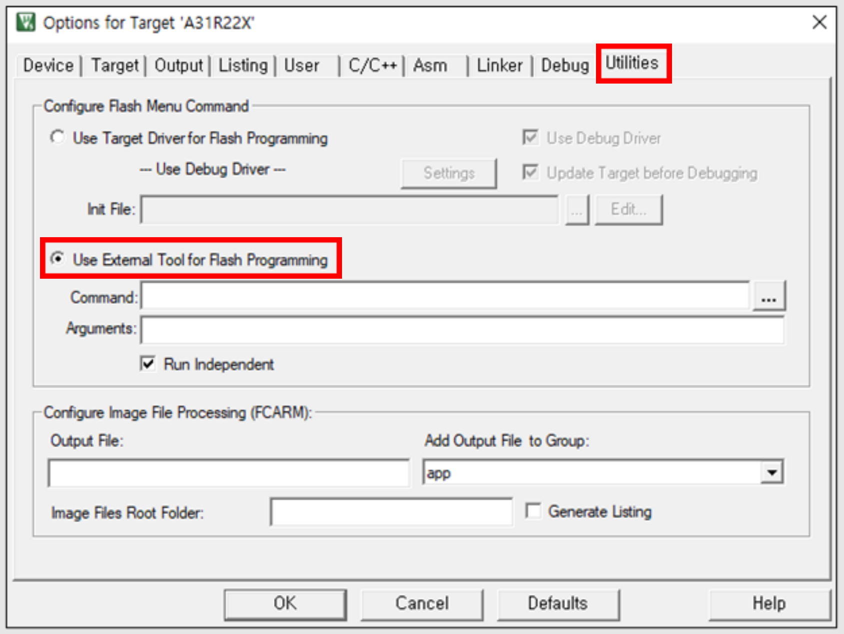
- load %L
  - Loads the project-built image.
- SP = 0x20000000
  - Sets the Stack Pointer register to 0x20000000.

- PC = 0x20000004
  - Sets the Program Counter register to 0x20000004.
- _WDWORD(0xE000ED08, 0x20000000);
  - Sets the Vector Table Offset Register (VTOR) to 0x20000000 so that all exceptions are processed correctly when an image is loaded to this address.
- xPSR=0xF1000000
  - Sets the **Thumb-2 mode** bit.

To prevent programming the flash memory when entering debug session, follow this procedure:

1. Open the **Options for Target** window in the KEIL MDK-ARM.
2. Go to the **Configure Flash Menu Command** item on the **Utilities** tab and select the **Use External Tool for Flash Programming**.

Figure 6: Flash Tool Selection



Options for Target 'A31R22X'

Configure Flash Menu Command

○ Use Target Driver for Flash Programming

☑ Use Debug Driver

--- Use Debug Driver ---

☑ Update Target before Debugging

Init File:

● Use External Tool for Flash Programming

Command:

Arguments:

☑ Run Independent

Configure Image File Processing (FCARM):

Output File:

Add Output File to Group: app

Image Files Root Folder:

□ Generate Listing

After setting all the options above, click the **Start Debug Session** to load the user code to the specified address in RAM and enter debug mode. In debug mode, you can use debugging functions to debug the user code running in RAM.

2. Limitations

Executing a user code from RAM has several limitations. User code must be configured and executed by considering the following limitations:

- Since RAM areas do not retain their contents permanently, executing code from RAM is available for testing purposes.
- Using a power restart or reset while running user code may cause a malfunction in user code operation.
- Due to the different access speeds between RAM and flash memory, it can cause differences in execution time.
- Executing a user code from RAM will reduce the available space for data in the same RAM area.

Revision History

Revision	Date	Notes
1.00	Sept. 1, 2023	Initial release.
1.01	Dec. 2, 2024	Updated the disclaimer.

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
93, 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

ABOV Disclaimer

IMPORTANT NOTICE – PLEASE READ CAREFULLY

ABOV Semiconductor ("ABOV") reserves the right to make changes, corrections, enhancements, modifications, and improvements to ABOV products and/or to this document at any time without notice. **ABOV DOES NOT GIVE WARRANTIES AS TO THE ACCURACY OR COMPLETENESS OF THE INFORMATION INCLUDED HEREIN.** Purchasers should obtain the latest relevant information of ABOV products before placing orders. Purchasers are entirely responsible for the choice, selection, and use of ABOV products and ABOV assumes no liability for application assistance or the design of purchasers' products. **NO LICENSE, EXPRESS OR IMPLIED, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY ABOV HEREIN. ABOV DISCLAIMES ALL EXPRESS AND IMPLIED WARRANTIES AND SHALL NOT BE RESPONSIBLE OR LIABLE FOR ANY INJUIRES OR DAMAGES RELATED TO USE OF ABOV PRODUCTS IN SUCH UNAUTHORIZED APPLICATIONS.** ABOV and the ABOV logo are trademarks of ABOV. For additional information about ABOV trademarks, please refer to https://www.abov.co.kr/en/about/corporate_identity.php. All other product or service names are the property of their respective owners. Information in this document supersedes and replaces the information previously supplied in any former versions of this document.