32-bit Microcontroller Application Note

IAR Project Setting Guide

AN Rev. 1.02

Introduction

This document is intended to help developers develop ABOV 32-bit microcontrollers (Arm Cortex-M0 / M0+ / M3 / M4) in an IAR EWARM integrated development environment.

This document describes how to prepare an IAR project with configuration files, how to set up the environment variables and the debugger, and how to compile and debug so that you can quickly set up an initial development environment.

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

Table 1. Applicable Devices

Base Product	Part Number
A31G21x	A31G213CL, A31G213SQ, A31G213KN, A31G213GR, A31G212CL, A31G212SQ, A31G212KN, A31G212GR
A31G22x	A31G226ML, A31G226MM, A31G226RM, A31G226RL, A31G226RL A31G224ML, A31G224MM, A31G224RM, A31G224RL, A31G224RL
A31G31x	A31G316MM, A31G316ML, A31G316RM, A31G316RL, A31G314MM, A31G314ML, A31G314RM, A31G314RL, A31G314CL, A31G314CU, A31G314SN, A31G313RM, A31G313RL, A31G313CL, A31G313CU, A31G313SN
A31G32x	A31G324RL, A31G324CL, A31G324CU, A31G323RL, A31G323CL, A31G323CU
A33G52x	A33G527VQ, A33G527VL, A33G527MM, A33G527RL, A33G527ML, A33G527RM, A33G526VQ, A33G526VL, A33G526MM, A33G526RL, A33G526ML, A33G526RM, A33G524MM, A33G524RL, A33G524ML, A33G524RM
A33G53x	A33G539VQ, A33G539VL, A33G539MM, A33G539RL, A33G538VQ, A33G538VL, A33G538MM, A33G538RL
A31M22x	A31M223CL, A31M223KN
A33M11x	A33M116RL, A33M116RM, A33M116CL, A33M114RL, A33M114CL
A34M41x	A34M418YL, A34M418VL, A34M418RL, A34M416VL, A34M416RL, A34M414VL, A34M414RL
A34M420	A34M420YL, A34M418VL
A34L716	A34L716VL, A34L716RL

Reference Document

The following documents are available on www.abovsemi.com.

- Reference software URL in IAR EW IDE Project Management and Building Guide
- User's Manual, Datasheet, Application Note for products in Table 1
- Reference software package for products in Table 1

Contents

List of Tables

List of Figures

1. IAR Project Preparation

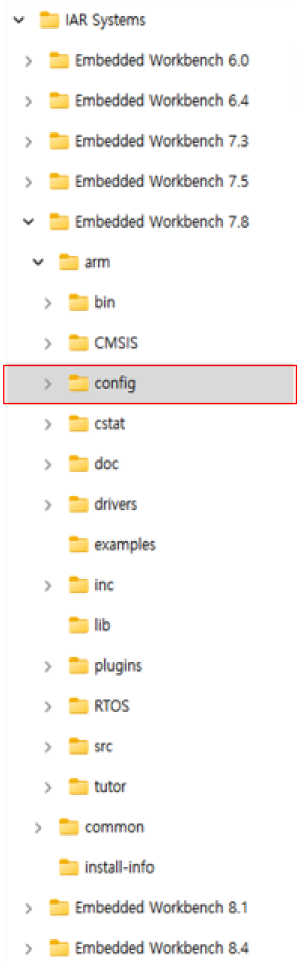
1.1 IAR Embedded Workbench for Arm Installation Path

The device's configuration files must be added to the IAR Embedded Workbench for Arm (hereafter called 'EWARM') installation path before using a project in the IAR EWARM environment.

The paths used in the process of copying and pasting configuration files are shown below:

- IAR installation path (Typical): ~\IAR Systems\Embedded Workbench X.Y (X.Y = Version)
  - The IAR EWARM installation path and the version may vary depending on the installed version of the IAR Embedded Workbench.
- The path where the product's configuration files are stored: ~\IAR Systems\Embedded Workbench X.Y**config**

Figure 1. IAR EWARM Installation Path



```
IAR Systems
  Embedded Workbench 6.0
  Embedded Workbench 6.4
  Embedded Workbench 7.3
  Embedded Workbench 7.5
  Embedded Workbench 7.8
    arm
      bin
      CMSIS
      config
      cstat
      doc
      drivers
      examples
      inc
      lib
      plugins
      RTOS
      src
      tutor
    common
    install-info
  Embedded Workbench 8.1
  Embedded Workbench 8.4
```

1.2 How to Register Product Configuration Files

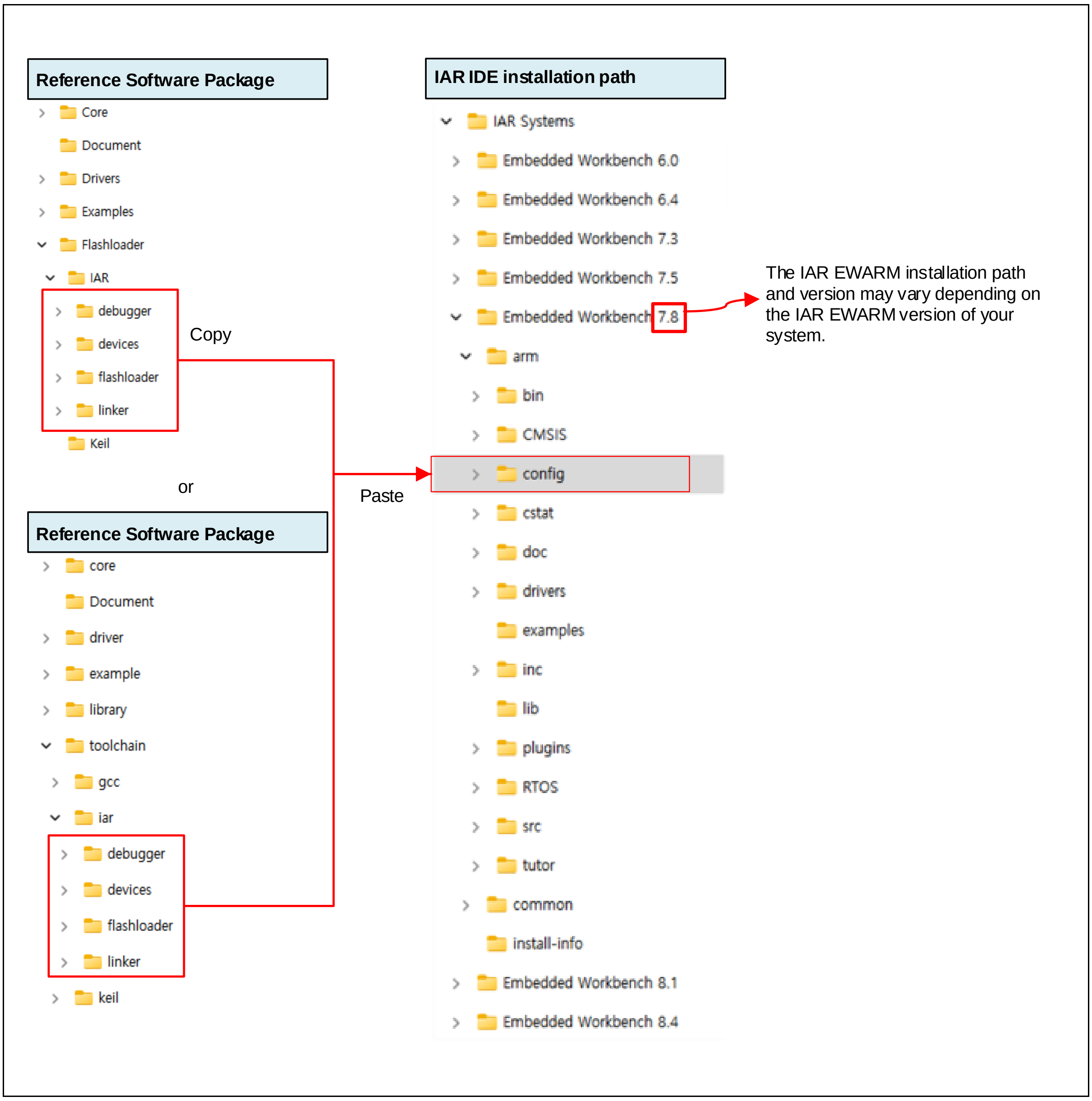
ABOV 32-bit Reference Software Packages include product configuration files that enable programming and debugging of the firmware in the IAR Embedded Workbench for the Arm environment.

Figure 2 shows the process of copying the folders containing configuration files from the ABOV 32-bit reference software package and pasting them under the 'config' directory in the IAR EWARM installation path.

NOTE:

1. Suppose a product configuration file already exists in the 'config' folder. In that case, the existing file remains unchanged, or if you add each configuration file (stored in the debugger, devices, flashloader, and linker directories) to the 'config' directory, the existing configuration file will be overwritten.

Figure 2. Copying Configuration Folders



The IAR EWARM installation path and version may vary depending on the IAR EWARM version of your system.

1.2.1 IAR EARM Configuration Files

IAR configuration files are located in the path below in the **Reference Software Package**:

- /Flashloader/iar/ or /toolchain/iar/

Table 2 and Table 3 show the directory names where the configuration files have to be installed in the IAR installation path and the list of the configuration files for the target device.

Follow the procedure below to complete the copy-and-paste process:

1. Copy the folders that contain the configuration files provided by **Reference Software Package**.
2. Move into the 'config' folder in your IAR EW for the Arm installation path.
3. Paste the copied folders in the '**config**' folder.

Table 2. Configuration Files for Target Device (ex. A34M420)

Directory Name	Configuration File
debugger/ABOV	a34m420.svd, a34m420.ddf
devices/ABOV	a34m420.i79, a34m420.menu
flashloader/ABOV	FlashA34M420.board, FlashA34M420.flash, FlashA34M420.out, FlashA34M420.mac(1)
linker/ABOV	a34m420.icf

NOTE:

1. Some devices that need to use Core reset require a mac file to reflect changes made after Flash Write in the user code (see Table 3).

Table 3. Target Devices and Flashloader Files

Target Device	Configuration File	Code Flash Area for Algorithm	RAM Area for Algorithm
A31G213 (64 KB)	FlashA31G213.board, FlashA31G213.flash, FlashA31G213.out	0x0000_0000 ~ 0x0000_FFFF (64 KB – Size: 0x10000)	0x2000_0000 ~ 0x2000_17FF (6 KB – Size: 0x1800)
A31G212 (32 KB)	FlashA31G212.board, FlashA31G212.flash, FlashA31G212.out	0x0000_0000 ~ 0x0000_7FFF (32 KB – Size: 0x8000)	
A31G226 (256 KB)	FlashA31G226.board, FlashA31G226.flash, FlashA31G226.out	0x0000_0000 ~ 0x0003_FFFF (256 KB – Size: 0x40000)	0x2000_0000 ~ 0x2000_4FFF (20 KB – Size: 0x5000)
A31G224 (128 KB)	FlashA31G224.board, FlashA31G224.flash, FlashA31G224.out	0x0000_0000 ~ 0x0001_FFFF (128 KB – Size: 0x20000)	
A31G316 (256 KB)	FlashA31G316.board, FlashA31G316.flash, FlashA31G316.out	0x0000_0000 ~ 0x0003_FFFF (256 KB – Size: 0x40000)	0x2000_0000 ~ 0x2000_3FFF (16 KB – Size: 0x4000)
A31G314 (128 KB)	FlashA31G314.board, FlashA31G314.flash, FlashA31G314.out	0x0000_0000 ~ 0x0001_FFFF (128 KB – Size: 0x20000)	
A31G313 (64 KB)	FlashA31G313.board, FlashA31G313.flash, FlashA31G313.out	0x0000_0000 ~ 0x0000_FFFF (64 KB – Size: 0x10000)	
A31G324 (128 KB)	FlashA31G324.board, FlashA31G324.flash, FlashA31G324.out	0x0000_0000 ~ 0x0001_FFFF (128 KB – Size: 0x20000)	0x2000_0000 ~ 0x2000_3FFF (16 KB – Size: 0x4000)
A31G323 (64 KB)	FlashA31G323.board, FlashA31G323.flash, FlashA31G323.out	0x0000_0000 ~ 0x0000_FFFF (64 KB – Size: 0x10000)	
A33G527 (384 KB)	FlashA33G527.board, FlashA33G527.flash, FlashA33G527.out	0x0000_0000 ~ 0x0005_FFFF (384 KB – Size: 0x60000)	0x2000_0000 ~ 0x2000_5FFF (24 KB – Size: 0x6000)
A33G526 (256 KB)	FlashA33G526.board, FlashA33G526.flash, FlashA33G526.out	0x0000_0000 ~ 0x0003_FFFF (256 KB – Size: 0x40000)	
A33G524 (128 KB)	FlashA33G524.board, FlashA33G524.flash, FlashA33G524.out	0x0000_0000 ~ 0x0001_FFFF (128 KB – Size: 0x20000)	
A33G539 (768 KB)	FlashA33G539.board, FlashA33G539.flash, FlashA33G539.out	0x0000_0000 ~ 0x000B_FFFF (768 KB – Size: 0xC0000)	0x2000_0000 ~ 0x2000_5FFF (24 KB – Size: 0x6000)
A33G538 (512 KB)	FlashA33G538.board, FlashA33G538.flash, FlashA33G538.out	0x0000_0000 ~ 0x0007_FFFF (512 KB – Size: 0x80000)	
A31M223 (64 KB)	FlashA31M223.board, FlashA31M223.flash, FlashA31M223.out	0x0000_0000 ~ 0x0000_FFFF (64 KB – Size: 0x10000)	0x2000_0000 ~ 0x2000_1FFF (8 KB – Size: 0x2000)
A33M116 (256 KB)	a33m116_flash.board, a33m116_flash.flash, a33m116_flash.out	0x0000_0000 ~ 0x0003_FFFF (256 KB – Size: 0x40000)	0x2000_0000 ~ 0x2000_3FFF (16 KB – Size: 0x4000)
A33M114 (128 KB)	a33m114_flash.board, a33m114_flash.flash, a33m114_flash.out	0x0000_0000 ~ 0x0001_FFFF (128 KB – Size: 0x20000)	
A31M418 (512 KB)	FlashA34M418.board, FlashA34M418.flash, FlashA34M418.out, FlashA34M418.mac	0x0000_0000 ~ 0x0007_FFFF (512 KB – Size: 0x80000)	0x2000_0000 ~ 0x2000_FFFF (64 KB – Size: 0x10000)
A31M416 (256 KB)	FlashA34M416.board, FlashA34M416.flash, FlashA34M416.out, FlashA34M416.mac	0x0000_0000 ~ 0x0003_FFFF (256 KB – Size: 0x40000)	
A31M414 (128 KB)	FlashA34M414.board, FlashA34M414.flash, FlashA34M414.out, FlashA34M414.mac	0x0000_0000 ~ 0x0001_FFFF (128 KB – Size: 0x20000)	0x2000_0000 ~ 0x2000_7FFF (32 KB – Size: 0x8000)
A31M420 (1024 KB)	FlashA34M420.board, FlashA34M420.flash, FlashA34M420.out, FlashA34M420.mac	0x0000_0000 ~ 0x000F_FFFF (1024 KB – Size: 0x100000)	0x2000_0000 ~ 0x2000_FFFF (64 KB – Size: 0x10000)

Table 3. Target Devices and Flashloader Files (continued)

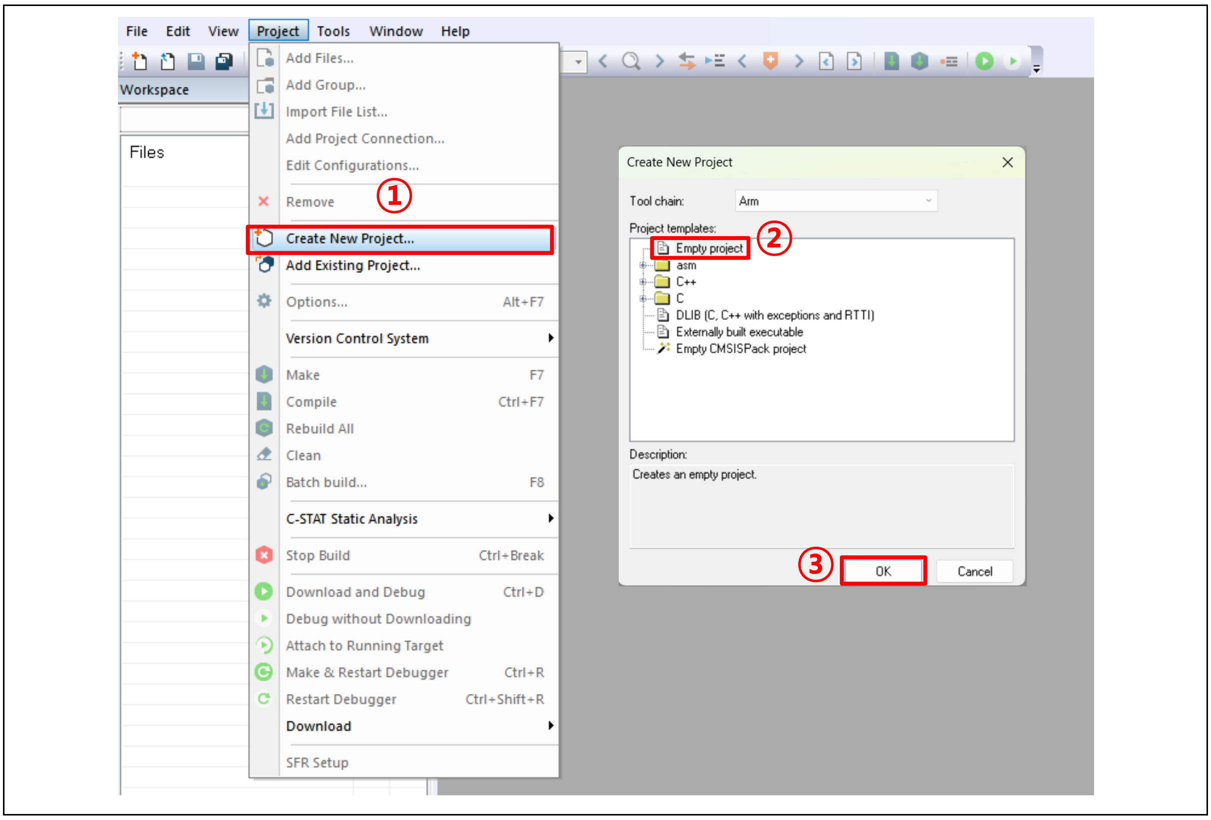
Target Device	Configuration File	Code Flash Area for Algorithm	RAM Area for Algorithm
A34L716 (256 KB)	FlashA34L716.board, FlashA34L716.flash, FlashA34L716.out, FlashA34L716.mac	0x0000_0000 ~ 0x0003_FFFF (256 KB – Size: 0x40000)	0x2000_0000 ~ 0x2000_5FFF (24 KB – Size: 0x6000)

2. Create New IAR EWARM Project

Once you have completed copying and pasting the product configuration files from the ABOV reference software package into the IAR installation path, follow the procedure below to create a new IAR project:

1. Select the '**Create New Project…**' in the Project menu.
2. Select the '**Empty project**' in the '**Create New Project**' dialog box.
3. Press the **OK** button to complete the project creation.

Figure 3. New IAR EWARM Project Menu



Tool chain: Arm

Project templates: Empty project

Description: Creates an empty project.

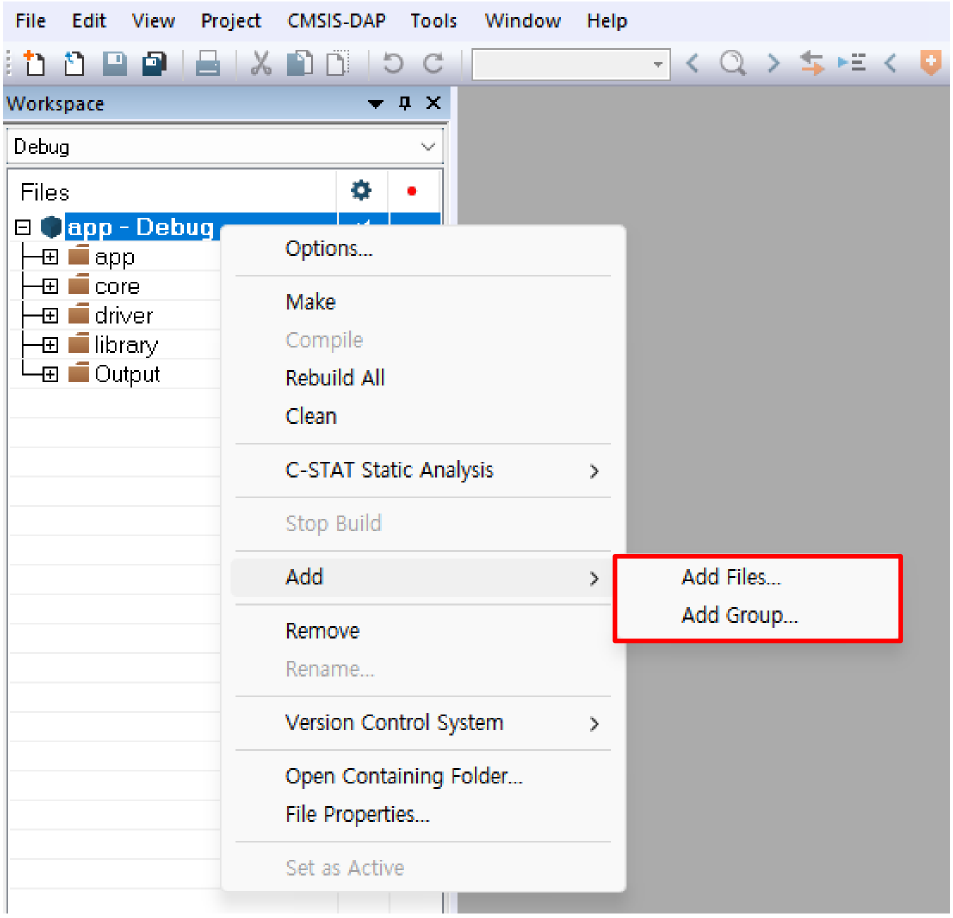
3. Add Group and Source Code in Workspace

You can add the source code from each directory of the downloaded software package to the newly created project by following the procedure below:

1. Right-click where you want to add a group in the Workspace pane.
2. Click the **Add > Add Group…** from the drop-down menu to add directories.
3. Create and name each group mapped to the software package's directory structure.
4. Click the **Add > Add Files…** from the drop-down menu and select source code files from each software package directory to add the source code files into each group.

Figure 4 shows an example of configuring groups and adding source codes to the IAR project in the Workspace pane.

Figure 4. Add Group… and Add Files… Menus

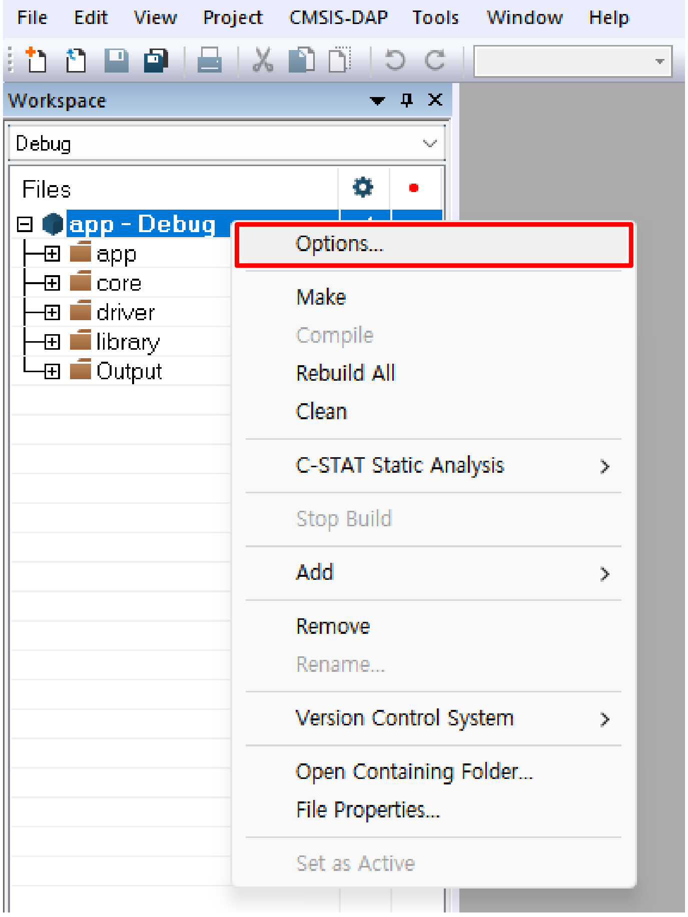


4. How to Set Up IAR EWARM

Once you have completed the instructions described in the previous chapters, you can set the environment options for building and debugging the IAR project in the Workspace in this chapter.

Right-clicking in the Workspace and selecting the ‘**Options…**’ from the drop-down menu will open the ‘**Options for node**’ window, described in the following sections.

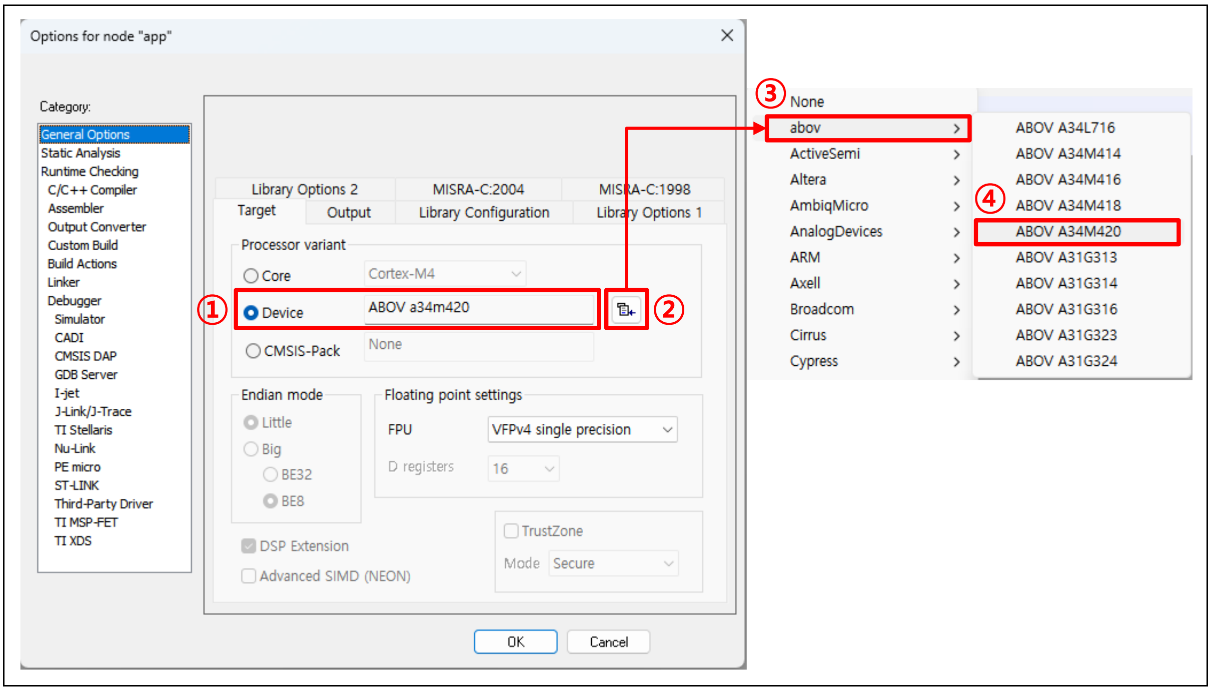
Figure 5. Options… Menu



4.1 General Options

Select the target device in the Device field of the Processor variant section on the Target tab of the General Options. (If you did not add the product configuration files described in Chapter 3, the product name may not appear.)

Figure 6. Device Selection on the Target Tab of General Options



Options for node "app"

Processor variant

○ Core Cortex-M4

● Device ABOV a34m420

○ CMSIS-Pack None

Endian mode

● Little

○ Big

○ BE32

● BE8

Floating point settings

FPU VFPv4 single precision

D registers 16

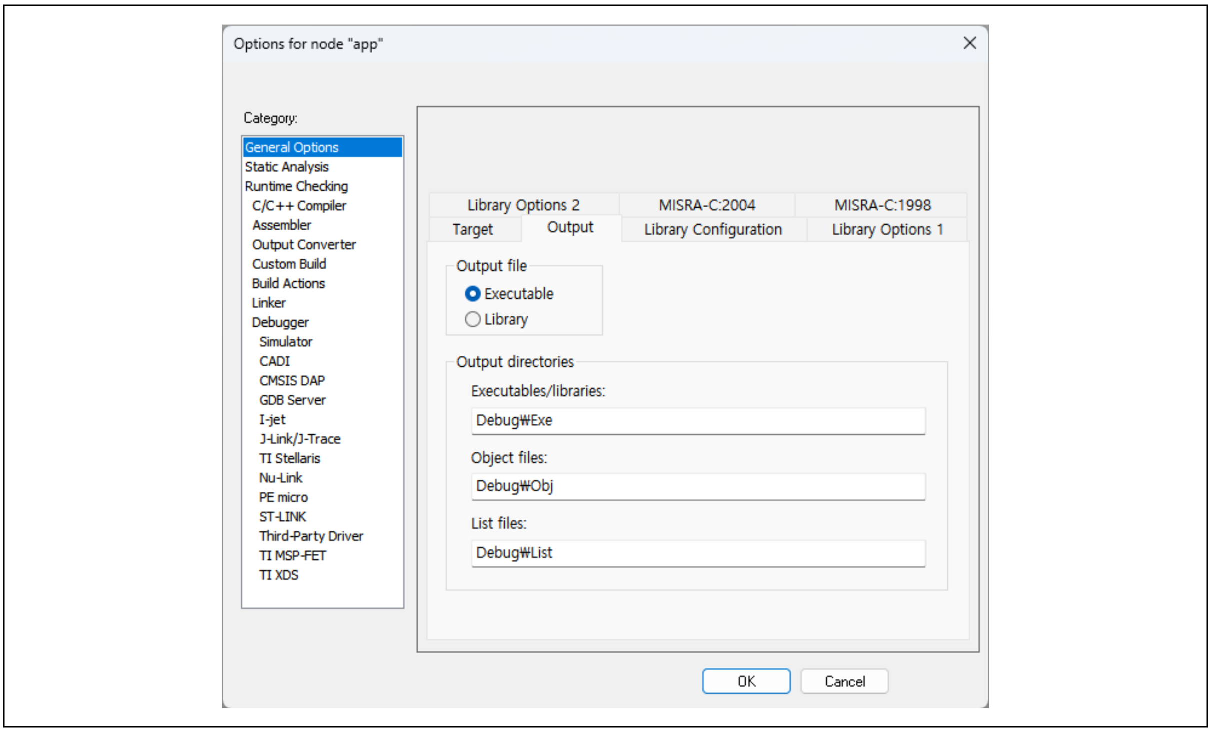
☑ DSP Extension

□ Advanced SIMD (NEON)

□ TrustZone

Mode Secure

Figure 7. Options on the Output Tab of General Options



Options for node "app"

Output file

● Executable

○ Library

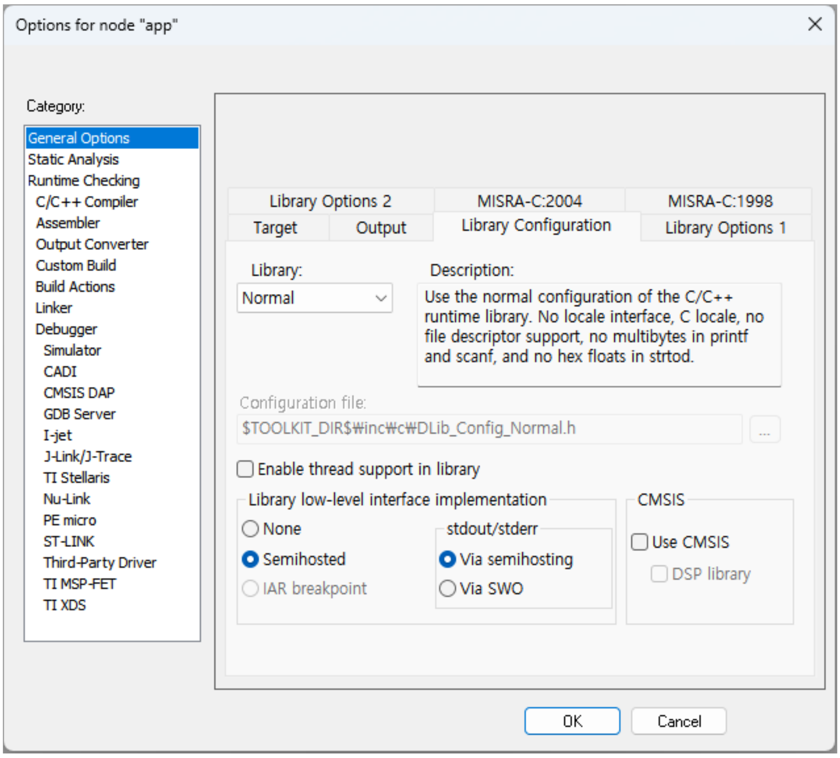
Output directories

Executables/libraries: Debug₩Exe

Object files: Debug₩Obj

List files: Debug₩List

Figure 8. Options on the Library Configuration Tab of General Options



Library: Normal

Description:
Use the normal configuration of the C/C++ runtime library. No locale interface, C locale, no file descriptor support, no multibytes in printf and scanf, and no hex floats in strtod.

Configuration file:
$TOOLKIT_DIR$₩inc₩c₩DLib_Config_Normal.h

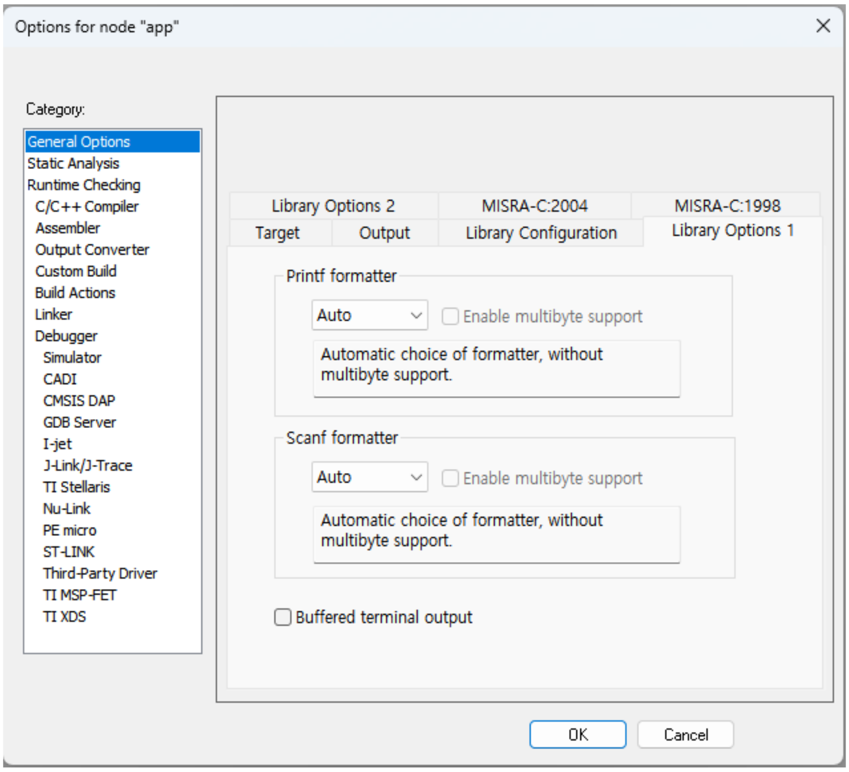
□ Enable thread support in library

Library low-level interface implementation
○ None
● Semihosted
○ IAR breakpoint

stdout/stderr
● Via semihosting
○ Via SWO

CMSIS
□ Use CMSIS
□ DSP library

Figure 9. Options on the Library Options Tab of General Options

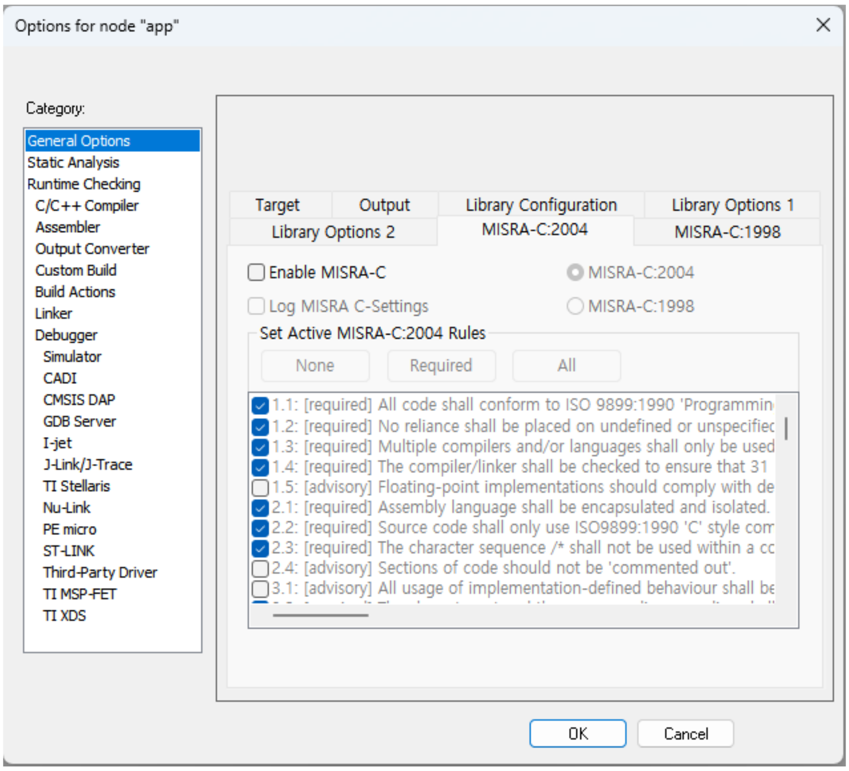


Printf formatter
Auto □ Enable multibyte support
Automatic choice of formatter, without multibyte support.

Scanf formatter
Auto □ Enable multibyte support
Automatic choice of formatter, without multibyte support.

□ Buffered terminal output

Figure 10. Options on the MISRA-C:2004 Tab of General Options



☐ Enable MISRA-C

● MISRA-C:2004

☐ Log MISRA C-Settings

○ MISRA-C:1998

Set Active MISRA-C:2004 Rules

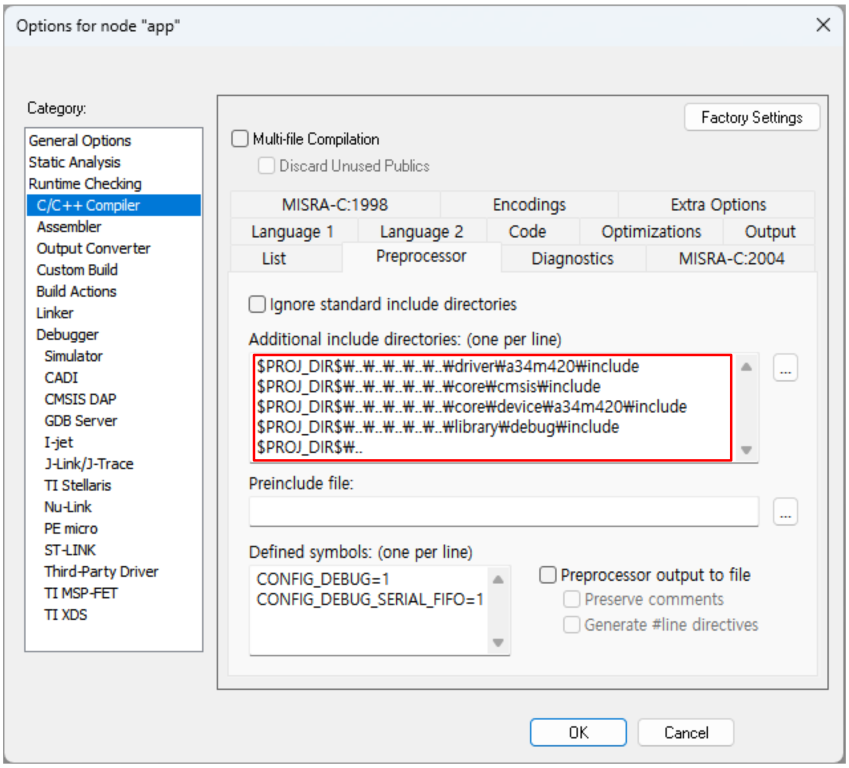
☑ 1.1: [required] All code shall conform to ISO 9899:1990 'Programmin
☑ 1.2: [required] No reliance shall be placed on undefined or unspecifiec
☑ 1.3: [required] Multiple compilers and/or languages shall only be used
☑ 1.4: [required] The compiler/linker shall be checked to ensure that 31
☐ 1.5: [advisory] Floating-point implementations should comply with de
☑ 2.1: [required] Assembly language shall be encapsulated and isolated.
☑ 2.2: [required] Source code shall only use ISO9899:1990 'C' style com
☑ 2.3: [required] The character sequence /* shall not be used within a cc
☐ 2.4: [advisory] Sections of code should not be 'commented out'.
☐ 3.1: [advisory] All usage of implementation-defined behaviour shall be

4.2 C/C++ Compiler

This section describes how to set the include path for the header files. The header files are required when building the source code added in the Workspace. In addition, this section describes how to set the predefined symbols.

As shown in Figure 11, you can add the reference directories on the **Preprocessor tab**, the **C/C ++ Compiler** submenu. The include directory paths in the Workspace can be defined using the IAR IDE's argument variables shown in Table 4.

Figure 11. Options on the Preprocessor Tab of C/C++ Compiler



□ Multi-file Compilation

□ Discard Unused Publics

□ Ignore standard include directories

Additional include directories: (one per line)

```
$PROJ_DIR$₩..₩..₩..₩..₩..₩driver₩a34m420₩include
$PROJ_DIR$₩..₩..₩..₩..₩..₩core₩cmsis₩include
$PROJ_DIR$₩..₩..₩..₩..₩..₩core₩device₩a34m420₩include
$PROJ_DIR$₩..₩..₩..₩..₩..₩library₩debug₩include
$PROJ_DIR$₩..
```

Preinclude file:

Defined symbols: (one per line)

```
CONFIG_DEBUG=1
CONFIG_DEBUG_SERIAL_FIFO=1
```

□ Preprocessor output to file

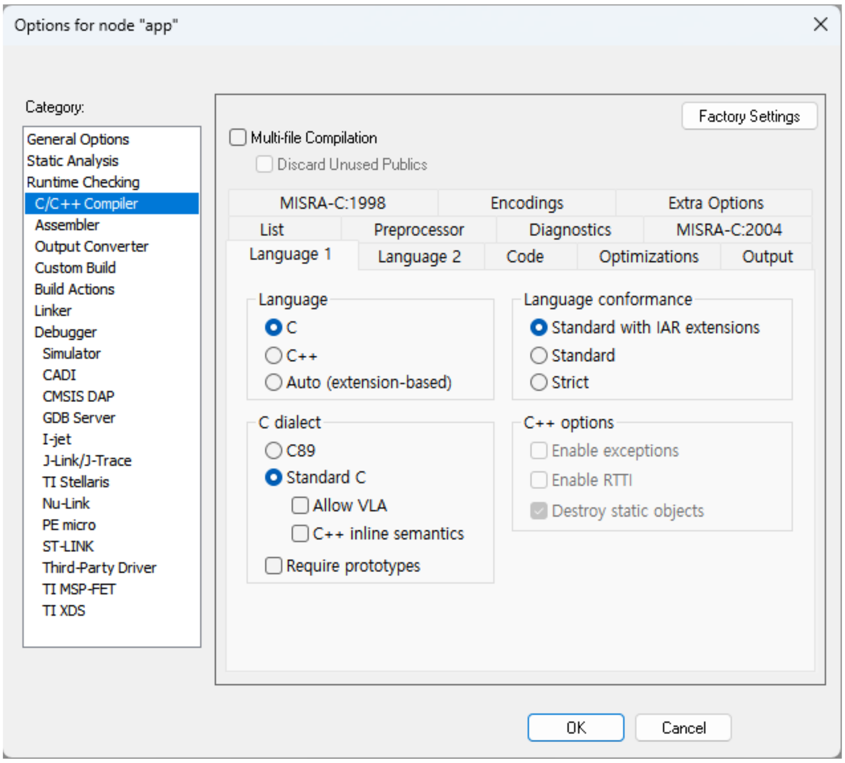
□ Preserve comments

□ Generate #line directives

Table 4. Path Argument Variables of IAR IDE on Preprocessor Tab

Path	Description
$TOOLKIT_DIR$	The path where IAR Embedded Workbench is installed.
$PROJ_DIR$	The path where the corresponding IAR EWARM project (*.eww) file is stored.

Figure 12. Options on the Language 1 Tab of C/C++ Compiler



□ Multi-file Compilation
□ Discard Unused Publics

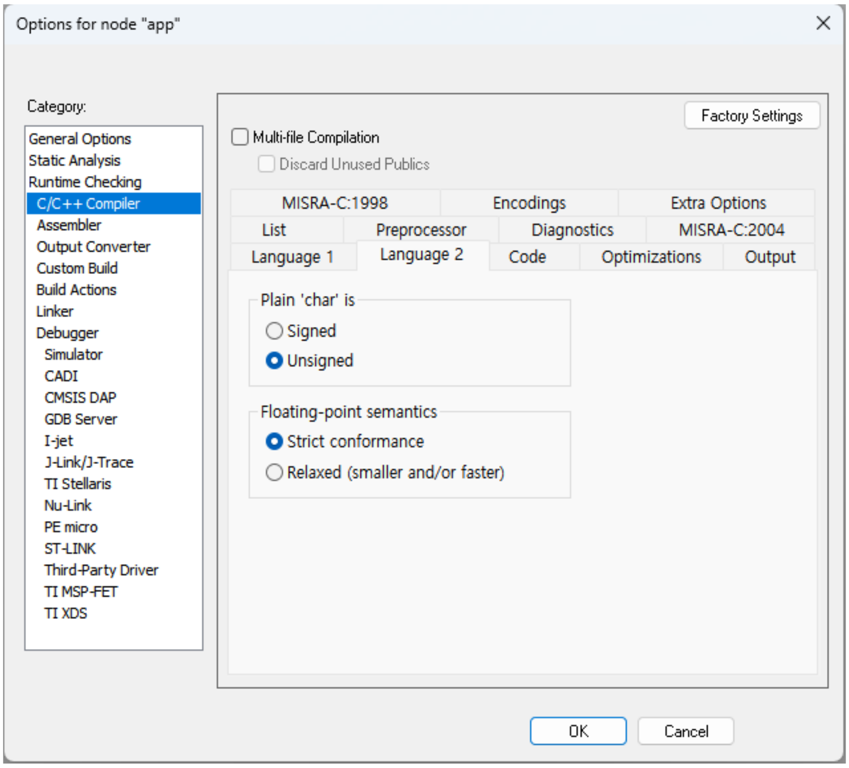
Language
● C
○ C++
○ Auto (extension-based)

Language conformance
● Standard with IAR extensions
○ Standard
○ Strict

C dialect
○ C89
● Standard C
□ Allow VLA
□ C++ inline semantics
□ Require prototypes

C++ options
□ Enable exceptions
□ Enable RTTI
☑ Destroy static objects

Figure 13. Options on the Language 2 Tab of C/C++ Compiler

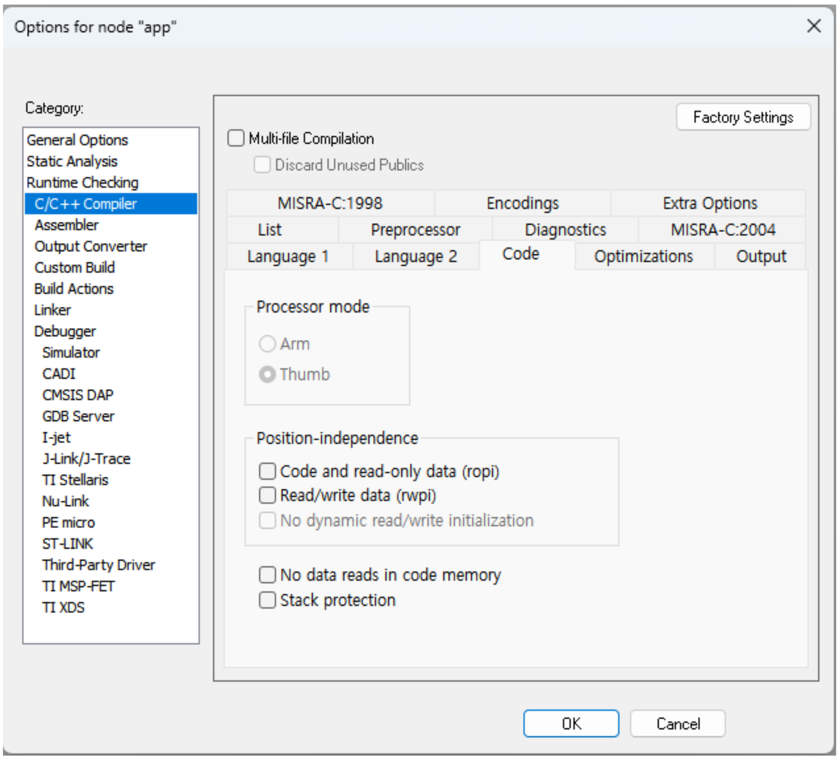


□ Multi-file Compilation
□ Discard Unused Publics

Plain 'char' is
○ Signed
● Unsigned

Floating-point semantics
● Strict conformance
○ Relaxed (smaller and/or faster)

Figure 14. Options on the Code Tab of C/C++ Compiler



Options for node "app"

Category: C/C++ Compiler

□ Multi-file Compilation

□ Discard Unused Publics

Processor mode

○ Arm

● Thumb

Position-independence

□ Code and read-only data (ropi)

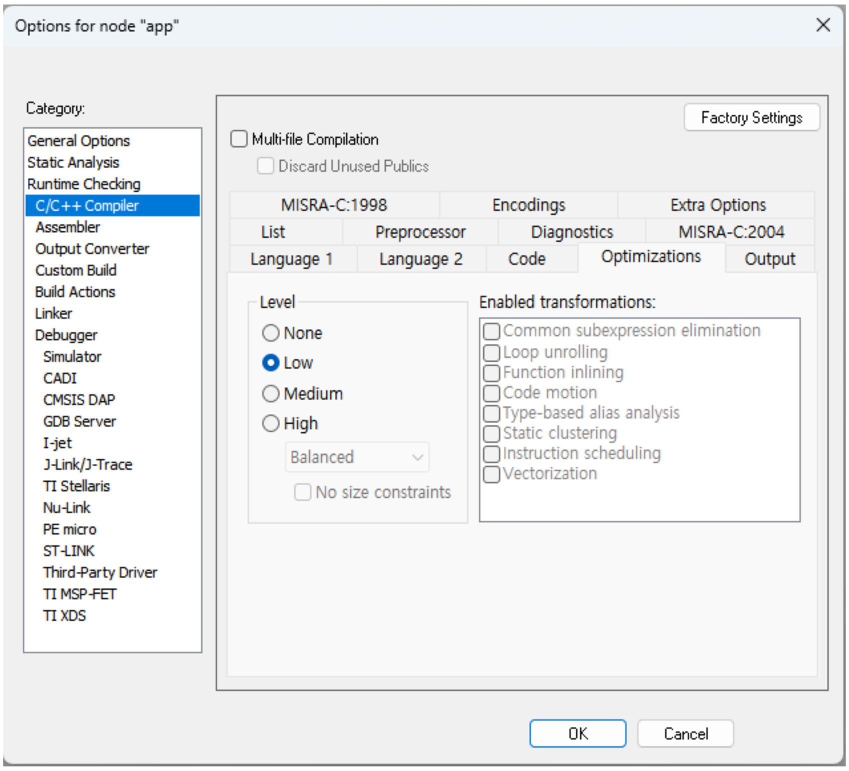
□ Read/write data (rwpi)

□ No dynamic read/write initialization

□ No data reads in code memory

□ Stack protection

Figure 15. Options on the Optimizations Tab of C/C++ Compiler



Options for node "app"

Category: C/C++ Compiler

□ Multi-file Compilation

□ Discard Unused Publics

Level

○ None

● Low

○ Medium

○ High

Balanced

□ No size constraints

Enabled transformations:

□ Common subexpression elimination

□ Loop unrolling

□ Function inlining

□ Code motion

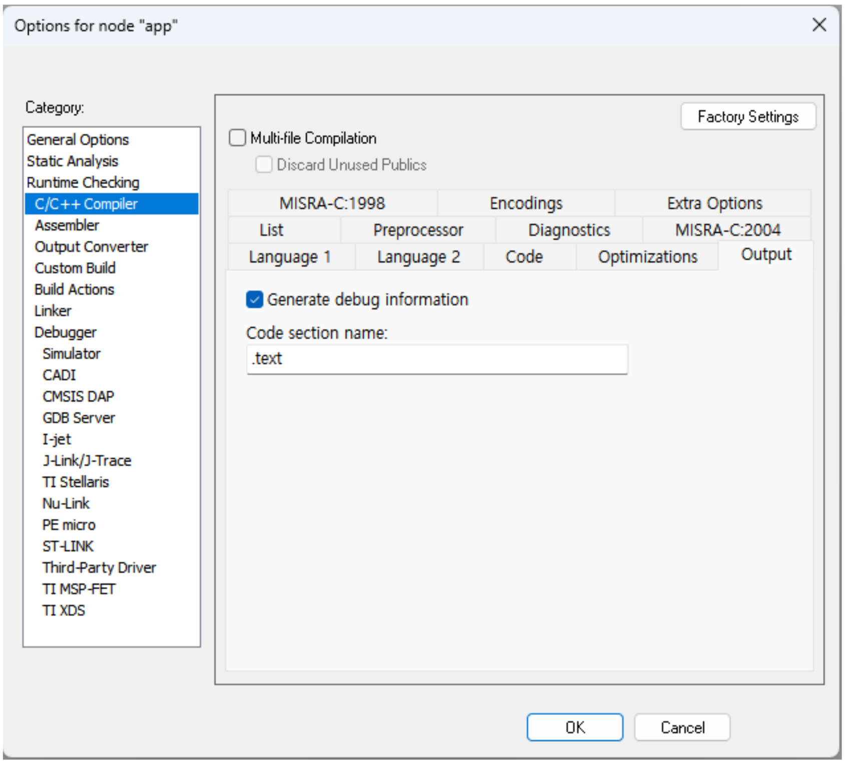
□ Type-based alias analysis

□ Static clustering

□ Instruction scheduling

□ Vectorization

Figure 16. Options on the Output Tab of C/C++ Compiler



□ Multi-file Compilation

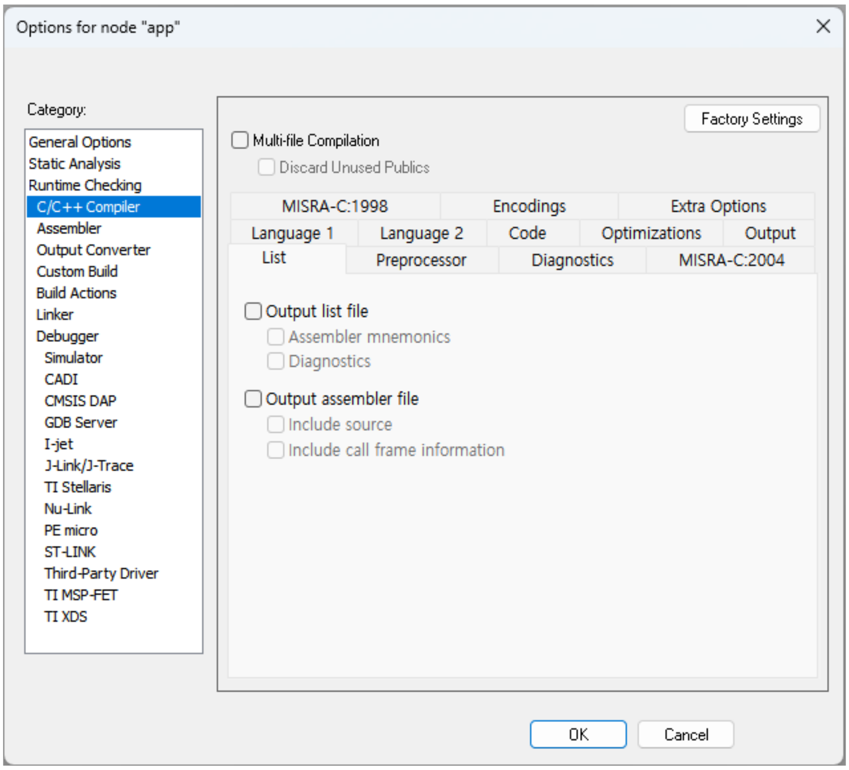
□ Discard Unused Publics

☑ Generate debug information

Code section name:

.text

Figure 17. Options on the List Tab of C/C++ Compiler



□ Multi-file Compilation

□ Discard Unused Publics

□ Output list file

□ Assembler mnemonics

□ Diagnostics

□ Output assembler file

□ Include source

□ Include call frame information

4.3 Linker

To add source code to the project, double-click on

You can set the Linker options referred to when building the IAR project by following the procedure below:

1. Select the **Linker** in the **Category** field, then select the **Config** tab.
2. Check the **Override default** checkbox in the **Linker configuration file field**.
   - A. Adding the product configuration file described in Chapter 3 shows that the '***.icf**' path is automatically set in this field. The ***.icf** file configures the target device's **Vector Table**, **Memory Regions**, and **Stack/Heap Sizes**.
   - B. **Memory Regions**:

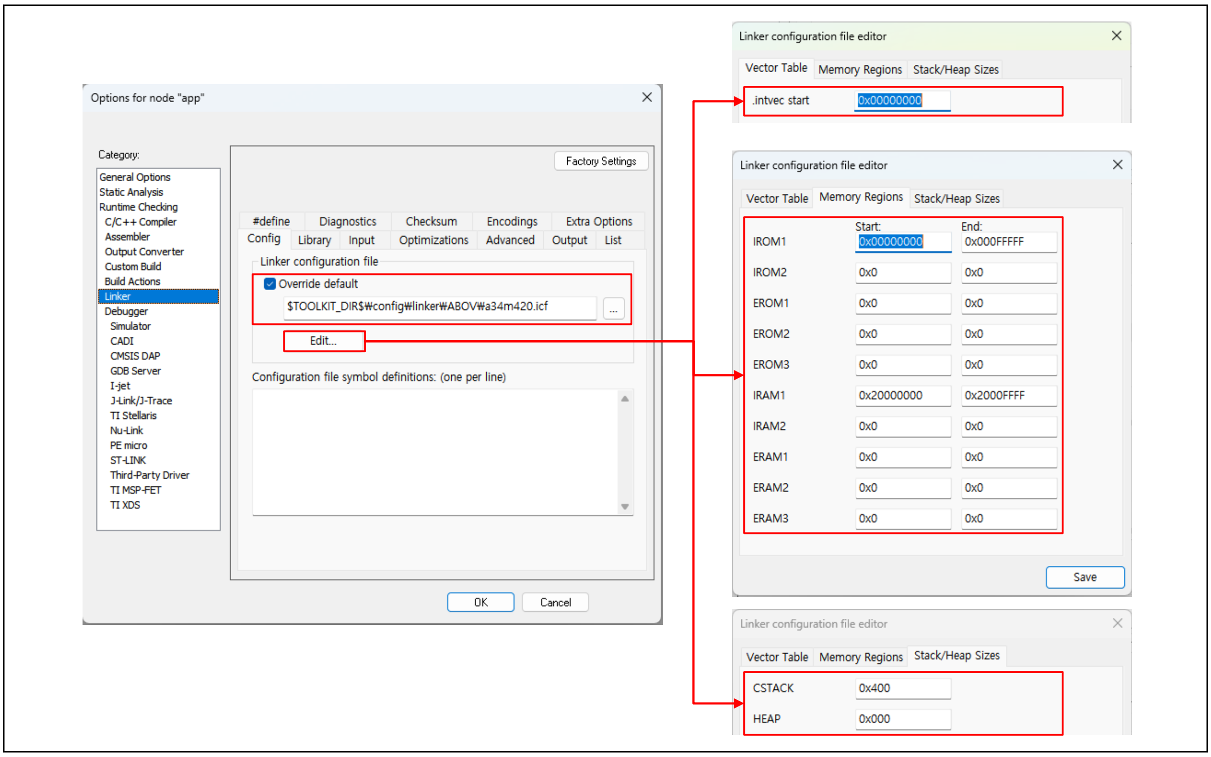
     Adjust the size of ROM and RAM according to the microcontroller memory specification[1].

     - IROM1: start 0x00000000, size 0x0007FFFF
     - IRAM1: start 0x20000000, size 0x2000FFFF

NOTES:

1. If the memory setting is incorrect, errors may occur when downloading firmware to the target device. Please refer to Table 3.
2. The size of the IROM is generally the same as the size of the Code Flash area of the target device.
3. However, in the case of a device that can separate the bank, please use it according to the size of the bank. If banks are not divided and used, all code flash areas can be used.
(ex. A34M420 Code Flash size = 0x0010_0000, Bank size = 0x0008_0000,
Not used bank function: IROM size: 0x000FFFFF, using the bank function: IROM size: 0x0007FFFF.)

Figure 18. Linker Configuration File Options for Target Device



Linker configuration file

☑ Override default

$TOOLKIT_DIR$#config#linker#ABOV#a34m420.icf

Configuration file symbol definitions: (one per line)

Linker configuration file editor — Vector Table | Memory Regions | Stack/Heap Sizes

.intvec start 0x00000000

Linker configuration file editor — Memory Regions

	Start:	End:
IROM1	0x00000000	0x000FFFFF
IROM2	0x0	0x0
EROM1	0x0	0x0
EROM2	0x0	0x0
EROM3	0x0	0x0
IRAM1	0x20000000	0x2000FFFF
IRAM2	0x0	0x0
ERAM1	0x0	0x0
ERAM2	0x0	0x0
ERAM3	0x0	0x0

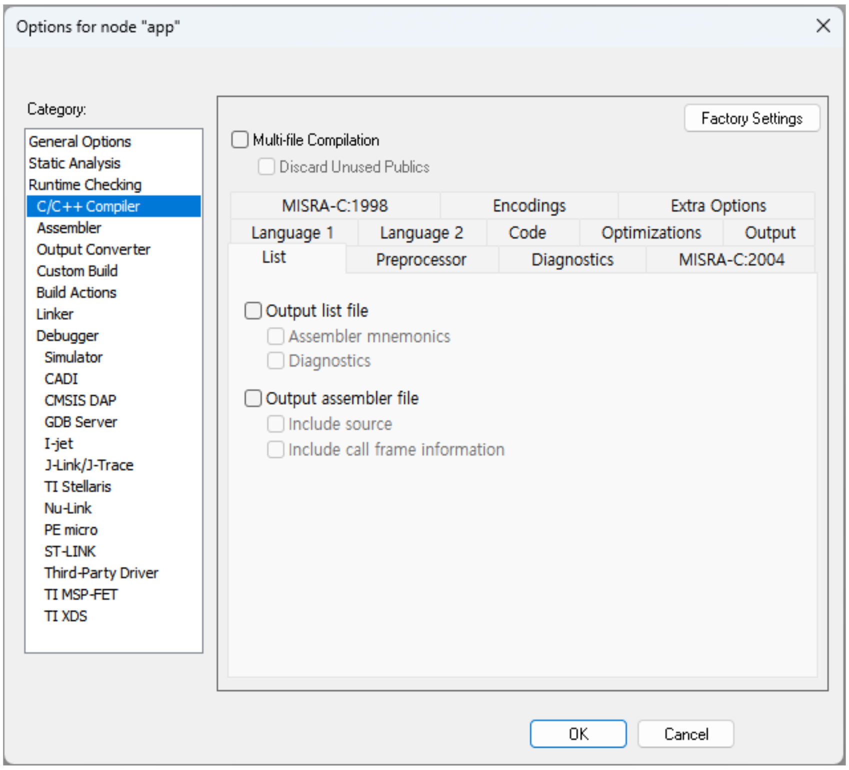
Linker configuration file editor — Stack/Heap Sizes

CSTACK	0x400
HEAP	0x000

3. Select the **Output** tab, then set the binary file name in the **Output filename** field after the project code is built to match the project name.

   A. The output binary file with *.out extension is created in the '**\Debug\Exe**' directory of the IAR project (See Figure 7).

Figure 19. Options on the Output Tab of Linker



☐ Multi-file Compilation
☐ Discard Unused Publics

☐ Output list file
☐ Assembler mnemonics
☐ Diagnostics

☐ Output assembler file
☐ Include source
☐ Include call frame information

4.4 Debug

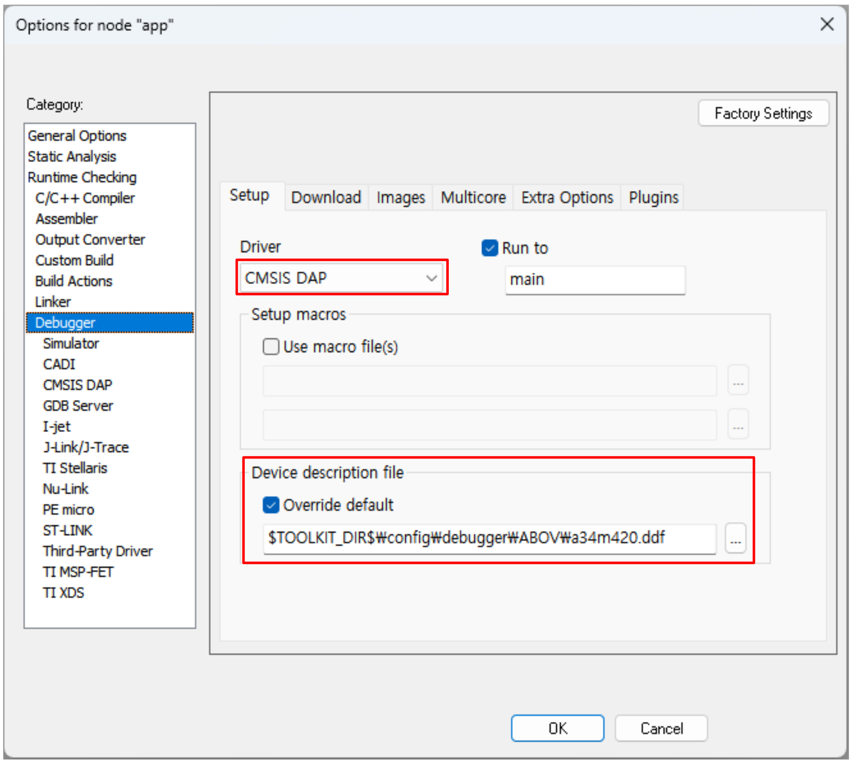
To debug the target ABOV 32-bit microcontrollers in the IAR project environment, you will need a debugging tool that supports the CMSIS-DAP interface, one of the tools listed below:

- 32-bit Microcontroller Starter Kit Board
- ABOV A-Link / A-LinkPro Debugger(CMSIS-DAP compatible)
- Segger J-Link

When the debugger tool is ready, you can follow the procedure below:

1. Select the **Debugger** in the **Category** field, then select the **Setup** tab.
2. Select the **CMSIS DAP** from the drop-down menu in the Driver field.
   A. If you have set up the **Device Description File** described in Chapter 1.2.1, you can see that the ***.ddf** file in the '**$TOOLKIT_DIR$config\debugger\ABOV\**' path is automatically set.

Figure 20. Options on the Setup Tab of Debugger



Driver: CMSIS DAP

☑ Run to: main

Setup macros

☐ Use macro file(s)

Device description file

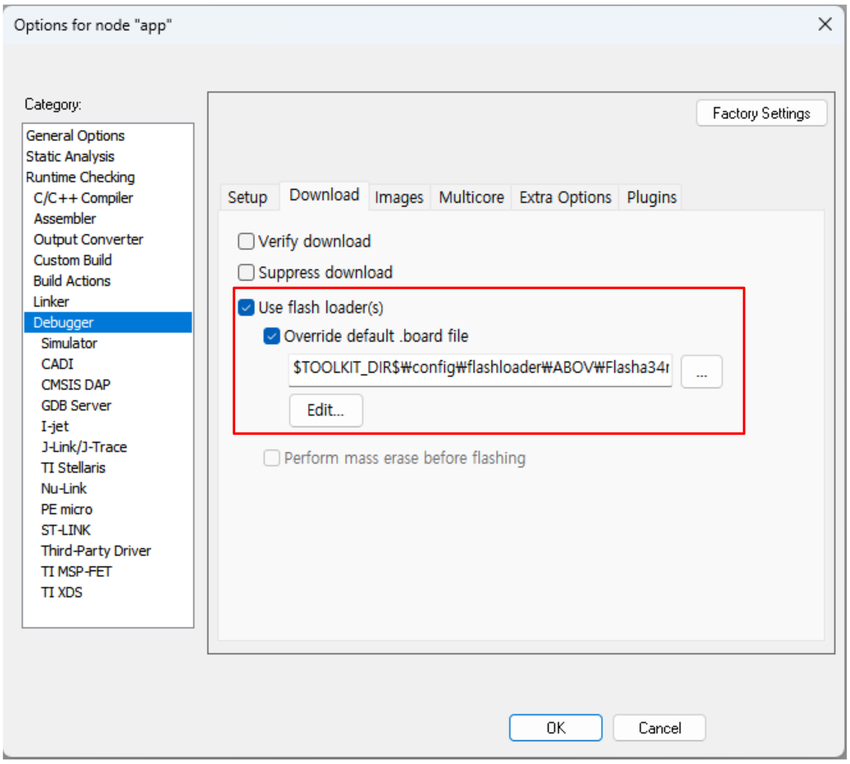
☑ Override default

$TOOLKIT_DIR$₩config₩debugger₩ABOV₩a34m420.ddf

3. Select the **Download** tab to set the flashloader file that is required when downloading the compiled binary file to the microcontroller.

   A. If you have set up the **Device Description File** described in Chapter 1.2.1, you can see that the *.out file in the '**$TOOLKIT_DIR$config\flashloader\ABOV\**' path is automatically set.

Figure 21. Options on the Download Tab of Debugger



Options for node "app"

Category: Debugger

Download

□ Verify download

□ Suppress download

☑ Use flash loader(s)

☑ Override default .board file

$TOOLKIT_DIR$₩config₩flashloader₩ABOV₩Flasha34r

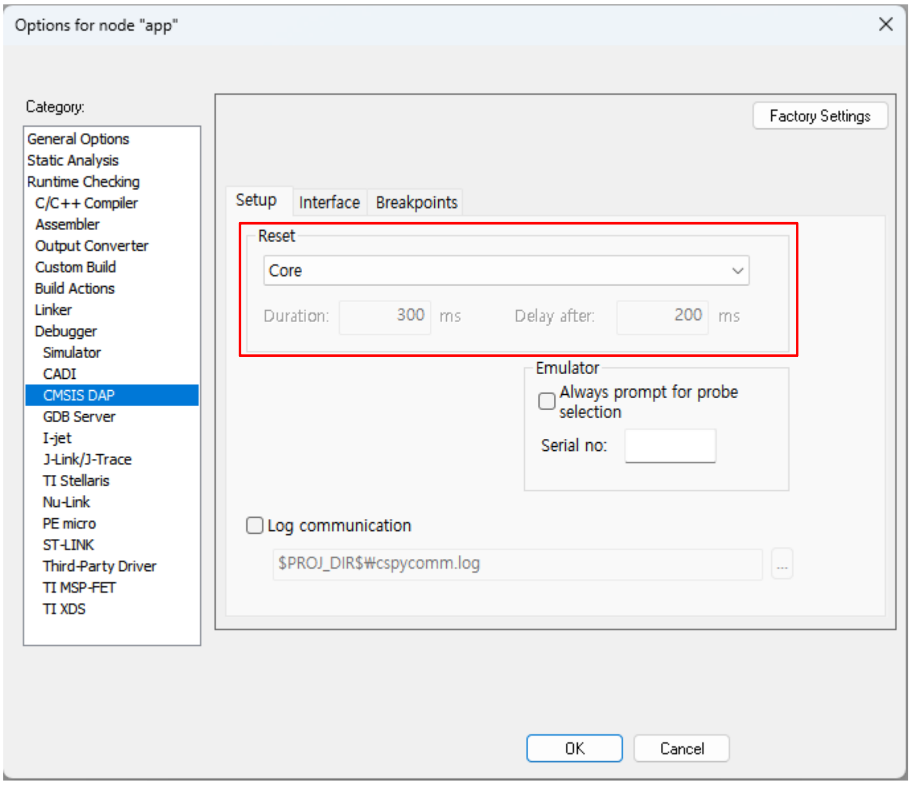
□ Perform mass erase before flashing

4.5 Debugger: CMSIS DAP

If the driver is set to **CMSIS DAP** on the **Setup** tab of the **Debugger**, as shown in Figure 20, you can set up the debugger by following the procedure below:

1. Select the **CMSIS DAP**, a submenu of the **Debugger** in the **Category** field.
2. In the **Setup** tab, set the option values appropriately from the **Reset** drop-down lists. Some devices must use **Core,** and other devices recommend **System (Default)** in Reset drop-down lists (see Table 5).

Figure 22. Options on the Setup Tab of Debugger>CMSIS DAP



Reset

Core

Duration: 300 ms Delay after: 200 ms

Emulator

□ Always prompt for probe selection

Serial no:

□ Log communication

$PROJ_DIR$₩cspycomm.log

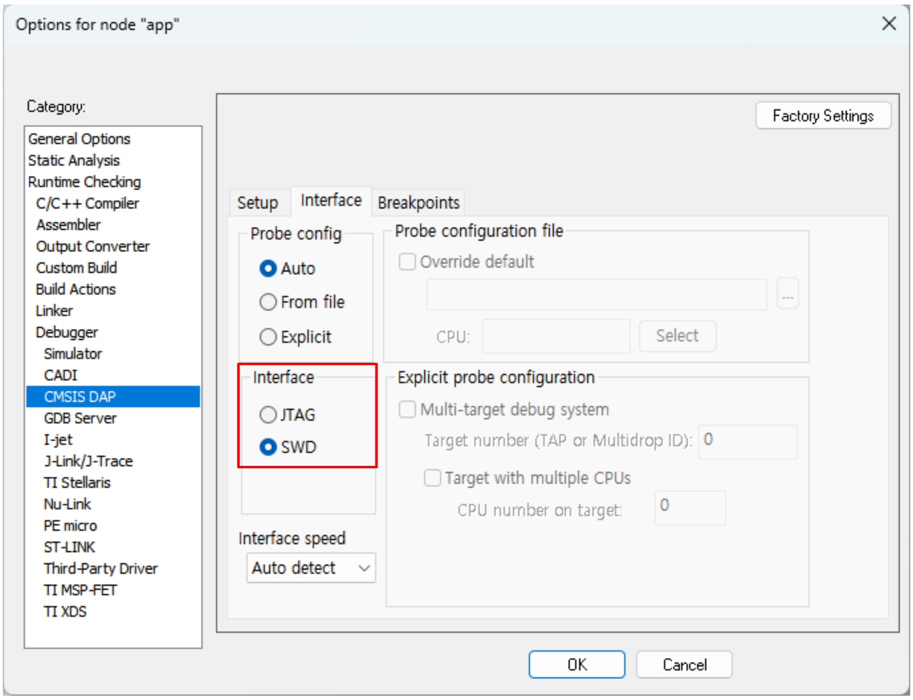
Table 5. Reset Type Selection Guide by Device

Reset Type	Device
System (Default)	A31G21x, A31G22x, A31G31x, A31G32x, A33G52x, A33G53x, A31M22x A33M11x
Core	A34M41x, A34M420, A34L716

3. Select the **SWD** in the **Interface** field on the **Interface** tab.

4. Set the options on the **Setup** and **Breakpoint** tabs to their default values.

Figure 23. Options on the Interface Tab of Debugger>CMSIS DAP



Options for node "app"

Category: CMSIS DAP

Probe config: ● Auto ○ From file ○ Explicit

Interface: ○ JTAG ● SWD

Interface speed: Auto detect

Probe configuration file: □ Override default

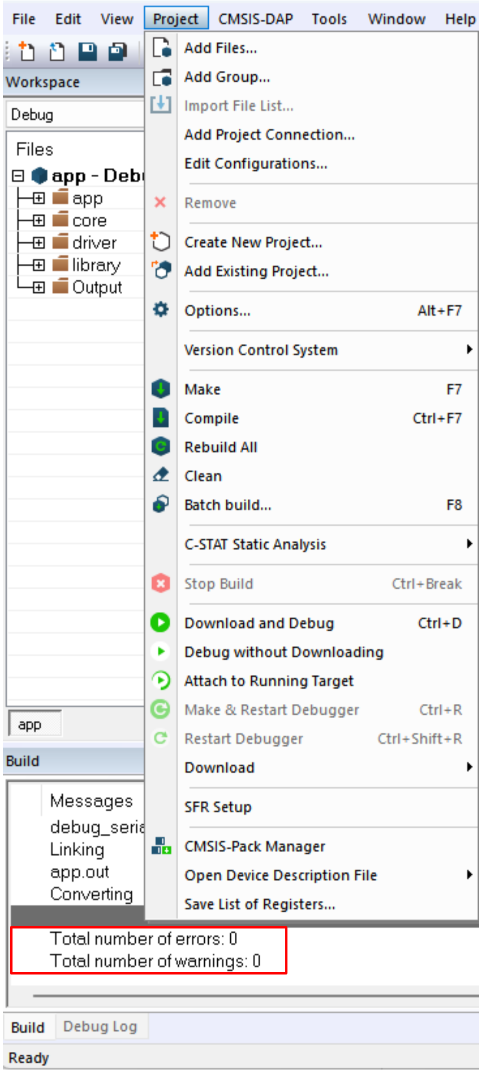
Explicit probe configuration: □ Multi-target debug system; Target number (TAP or Multidrop ID): 0; □ Target with multiple CPUs; CPU number on target: 0

5. Build and Debug IAR EWARM Project

This chapter describes how to compile the project, download firmware to the target board, and debug target devices in the IAR EWARM, as follows:

1. Select the Make or Rebuild All from the Project menu to build the source code.
2. After building the source code, check the results in the Build pane. If warnings or errors occur, locate the point where they occurred, correct the source code, and rebuild it.
3. Once the project has been successfully built, select Download and Debug from the Project menu to download the compiled firmware to the target board. When the download is complete, the project automatically enters debugging mode.

Figure 24. Build, Download, and Debug Commands of the IAR EWARM Project Window



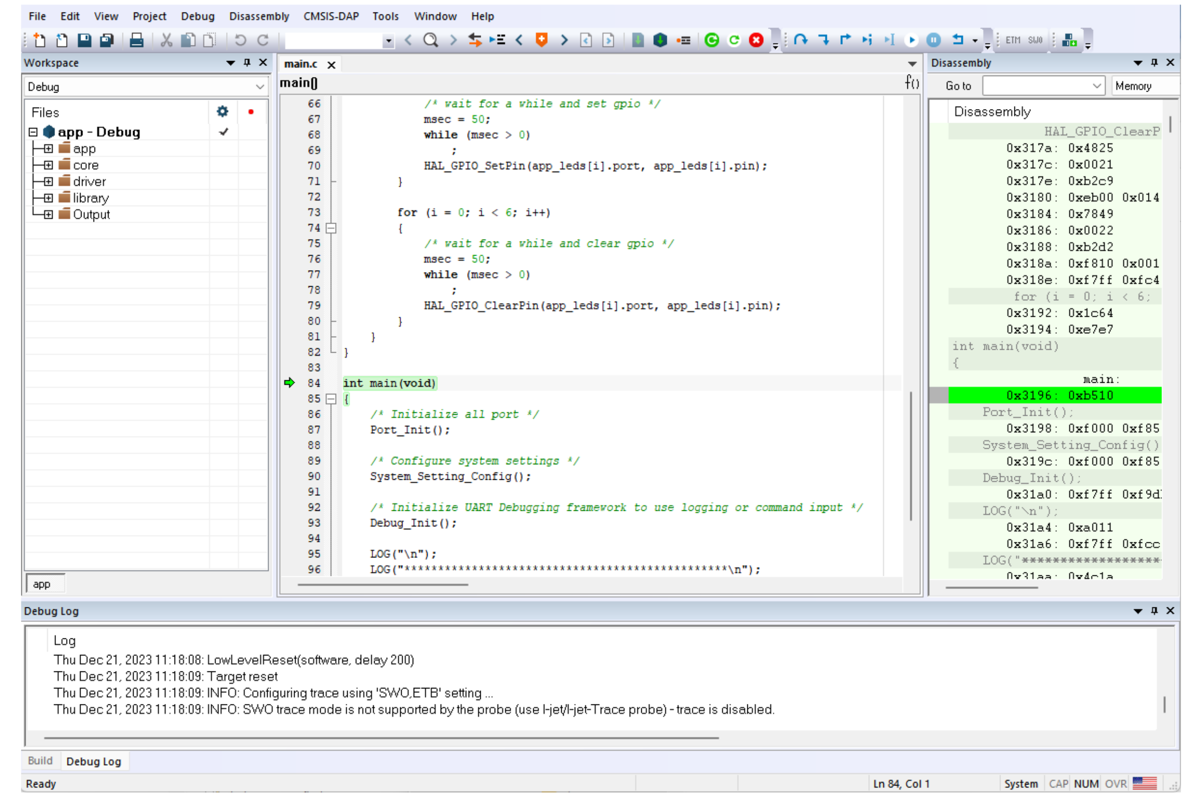
```
Messages
debug_seria
Linking
app.out
Converting

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

4. When the download is complete, you can execute and debug the program from the corresponding debugging screen.

5. On the debugging screen, as shown in Figure 25, select the **Go** command from the **Debug** menu to run the program and select the **Break** command to stop the program operation. You can also run the program step by step through the **Step** command option.

Figure 25. IAR Project Debugging Screen



```
            /* wait for a while and set gpio */
            msec = 50;
            while (msec > 0)
                ;
            HAL_GPIO_SetPin(app_leds[i].port, app_leds[i].pin);
        }

        for (i = 0; i < 6; i++)
        {
            /* wait for a while and clear gpio */
            msec = 50;
            while (msec > 0)
                ;
            HAL_GPIO_ClearPin(app_leds[i].port, app_leds[i].pin);
        }
    }
}

int main(void)
{
    /* Initialize all port */
    Port_Init();

    /* Configure system settings */
    System_Setting_Config();

    /* Initialize UART Debugging framework to use logging or command input */
    Debug_Init();

    LOG("\n");
    LOG("***********************************************\n");
```

```
Disassembly
                HAL_GPIO_ClearP
        0x317a: 0x4825
        0x317c: 0x0021
        0x317e: 0xb2c9
        0x3180: 0xeb00 0x014
        0x3184: 0x7849
        0x3186: 0x0022
        0x3188: 0xb2d2
        0x318a: 0xf810 0x001
        0x318e: 0xf7ff 0xfc4
     for (i = 0; i < 6;
        0x3192: 0x1c64
        0x3194: 0xe7e7
int main(void)
{
                     main:
        0x3196: 0xb510
    Port_Init();
        0x3198: 0xf000 0xf85
    System_Setting_Config()
        0x319c: 0xf000 0xf85
    Debug_Init();
        0x31a0: 0xf7ff 0xf9d
    LOG("\n");
        0x31a4: 0xa011
        0x31a6: 0xf7ff 0xfcc
    LOG("******************
        0x31aa: 0x4c1a
```

```
Log
Thu Dec 21, 2023 11:18:08: LowLevelReset(software, delay 200)
Thu Dec 21, 2023 11:18:09: Target reset
Thu Dec 21, 2023 11:18:09: INFO: Configuring trace using 'SWO,ETB' setting ...
Thu Dec 21, 2023 11:18:09: INFO: SWO trace mode is not supported by the probe (use I-jet/I-jet-Trace probe) - trace is disabled.
```

Revision History

Revision	Date	Notes
1.00	Dec. 21, 2023	Initial release.
1.01	May 31, 2024	Added the A34L716 device to Table 1, Table 3, and Table 5.
1.02	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
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

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.