User Manual

Document No.: AN1129

G32R5xx Keil Debug Tool User Manual

Version: V1.2

1 Introduction

Due to the chip characteristics of the G32R5xx series MCU, certain debug data streams must be configured before debug. For example, it is necessary to set the BOOT address and DCSM configuration. “keil_dbg_tool” automates these setting processes and ensures correct configuration of the debug environment.

Note: G32R501 projects use “r501_dbg.ini”; G32R502 projects use “r502_dbg.ini”. If a G32R502 example MDK directory is missing the ini, copy “utilities/keil_dbg_tool/r502_dbg.ini.template” to “<example>/project/MDK/r502_dbg.ini”. The current SDK does **not** ship “r501_dbg.ini.template”; G32R501 examples usually already include “r501_dbg.ini”. Then reference “.\r501_dbg.ini” or “.\r502_dbg.ini” under Options for Target → Debug → Initialization File. Device selection and the ini file name must match the target MCU.

Read this instruction document before using the “keil_dbg_tool”.

Operating environment of the tool:

- Windows 10/11

For daily use, the delivered “keil_dbg_tool.exe” under “utilities/keil_dbg_tool/” is sufficient; a separate Python install is **not** required. A local Python 3.11 environment is needed only when running the tool from source.

Contents

2 About Tool

The keil_dbg_tool tool provides a command line interface for users. Its main function is to automatically modify the debugging initialization scripts (INI file) in MDK-ARM by parsing AXF (ELF) files and INI setting files. These INI files set the key stack pointer (SP) and program counters (PC), in addition to BOOT configuration and DCS key configuration.

keil_dbg_tool is responsible for parsing the stack pointer (SP) and program counter (PC) in AXF files, and modifying the content of the target INI files accordingly. This tool can automate and simplify the environment configuration process during debug, especially suitable for complex G32R5xx series MCU.

3 Supported Commands

The following are the command line parameters supported by keil_dbg_tool and the corresponding descriptions:

```
keil_dbg_tool -a <axf(elf) file path> [-r] -d <output debug file path> [-v]
```

- -a <axf file path>: The path of the AXF (ELF) file.
- -d <parsed debug file path>: The path of the parsed debug INI file.
- -r: (Optional) During debugging, each reset will skip the boot code and directly set the SP and PC of the current firmware.
- -v: Display the version and the build date.
- -h: Display the help information.

4 Use Examples

Parse AXF files and configure debug INI files:

```
keil_dbg_tool -a project.axf -d project_dbg.ini
```

Display the version and the build date:

```
keil_dbg_tool -v
```

5 INI File Examples

The basic content of the INI file used for debug sequences is as follows:

```
FUNC void DCS_KEY_Setup()
{
// DCS Zone1 CSM
_WDWORD(0x50024020, DCS_ZONE1_CSM0);
_WDWORD(0x50024024, DCS_ZONE1_CSM1);
_WDWORD(0x50024028, DCS_ZONE1_CSM2);
_WDWORD(0x5002402C, DCS_ZONE1_CSM3);
// DCS Zone2 CSM
_WDWORD(0x500240A0, DCS_ZONE2_CSM0);
_WDWORD(0x500240A4, DCS_ZONE2_CSM1);
_WDWORD(0x500240A8, DCS_ZONE2_CSM2);
_WDWORD(0x500240AC, DCS_ZONE2_CSM3);
}
FUNC void Set_SP_PC_Setup(void)
{
SP= 0x20004000;
PC= 0x000009F0;
xPSR |= (1 << 24);
}
FUNC void Setup(void) {
Init_CPU();
Set_SP_PC_Setup();
}
FUNC void OnResetExec (void) {   // Executes upon software RESET
DCS_KEY_Setup();
Setup();                     // Setup for running
}
DCS_KEY_Setup();
LOAD %L INCREMENTAL
Setup();
//g, main
```

The DCS_KEY_Setup function is used to set the key for DCS (Device Configuration and Security Module). Safe zones for DCS protection and configuration devices:

- Zone1 and Zone2: G32R5xx MCU usually has two DCS zones. Each zone has its own key configuration.

- Key setting: Codes such as “_WDWORD(0x50024020, 0xFFFFFFFF);” write key values to specific DCS registers. The register addresses and key values are defined by the chip manuals. The values “0xFFFFFFFF” and “0xFFFFFFDC” in the example are specific key values.

6 How to Use

To use keil_dbg_tool in MDK-ARM, add a custom post-build step. The following is an example of how to modify INI files after a project is built.

6.1 Add Custom Post-build Steps

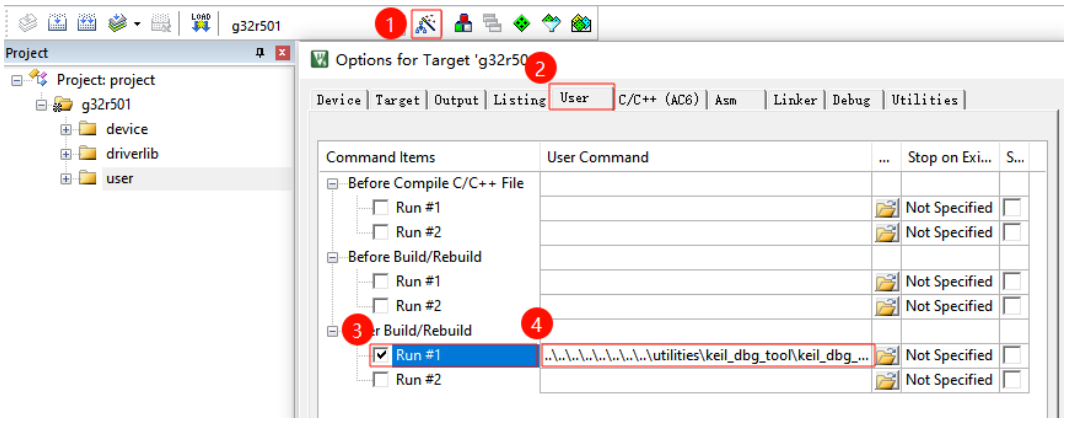
Add the custom post-build steps in MDK as follows:

1. Place “keil_dbg_tool.exe” in a known location in the project directory structure (for example, the “utilities\” directory).
2. Open the MDK project.
3. Enter the project settings window: Click “Project” in the menu bar or select “Options for Target” from the toolbar.
4. Select the “User” tab.
5. Add custom post-build steps: Locate “After Build/Rebuild”, select “Run #1” or “Run #2”, and enter the corresponding command.
6. Save configuration: Click “OK”.

The UI for adding a custom post-build step is as shown in Figure 1.

Note: The following UI uses a G32R501 example (target name commonly “g32r501”) as reference. When developing G32R502, select the matching G32R502 device in Device / Target, replace “g32r501” with “g32r502” in paths, and use “r502_dbg.ini” as the MDK debug initialization file.

Figure 1 How to Add Custom Post-build Steps



One of the following conditions must be met when using the tool:

- The path should not contain any spaces.
- If the path contains spaces, enclose the path in English double quotation marks (").

Examples of incorrect and correct paths are as follows:

- Incorrect path:

```
fromelf --bin --output $Loutput.bin
#L ..\..\..\..\..\..\..\..\utilities\keil_dbg_tool\keil_dbg_tool -r -a #L -d .\r501_dbg.ini
```

- Correct path:

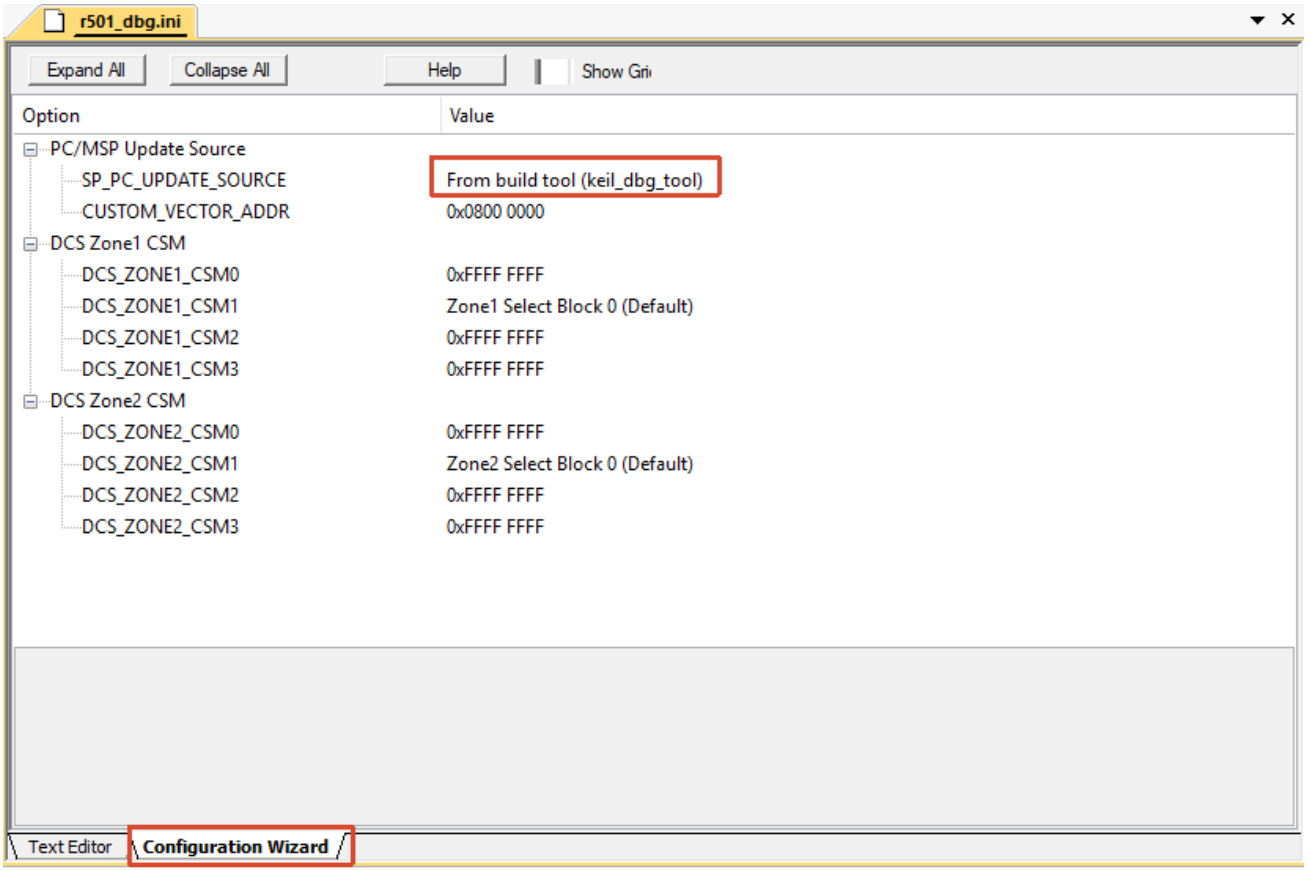
```
fromelf --bin --output "$Loutput.bin"
"#L" ..\..\..\..\..\..\..\..\utilities\keil_dbg_tool\keil_dbg_tool -r -a "#L" -d .\r501_dbg.ini
```

Note: The number of "..\" levels depends on example depth: single-core ".../project/MDK" to the SDK root is commonly 8 levels; dual-core ".../cpu0|cpu1/project/MDK" is commonly 9 levels. Use a path that resolves to "utilities/keil_dbg_tool". For G32R502 projects, change "-d" to ".\r502_dbg.ini".

7 Configuration Wizard

"r502_dbg.ini" (and template "r502_dbg.ini.template") supports the Keil **Configuration Wizard**. Open the ini in µVision, then switch to the **Configuration Wizard** tab at the bottom of the editor to configure debug parameters graphically. The UI is shown in Figure 2 (screenshot uses "r501_dbg.ini" as an example; "r502_dbg.ini" works the same way).

Figure 2 Configuration Wizard Settings UI



7.1 PC/MSP Update Source

The key item is "SP_PC_UPDATE_SOURCE":

- **0 (From build tool):** After each build, "keil_dbg_tool" writes SP/PC back into the ini from the firmware vector table. This is the recommended default for evaluation-board examples.
- **1 (Custom vector address):** The Debugger reads the vector table from "CUSTOM_VECTOR_ADDR" (offset 0 = MSP, offset 4 = PC). Typical values: CBUS Flash "0x08000000", ITCM RAM "0x00000000" (choose by device and linker script).

When you need a fixed vector base for PC/MSP, or do not want the tool to overwrite SP/PC in the ini on every build, set "SP_PC_UPDATE_SOURCE" to 1 and set "CUSTOM_VECTOR_ADDR" correctly.

DCS Zone1/Zone2 CSM and related key items can also be edited in the Wizard; they must

match the actual device keys.

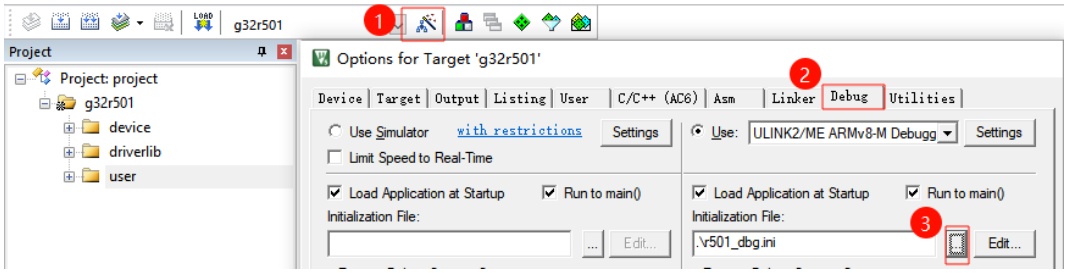
7.2 Debug Script Selection

After the corresponding debug script is modified, configure the project so that MDK runs the script during debug:

1. Open the MDK project.
2. Enter Options for Target.
3. Select the “Debug” tab.
4. Add the debugging script: use “…” and select the matching “.ini” file.
5. Save configuration: Click “OK”.

Debug script selection is shown in Figure 3.

Figure 3 Selecting Debug Scripts



8 Revision

Table 1 Document revision

Date	Version	Description
2025.1	1.0	● Initial release
2025.4	1.1	● In “Add Custom Post-build Steps”, note that paths with spaces require double quotes
2026.8	1.2	● Applicable products listed as G32R501 and G32R502 ● Added “r502_dbg.ini” and G32R502 project configuration notes ● Added “r502_dbg.ini” Configuration Wizard and “SP_PC_UPDATE_SOURCE” description ● Corrected r501 template path wording

Statement

This document is formulated and published by Geehy Semiconductor Co., Ltd. (hereinafter referred to as “Geehy”). The contents in this document are protected by laws and regulations of trademark, copyright and software copyright. Geehy reserves the right to make corrections and modifications to this document at any time. Read this document carefully before using Geehy products. Once you use the Geehy product, it means that you (hereinafter referred to as the “users”) have known and accepted all the contents of this document. Users shall use the Geehy product in accordance with relevant laws and regulations and the requirements of this document.

1. Ownership

This document can only be used in connection with the corresponding chip products or software products provided by Geehy. Without the prior permission of Geehy, no unit or individual may copy, transcribe, modify, edit or disseminate all or part of the contents of this document for any reason or in any form.

The “极海” or “Geehy” words or graphics with “®” or “TM” in this document are trademarks of Geehy. Other product or service names displayed on Geehy products are the property of their respective owners.

2. No Intellectual Property License

Geehy owns all rights, ownership and intellectual property rights involved in this document.

Geehy shall not be deemed to grant the license or right of any intellectual property to users explicitly or implicitly due to the sale or distribution of Geehy products or this document.

If any third party’s products, services or intellectual property are involved in this document, it shall not be deemed that Geehy authorizes users to use the aforesaid third party’s products, services or intellectual property. Any information regarding the application of the product, Geehy

hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party, unless otherwise agreed in sales order or sales contract.

3. Version Update

Users can obtain the latest document of the corresponding models when ordering Geehy products.

If the contents in this document are inconsistent with Geehy products, the agreement in the sales order or the sales contract shall prevail.

4. Information Reliability

The relevant data in this document are obtained from batch test by Geehy Laboratory or cooperative third-party testing organization. However, clerical errors in correction or errors caused by differences in testing environment may occur inevitably. Therefore, users should understand that Geehy does not bear any responsibility for such errors that may occur in this document. The relevant data in this document are only used to guide users as performance parameter reference and do not constitute Geehy's guarantee for any product performance.

Users shall select appropriate Geehy products according to their own needs, and effectively verify and test the applicability of Geehy products to confirm that Geehy products meet their own needs, corresponding standards, safety or other reliability requirements. If losses are caused to users due to user's failure to fully verify and test Geehy products, Geehy will not bear any responsibility.

5. Legality

USERS SHALL ABIDE BY ALL APPLICABLE LOCAL LAWS AND REGULATIONS WHEN USING THIS DOCUMENT AND THE MATCHING GEEHY PRODUCTS. USERS SHALL UNDERSTAND THAT THE PRODUCTS MAY BE RESTRICTED BY THE EXPORT, RE-EXPORT OR OTHER LAWS OF THE COUNTRIES OF THE PRODUCTS SUPPLIERS, GEEHY, GEEHY

DISTRIBUTORS AND USERS. USERS (ON BEHALF OR ITSELF, SUBSIDIARIES AND AFFILIATED ENTERPRISES) SHALL AGREE AND PROMISE TO ABIDE BY ALL APPLICABLE LAWS AND REGULATIONS ON THE EXPORT AND RE-EXPORT OF GEEHY PRODUCTS AND/OR TECHNOLOGIES AND DIRECT PRODUCTS.

6. Disclaimer of Warranty

THIS DOCUMENT IS PROVIDED BY GEEHY "AS IS" AND THERE IS NO WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, TO THE EXTENT PERMITTED BY APPLICABLE LAW.

GEEHY'S PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED FOR USE AS CRITICAL COMPONENTS IN MILITARY, LIFE-SUPPORT, POLLUTION CONTROL, OR HAZARDOUS SUBSTANCES MANAGEMENT SYSTEMS, NOR WHERE FAILURE COULD RESULT IN INJURY, DEATH, PROPERTY OR ENVIRONMENTAL DAMAGE.

IF THE PRODUCT IS NOT LABELED AS "AUTOMOTIVE GRADE," IT SHOULD NOT BE CONSIDERED SUITABLE FOR AUTOMOTIVE APPLICATIONS. GEEHY ASSUMES NO LIABILITY FOR THE USE BEYOND ITS SPECIFICATIONS OR GUIDELINES.

THE USER SHOULD ENSURE THAT THE APPLICATION OF THE PRODUCTS COMPLIES WITH ALL RELEVANT STANDARDS, INCLUDING BUT NOT LIMITED TO SAFETY, INFORMATION SECURITY, AND ENVIRONMENTAL REQUIREMENTS. THE USER ASSUMES FULL RESPONSIBILITY FOR THE SELECTION AND USE OF GEEHY PRODUCTS. GEEHY WILL BEAR NO RESPONSIBILITY FOR ANY DISPUTES ARISING FROM THE SUBSEQUENT DESIGN OR USE BY USERS.

7. Limitation of Liability

IN NO EVENT, UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL GEEHY OR ANY OTHER PARTY WHO PROVIDES THE DOCUMENT AND PRODUCTS

"AS IS", BE LIABLE FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, DIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE DOCUMENT AND PRODUCTS (INCLUDING BUT NOT LIMITED TO LOSSES OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY USERS OR THIRD PARTIES). THIS COVERS POTENTIAL DAMAGES TO PERSONAL SAFETY, PROPERTY, OR THE ENVIRONMENT, FOR WHICH GEEHY WILL NOT BE RESPONSIBLE.

8. Scope of Application

The information in this document replaces the information provided in all previous versions of the document.