# Application Note

**Document No.: AN1185**

**G32R430 Load Restrictions and Handling for IAR Interrupt Calls to ITCM Functions**

**Version: V1.0**

# 1 Introduction

The G32R430 supports executing selected functions from ITCM (Instruction Tightly Coupled Memory), which is on-chip RAM, to reduce instruction-fetch latency. In the G32R430 DDL SDK, section attributes can place functions in ITCM. For example, SECTION_ITCM_RAMFUNC places functions in the “itcm.ramfunc” linker section.

The current IAR toolchain has a limitation: the linker treats functions referenced by the interrupt vector table as “needed for initialization,” meaning they are required during initialization. If an interrupt service routine (ISR) referenced by the vector table calls a function in RAM or TCM, including ITCM, and the C runtime library loads that function from Flash using “initialize by copy,” the linker may not establish the required load dependency correctly. As a result, the CPU may attempt to execute code before it has been loaded correctly. The default SDK design and configuration described in this note provide a practical solution to this limitation.

This note explains:

- The default DDL SDK design for IAR Flash-boot applications and the reasons for this design;
- The approximate startup-cycle overhead measured during board-level testing, to help evaluate user-side loading;
- How to switch to IAR runtime-library loading when no interrupt path calls ITCM functions.

# Contents

# 2 Principles

When booting from Flash, ITCM code normally has a load image (LMA) stored in Flash. After power-up, this image must be copied to its ITCM execution address (VMA, starting at approximately 0x00000000) before the CPU can fetch and execute the instructions correctly.

The IAR linker script provides two loading mechanisms:

- **initialize by copy**: The IAR C runtime library automatically performs the copy before entering main.
- **initialize manually**: The linker generates the load image, and user startup code copies it early in the startup process, for example, in ItcmData_Init.

Typical ITCM-related sections in the DDL SDK include itcm.ramfunc, itcm.instruction, and atan2_instruction. If an application places functions in these sections, the Flash→ITCM copy must finish before the functions are first executed.

The two Flash-boot paths can be summarized as follows:

- **User-side path (SDK default)**: Reset → SystemInit → ItcmData_Init (copies ITCM-related sections) → C library (copies normal RW sections, etc.) → main.
- **Runtime-library path (optional)**: Reset → SystemInit → C library (uses “initialize by copy” for ITCM-related sections) → main (ItcmData_Init is not called).

**Note:**

（1） Use the user-side path when an ISR may call an ITCM function.

（2） Use the runtime-library path only when no path reachable from the interrupt vector table calls an ITCM function. See Chapter 5.

# 3 Current SDK IAR Design and Rationale

## 3.1 Default Policy (Recommended)

Official linker script path:

```
G32R430_DDL_SDK/Libraries/Device/Geehy/G32R4xx/Source/iar/g32r430xb_flash.icf
```

Default configuration:

```
define symbol __itcm_init_mode__ = 0x0;
```

The default is user-side loading ("__itcm_init_mode__ = 0"). In this mode, the ICF file uses "initialize manually" for ITCM-related sections. In the Reset_Handler of the startup_g32r430xx.c device startup file, ItcmData_Init() is called before control enters the C library. This function copies the load image from Flash to ITCM one byte at a time.

This default policy is used for the following reasons:

- The IAR linker treats functions referenced by the interrupt vector table as "needed for initialization."
- If an ISR calls a function in RAM or TCM, such as ITCM, relying only on "initialize by copy" may not correctly establish the Flash→ITCM load dependency.
- Using "initialize manually" for the relevant sections and completing the copy in ItcmData_Init before entering the C library ensures that the ISR→ITCM call path is valid at run time.

The default SDK configuration is summarized in Table 1.

Table 1 SDK Default IAR ITCM Load Configuration

| Item | Default Value/Behavior |
|---|---|
| ICF symbol __itcm_init_mode__ | 0 (user-side) |
| ITCM-related sections | initialize manually |
| Device startup | Calls ItcmData_Init () when IAR_ITCM_AUTO_INIT is not defined |
| ItcmData_Init implementation | Byte-wise copy loop |
| Applicable scenarios | An ISR may call an ITCM function, or the project must remain consistent with the default SDK behavior |

## 3.2 Difference from the Full Image in ITCM Script

The g32r430xb_itcm_ram.icf script places read-only code directly in ITCM, and the debugger or download process writes the image to ITCM. Because no Flash→ITCM section copy is required, the __itcm_init_mode__ switch is not needed.

Chapters 4 and 5 apply to the g32r430xb_flash.icf scenario: Flash boot with ITCM sections that must be copied.

## 3.3 Notes on the Copy Implementation

The official SDK implementation of ItcmData_Init uses byte-wise copying. Compared with runtime-library loading, this method adds startup cycles. The overhead increases with the number of ITCM bytes copied; see Chapter 4. For projects with large ITCM load images, see Section 6.2 for an optimization approach that can be evaluated separately.

# 4 Design Verification and Test Data

## 4.1 Test Purpose

The purpose of this test is to measure the relative delay in reaching main when IAR user-side ItcmData_Init is used instead of other loading methods.

The timing method is consistent with common GPIO startup-timing examples:

1. After SystemInit finishes, clear and enable the DWT cycle counter. SystemInit is not included in the measurement.
2. In IAR user-side load mode, execute ItcmData_Init.
3. Enter C library initialization.
4. Read CycleCount (DWT CYCCNT) at the entry to main.

Relative delay is defined as:

```
relative_delay = CycleCount(IAR user-side) – CycleCount(reference scenario)
```

A positive value means that user-side loading takes longer to reach main. At 8 MHz, the time in microseconds is estimated as follows: μs ≈ cycles / 8.

## 4.2 Test Environment

The board, clock, number of repetitions, functional scenario, load size, and optimization levels used in this test are as follows.

- Board: G32R430 Tiny evaluation board.
- Core frequency at the measurement point: approximately 8 MHz HSI. When CycleCount is read at the entry to main, the application has not yet switched to the PLL.
- Repetitions: At least five runs per configuration, using the median result.
- Functional scenario: A SysTick ISR calls a function placed in ITCM. The function address is within the ITCM range, for example, 0x00000009. This confirms that the ISR→ITCM path works.
- ITCM load size: Only ITCM_ToggleLed was placed in itcm.ramfunc for this test. The Flash→ITCM byte count was obtained from the linker map. See Section 4.3 for instructions and Table 2 for a summary. Projects with larger ITCM sections usually require more startup copy time. The CycleCount and relative-delay values in Tables 2 and 3 apply only to this test's load size and must not be extrapolated directly to other sizes.
- Optimization levels: O0 / O2 / Ofast. IAR does not provide a literal Ofast setting, so None / High / High + Speed were used. "IAR ≈ Ofast" refers to High + Speed in this report.

The IDE and compiler versions used in this test are as follows.

- IAR
  - Version: IAR Embedded Workbench for Arm 9.60.2
  - Toolchain: IAR C/C++ Compiler for Arm (bundled with EWARM 9.60.2; the linker map identifies V9.60.2)
- MDK
  - Version: Keil MDK 5.43 (µVision)
  - Toolchain: Arm Compiler for Embedded 6.24 (ArmClang/armlink)
- Eclipse
  - Version: Eclipse IDE 4.35.0 RC1 + Embedded CDT
  - Toolchain: LLVM Embedded Toolchain for Arm 19.1.5 (clang 19.1.5)

## 4.3 How to Confirm the Number of Bytes to Copy Using the Map

Use the linker map to determine the Flash→ITCM copy size for a project. Follow these steps:

1. In IAR, rebuild the target configuration and open the generated map file. The file is normally in the List folder under the project configuration directory, for example, IAR_ITCM_ManualInit.map.
2. Search the map for itcm.ramfunc and itcm.ramfunc_init. If the project uses itcm.instruction or atan2_instruction, also search for their corresponding _init names.
3. For user-side loading (the SDK default using ItcmData_Init), use the Size of each Flash-side load-image block. For itcm.ramfunc, find the itcm.ramfunc_init block or the Size on the “Initializer bytes” line marked “for itcm.ramfunc.” Alternatively, calculate “itcm.ramfunc_init$$Limit − itcm.ramfunc_init$$Base” in the ENTRY LIST. If itcm.instruction_init or atan2_instruction_init also exists, add the Size of each init block. The sum is the total number of bytes copied by ItcmData_Init.
4. For runtime-library loading using “initialize by copy,” find the Size of itcm.ramfunc and any other ITCM sections in the PLACEMENT SUMMARY. This is the size of the execution image placed in ITCM and can be compared with the user-side path.

Always use the latest map file for the project. Optimization settings and added or removed functions change the section Size. Re-evaluate startup time whenever the load size changes; see Table 2.

The following excerpt is from the map for the IAR user-side O0 test configuration. The Size of itcm.ramfunc_init is 0x34, or 52 bytes. This is the IAR user-side O0 load size shown in Table 2.

```
Section              Kind         Address  Alignment    Size  Object
------               ----         -------  ---------    ----  ------
"P2":                                                     0x34
  itcm.ramfunc                          0x0              0x34  <Block>
    itcm.ramfunc-1                      0x0          4   0x34  <Init block>
      Veneer           inited           0x0          4    0x8  - Linker created -
      itcm.ramfunc     inited           0x8          4   0x2c  main.o [1]
                                    - 0x34                0x34
  itcm.ramfunc_init              0x800'1494               0x34  <Block>
    Initializer bytes  const     0x800'1494          4   0x34  <for itcm.ramfunc-1>
itcm.ramfunc$$Base               0x0          --   Gb  - Linker created -
itcm.ramfunc$$Limit             0x34          --   Gb  - Linker created -
itcm.ramfunc_init$$Base
                         0x800'1494          --   Gb  - Linker created -
itcm.ramfunc_init$$Limit
                         0x800'14c8          --   Gb  - Linker created -
```

**Note:**

（1） In the excerpt, the itcm.ramfunc function body in the execution region is 0x2c, while the init block, itcm.ramfunc_init, is 0x34 because it includes a linker-generated veneer and other data. ItcmData_Init copies the complete init block, so 0x34 is used.

（2） This test did not use itcm.instruction or atan2_instruction, so the map contains no corresponding _init lines. If a project uses these sections, include their sizes in the total.

## 4.4 Key Test Code Excerpts

The following excerpts show how the verification project creates the ISR→ITCM path and timing window used for the board tests. Projects can use the same structure to place functions in ITCM and then select user-side or runtime-library loading as described in Chapters 3 and 5. In the startup file, ItcmData_Init is called when IAR_ITCM_AUTO_INIT is not defined and skipped when the macro is defined; see Table 4.

Placing a function in ITCM (itcm.ramfunc):

```
SECTION_ITCM_RAMFUNC void ITCM_ToggleLed(void)
{
    g_systickCount++;
    if ((g_systickCount % 500U) == 0U)
    {
        DDL_GPIO_TogglePin(GPIOA, DDL_GPIO_PIN_1);
    }
}
```

Calling the ITCM function from an ISR referenced by the vector table:

```
void SysTick_Handler(void)
{
    ITCM_ToggleLed();
}
```

Starting the timing window after reset and calling ItcmData_Init in user-side load mode before entering the C library:

```
SystemInit();
/* clear and enable DWT CYCCNT (SystemInit is not counted in CycleCount) */
CoreDebug->DEMCR |= (1UL << 24);
DWT->CYCCNT = 0U;
DWT->CTRL |= (1UL << 0);

# if defined(__ICCARM__) && !defined(IAR_ITCM_AUTO_INIT)
    ItcmData_Init();

# endif
__PROGRAM_START();
```

Reading CycleCount at the entry to main:

```
int main(void)
{
    CycleCount = (*(volatile uint32_t *)0xE0001004U);
    /* … subsequent clock/USART/peripheral initialization … */
}
```

**Note:**

When functional verification passes, the ITCM function address should be within the ITCM range. In this test, the address was approximately 0x00000009.

## 4.5 Results Summary

As shown in Table 2, this section combines the ITCM load sizes obtained from the map with the CycleCount measured before entering main. Each result is the median of five runs. Each cell shows “load bytes / CycleCount.” O0 / O2 / Ofast results for the same environment are listed in one row. Each CycleCount applies only to the byte count shown in the same cell.

Table 2 ITCM Load Bytes and CycleCount before Entering Main in This Test (Median, n = 5)

| No. | Environment | O0 (bytes/cycles) | O2 (bytes/cycles) | Ofast (bytes/cycles) |
|---|---|---|---|---|
| 1 | IAR user-side | 52/2556 | 56/1451 | 56/2727 |
| 2 | IAR runtime library | 44/491 | 48/509 | 48/553 |
| 3 | MDK | 58/1214 | 58/1500 | 58/1505 |
| 4 | Eclipse | 60/2891 | 100/956 | 100/982 |

**Notes:**

- For the IAR user-side row, the byte count is taken from itcm.ramfunc_init in the map, that is, the number of bytes actually copied by ItcmData_Init.
- In the Ofast column, IAR corresponds to optimization level High with a speed bias (High+Speed). IAR has no literal Ofast setting; see Section 4.2 for the mapping.
- The IAR user-side value is based on the init block. That block may be slightly larger than the function body because of alignment and other content generated by the linker.
- The optimization level changes the size of the generated code, so the byte count can differ across optimization levels in the same environment.
- The user should check the latest map of this project. If the number of ITCM bytes to be copied changes, the startup time must be re-evaluated.

As shown in Table 3, the following table lists the delay of the IAR user-side path relative to the reference scenarios.

Table 3 Delay of IAR User-Side Relative to Reference Scenarios (User-Side − Reference)

| Optimization Level | Relative to IAR Runtime Library | Relative to MDK | Relative to Eclipse |
|---|---|---|---|
| O0 | +2065 (≈258 μs) | +1342 (≈168 μs) | −335 (≈−42 μs) |
| O2 | +942 (≈118 μs) | −49 (≈−6 μs) | +495 (≈62 μs) |
| Ofast | +2174 (≈272 μs) | +1222 (≈153 μs) | +1745 (≈218 μs) |

Key conclusions:

- For the ITCM load size used in this test, approximately 52–56 bytes as shown in row 1 of Table 2, user-side loading was about 900–2200 cycles slower than IAR runtime-library loading. This is approximately 0.12–0.27 ms at 8 MHz.

- If a project has almost no ITCM content to copy, such as a simple GPIO toggle example, the CycleCount difference between IAR and MDK may be only a few dozen cycles. Such a result is not directly comparable with an ISR→ITCM test that includes an actual ITCM load.
- As the number of ITCM bytes increases, startup time generally increases for both user-side and runtime-library loading. Re-evaluate startup time using the actual section sizes in the project's map; see Table 2.
- With the small load size used in this test, the CycleCount for IAR ≈ Ofast can be higher than for O2. See Section 6.2 for an explanation.

## 4.6 Boundaries for Interpreting the Data

- CycleCount comparisons across toolchains include differences in each vendor's C library and startup path. They must not be treated as a direct comparison of copy-algorithm efficiency.
- Small timing differences when the ITCM copy size is nearly zero do not remove the need for user-side loading in ISR→ITCM scenarios.
- Tables 2 and 3 apply to the same board-test matrix. Actual values vary with ITCM section size, optimization settings, and library version. Re-measure under consistent conditions.

# 5 Runtime-Library Loading: When to Use and How to Configure

## 5.1 Applicability Conditions

IAR runtime-library loading ("__itcm_init_mode__ = 1") may be used only when all the following conditions are met:

- No path reachable from the interrupt vector table calls a function in ITCM, including through an indirect call.
- ITCM functions are called only from main or thread context, or the project does not use ITCM code sections.

Keep the SDK's default user-side loading in the following case:

- An ISR, or any handler referenced by the vector table, calls a function in ITCM.

**Note:**

（1） Enabling runtime-library loading incorrectly in an ISR→ITCM scenario may cause abnormal behavior or intermittent faults.

（2） After changing the loading method, perform a full rebuild followed by functional and startup regression testing.

## 5.2 Steps to Switch to Runtime-library Loading

1. Open the g32r430xb_flash.icf file used by the project. This may be the script under the SDK Device path or a copy in the project directory.
2. Change __itcm_init_mode__ from 0 (default user-side loading) to 1 (runtime-library loading). For example, change:

```
define symbol __itcm_init_mode__ = 0x0;
```

to:

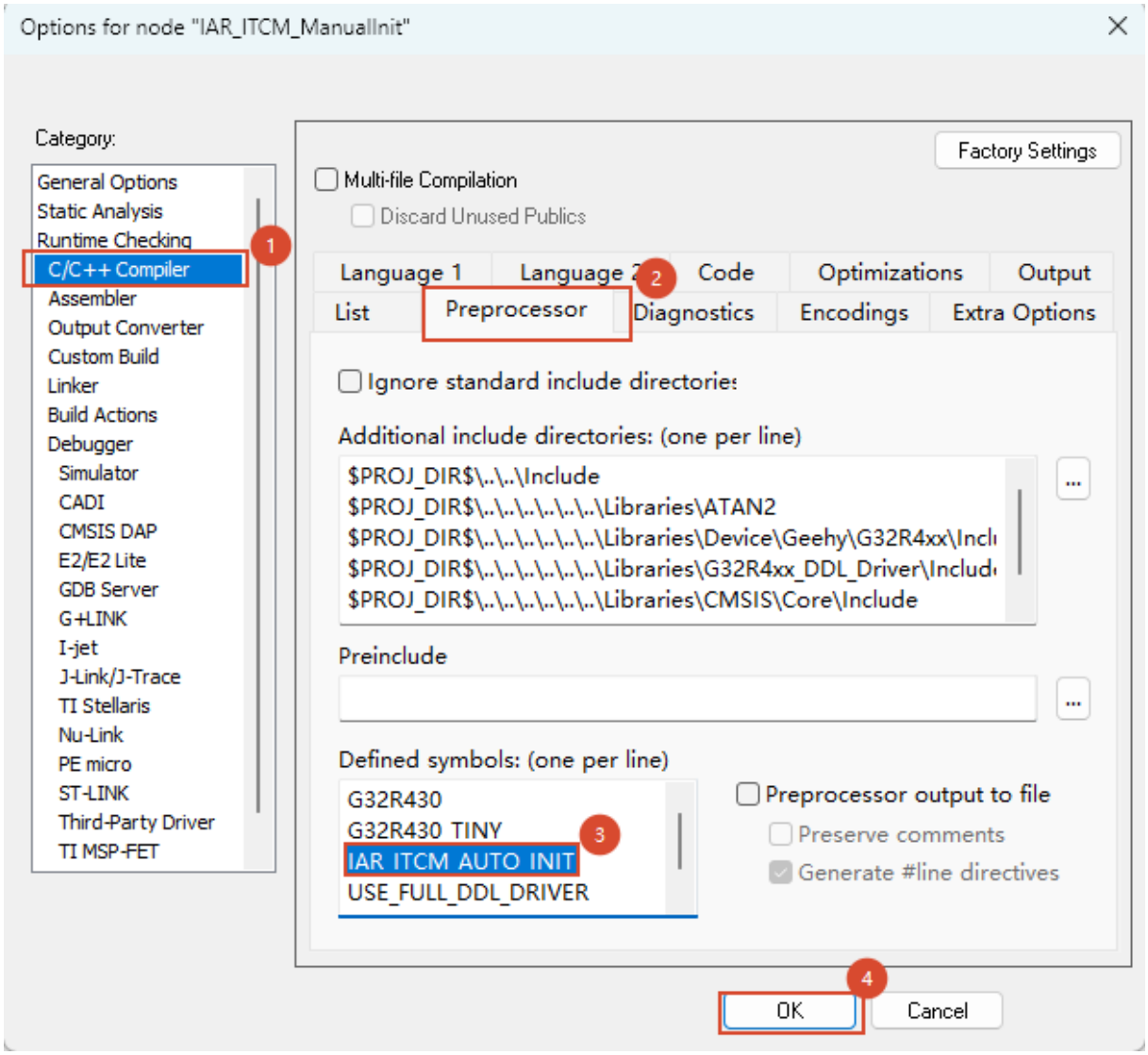
```
define symbol __itcm_init_mode__ = 0x1;
```

3. Add the predefined macro IAR_ITCM_AUTO_INIT to the IAR project options. This makes the device startup code skip ItcmData_Init(), preventing a duplicate copy or conflict with the C library's automatic copy. Figure 1 shows the procedure, corresponding to ①–④:

- Select C/C++ Compiler under Category
- Open the Preprocessor tab
- Add IAR_ITCM_AUTO_INIT under Defined symbols, with one macro per line
- Click OK to confirm

As shown in Figure 1, add the predefined macro IAR_ITCM_AUTO_INIT in IAR project options.

Figure 1 Adding the Predefined Macro IARITCMAUTOINIT in IAR Project Options




4. Rebuild, download, and run the project. Confirm the following:

- If ITCM functions are present, their addresses are within the ITCM range;
- Application functions and interrupt behavior operate as expected.

As shown in Table 4, the following checklist compares user-side and runtime-library configurations.

Table 4 User-Side/Runtime-Library Load Configuration Checklist

| Check Item | User-side (Default) | Runtime Library |
|---|---|---|
| __itcm_init_mode__ | 0 | 1 |

| Check Item | User-side (Default) | Runtime Library |
|---|---|---|
| Project macro IAR_ITCM_AUTO_INIT | Not defined | Must be defined |
| ItcmData_Init | Called | Skipped |
| Party responsible for loading ITCM sections | User-side copy during startup | IAR C library "initialize by copy" |
| ISR→ITCM | Recommended | Not recommended |

**Note:**

__itcm_init_mode__ and IAR_ITCM_AUTO_INIT must always be changed together. Do not change only one of them.

## 5.3 Reverting to User-side Loading

1. Restore __itcm_init_mode__ to 0.
2. Remove IAR_ITCM_AUTO_INIT from the project's predefined macros.
3. Rebuild the project and perform regression testing.

# 6 Additional Notes

## 6.1 MDK and Eclipse

Keil MDK uses scatter loading, and Eclipse/Clang uses a copy table. These toolchains generally do not have the same “initialize by copy” limitation as IAR when an ISR referenced by the vector table calls an ITCM function. For these project types, the scatter-loading or linker script supplied with the SDK is sufficient to load RW and ITCM sections. Startup-cycle differences between toolchains mainly result from differences in the C library and copy implementation. See Section 4.6 when interpreting the results.

## 6.2 Optimization and Copy Implementation

The default ItcmData_Init implementation in the official SDK uses byte-wise copying. Do not rely only on changing the optimization level to Ofast to reduce the copy time. With the small load size used in this test, Ofast did not always reduce the execution time of the small copy loop.

If a project has a large amount of ITCM content and startup time becomes a bottleneck, ItcmData_Init may be evaluated with word-wise or alignment-width copying and a higher optimization level, such as O2/High. The correct loading order must be preserved, and complete functional and startup regression testing is required after any change. The default SDK continues to use byte-wise copying.

## 6.3 Other Cautions

- For vector-table remapping, secondary boot, IAP multi-image, and similar scenarios, evaluate ITCM section ownership and loading time separately. These scenarios are outside the scope of this note.
- The timing and functional tests in this note used a Flash-boot reference project with an ISR→ITCM call path. Configure each project according to the checklist in this chapter, and measure startup time using the project’s actual ITCM section size.

# 7 References

**Issues with Running Interrupt Service Routines in TCM/RAM with IAR**:

https://mp.weixin.qq.com/s/jRulLkSzGWEQ1mB-Z65Mpw

# 8 Revision

The document revision history is given in Table 5.

Table 5 Document revision

| Date | Version | Description |
|---|---|---|
| July, 2026 | 1.0 | ● Initial release |

# 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.