

应用笔记

Application Note

文档编号: **AN1185**

**G32R430 IAR 中断调用 ITCM 函数时的装载
限制与处理**

版本: **V1.0**

1 引言

G32R430 支持将部分函数放置到 ITCM (Instruction TCM, 指令紧耦合存储器, 属片上 RAM) 中执行, 以降低取指路径延迟。在 G32R430 DDL SDK 中, 可通过段属性 (如 **SECTION_ITCM_RAMFUNC**, 链接段名 `itcm.ramfunc`) 将函数放到 ITCM。

当前 IAR 工具链存在如下限制: 链接器会将中断向量表中引用的函数视为 **needed for initialization** (初始化阶段所需对象)。由此, 当向量表中的服务程序 (ISR) 进一步调用位于 RAM/TCM (含 ITCM 这类片上 RAM) 的函数时, 若仍依赖 **initialize by copy** 由 C 运行库自动完成 Flash→RAM/TCM 装载, 则相关装载依赖可能无法正确建立, 导致运行期执行到尚未有效装载的代码。本文所述 SDK 默认设计与配置方法, 即针对该限制给出**可实施**的处理方案。

本文说明:

- 当前 DDL SDK 在 IAR、Flash 启动场景下的默认设计及原因;
- 板级验证得到的启动周期数量级, 便于评估用户侧装载开销;
- 当中断路径**未**使用 ITCM 函数时, 如何改为 IAR 运行库装载。

目录

1	引言	1
2	原理	3
3	当前 SDK IAR 设计及原因.....	4
3.1	默认策略（推荐）	4
3.2	与整镜像在 ITCM 脚本的区别.....	4
3.3	拷贝实现说明	5
4	设计验证与测试数据.....	6
4.1	测试目的	6
4.2	测试环境.....	6
4.3	如何用 map 确认待拷贝字节数.....	7
4.4	测试关键代码段	8
4.5	结果摘要	9
4.6	数据解读边界	10
5	何时可用运行库装载以及如何配置	11
5.1	适用条件	11
5.2	改为运行库装载的步骤	11
5.3	恢复为用户侧装载	13
6	其它说明	14
6.1	MDK 与 Eclipse	14
6.2	优化与拷贝实现	14
6.3	其它注意	14
7	参考文献	15
8	版本历史	16

2 原理

Flash 启动时, ITCM (片上 RAM) 中的代码通常在 Flash 中保留一份装载镜像 (LMA), 上电后拷贝到 ITCM 执行地址 (VMA, 约 0x00000000 起) 后才能正确取指执行。

IAR 链接脚本中与装载相关的两种方式如下:

- **initialize by copy:** 由 IAR C 运行库在进入 main 前自动完成拷贝。
- **initialize manually:** 链接器生成装载镜像, 由用户在启动早期 (如 ItcmData_Init) 自行拷贝。

DDL SDK 中与 ITCM 相关的典型段包括: itcm.ramfunc、itcm.instruction、atan2_instruction。应用将函数放入这些段后, 必须保证在首次执行前完成 Flash→ITCM 装载。

Flash 启动下两条路径可概括为:

- **用户侧路径 (SDK 默认):** 复位 → SystemInit → ItcmData_Init (拷贝 ITCM 相关段) → C 库 (拷贝普通 RW 等) → main。
- **运行库路径 (可选):** 复位 → SystemInit → C 库 (含 ITCM 相关段的 initialize by copy) → main (不调用 ItcmData_Init)。

注意:

- (1) 用户侧路径适用于 ISR 可能调用 ITCM 函数的场景。
- (2) 运行库路径仅适用于中断向量可达路径不调用 ITCM 函数的场景, 详见第 5 章。

3 当前 SDK IAR 设计及原因

3.1 默认策略（推荐）

正式链接脚本路径:

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

默认配置如下:

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

即用户侧装载 (“__itcm_init_mode__ = 0”)。此时 ICF 对 ITCM 相关段使用 initialize manually, 并在 Device 启动文件 startup_g32r430xx.c 的 Reset_Handler 中, 在进入 C 库之前调用 ItcmData_Init(), 按字节将 Flash 中的装载镜像拷入 ITCM。

采用该默认策略的原因:

- IAR 链接器将中断向量表中引用的函数视为 needed for initialization。
- 当 ISR 继续调用位于 ITCM 等 RAM/TCM 区域的函数时, 仅依赖 initialize by copy 可能无法正确建立 Flash→ITCM 装载依赖。
- 对相关段使用 initialize manually, 并在进入 C 库之前由 ItcmData_Init 完成拷贝, 可保证 ISR→ITCM 调用路径在运行期可用。

默认配置一览见表格 1。

表格 1 SDK 默认 IAR ITCM 装载配置

项	默认值 / 行为
ICF 符号 __itcm_init_mode__	0 (用户侧)
ITCM 相关段	initialize manually
Device startup	调用 ItcmData_Init() (未定义 IAR_ITCM_AUTO_INIT 时)
ItcmData_Init 实现	字节循环拷贝
适用场景	ISR 可能调用 ITCM 函数; 以及需与 SDK 默认行为保持一致的工程

3.2 与整镜像在 ITCM 脚本的区别

g32r430xb_itcm_ram.icf 将只读代码直接放置在 ITCM 区域, 由调试器或下载流程把映像写入 ITCM, 不存在 Flash→ITCM 的段拷贝配置问题, 因此不需要 __itcm_init_mode__ 开关。

本文第 4、5 章均针对 Flash 启动且 ITCM 段需拷贝的 g32r430xb_flash.icf 场景。

3.3 拷贝实现说明

ItcmData_Init 在正式 SDK 中保持**字节拷贝**实现。启动阶段相对运行库装载会多出一定周期开销，且开销随 ITCM 待拷贝字节数增大而增加（见第 4 章）。用户工程在装载量较大时可自行评估的做法见第 6.2 节。

4 设计验证与测试数据

4.1 测试目的

本节验证目标：量化 IAR 用户侧 ItcmData_Init 相对其它场景进入 main 的**相对延迟**。

测时口径与常见 GPIO 启动测时例程一致：

1. SystemInit 执行完毕后清零并使能 DWT 周期计数（SystemInit 本身不计入）。
2. 若为 IAR 用户侧装载模式，则执行 ItcmData_Init。
3. 进入 C 库初始化。
4. 在 main 入口读取 CycleCount（DWT CYCCNT）。

相对延迟定义为：

```
relative_delay = CycleCount (IAR 用户侧) - CycleCount (对照场景)
```

正值表示用户侧装载进入 main 的相对延迟为正。 μs 按 8 MHz 估算： $\mu s \approx \text{周期数} / 8$ 。

4.2 测试环境

本次测试的板级、时钟、重复次数、功能场景、装载数据量与优化等级如下。

- 板级：G32R430 Tiny 评估板。
- 测时点核频约为 HSI 8 MHz（进入 main 读取 CycleCount 时，尚未切换到应用 PLL）。
- 每配置重复不少于 5 次，取中位数。
- 功能场景：SysTick ISR 调用放置在 ITCM 的函数（地址落在 ITCM 区间，例如 0x00000009），用于确认 ISR→ITCM 路径有效。
- ITCM 装载数据量：本次测试仅将 ITCM_ToggleLed 放入 itcm.ramfunc。Flash→ITCM 需拷贝的字节数以链接 map 为准，确认方法见 4.3，汇总见表格 2。用户工程若 ITCM 段更大，启动拷贝时间通常更长；表格 2、表格 3 中的 CycleCount / 相对延迟**仅对应该装载数据量**，不可直接外推到任意规模。
- 优化等级：O0 / O2 / Ofast（IAR 无字面 Ofast 时，映射为 None / High / High+Speed，报告中记为 IAR≈Ofast）。

本次测试所用 IDE 与编译器版本如下。

- IAR
 - 版本：IAR Embedded Workbench for Arm 9.60.2

- 工具链: IAR C/C++ Compiler for Arm (与 EWARM 9.60.2 配套; 链接器 map 标识 V9.60.2)
- MDK
 - 版本: Keil MDK 5.43 (µVision)
 - 工具链: Arm Compiler for Embedded 6.24 (ArmClang / armlink)
- Eclipse
 - 版本: Eclipse IDE 4.35.0 RC1 + Embedded CDT
 - 工具链: LLVM Embedded Toolchain for Arm 19.1.5 (clang 19.1.5)

4.3 如何用 map 确认待拷贝字节数

用户可用链接 map 自行核对本工程 Flash→ITCM 的待拷贝数据量。步骤如下:

1. 在 IAR 中对目标配置执行 Rebuild, 打开生成的 map 文件 (通常位于工程配置目录下的 List 文件夹, 例如 IAR_ITCM_ManualInit.map)。
2. 在 map 中搜索 itcm.ramfunc、itcm.ramfunc_init, 以及若工程使用了 itcm.instruction、atan2_instruction 时对应的 _init 名称。
3. 用户侧装载 (SDK 默认, ItcmData_Init): 以各段 Flash 侧装载镜像块的 Size 为准。对 itcm.ramfunc, 查找名为 itcm.ramfunc_init 的块, 或其下 Initializer bytes (标注为 for itcm.ramfunc) 一行的 Size; 亦可在 ENTRY LIST 中用“itcm.ramfunc_init\$\$Limit - itcm.ramfunc_init\$\$Base”得到字节数。若同时存在 itcm.instruction_init、atan2_instruction_init, 应将各 init 块 Size 相加, 即为 ItcmData_Init 的总拷贝量。
4. 运行库装载 (initialize by copy): 可在 PLACEMENT SUMMARY 中查看 itcm.ramfunc (及同类 ITCM 段) 的 Size; 该值反映放入 ITCM 的执行映像规模, 供与用户侧路径对照。

以本工程最新 map 为准; 优化等级、函数增减都会改变 Size, 装载量变化后须重新评估启动时间 (见表格 2)。

下列摘录来自本次测试 IAR 用户侧装载、O0 配置的 map (节选)。其中 itcm.ramfunc_init 的 Size 为 0x34 (十进制 52), 即表格 2 中 IAR 用户侧 O0 的装载字节数。

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 -

```

说明:

- (1) 上述摘录中, 执行区 itcm.ramfunc 函数体为 0x2c, Init block / itcm.ramfunc_init 为 0x34 (含链接器生成的 Veneer 等), ItcmData_Init 按 init 块拷贝, 故取 0x34。
- (2) 本次测试未使用 itcm.instruction、atan2_instruction, 故无对应 _init 行; 用户工程若有, 须一并累加。

4.4 测试关键代码段

下列摘录用于说明验证工程如何构造「ISR→ITCM」与测时窗 (与板测数据对应)。用户工程可按同样结构放置 ITCM 函数, 并按第 3、5 章选择用户侧或运行库装载。启动文件中: 未定义 IAR_ITCM_AUTO_INIT 时调用 ItcmData_Init; 定义该宏时跳过 (见表格 4)。

将函数放入 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);
  }
}

```

向量表 ISR 调用 ITCM 函数:

```

void SysTick_Handler(void)
{
  ITCM_ToggleLed();
}

```

复位后建立测时窗, 并在用户侧装载模式下调用 ItcmData_Init (进入 C 库之前):

```

SystemInit();

```

```
/* 清零并使能 DWT CYCCNT (SystemInit 不计入 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();
```

在 main 入口采样 CycleCount:

```
int main(void)
{
    CycleCount = (*(volatile uint32_t *)0xE0001004U);
    /* ... 后续时钟 / USART / 外设初始化 ... */
}
```

说明:

功能验证通过时, ITCM 函数地址应落在 ITCM 区间 (本次测试约 0x00000009)。

4.5 结果摘要

本次测试各环境的 ITCM 装载字节数 (摘自 map) 与进入 main 前 CycleCount (中位数, n=5) 合并见表格 2。每格为「装载字节 / CycleCount」; 同一环境将 O0 / O2 / Ofast 列在一行。CycleCount 仅对应该格左侧字节数。

表格 2 本次测试 ITCM 装载字节数与进入 main 前 CycleCount (中位数, n=5)

序号	环境	O0 (字节 / 周期)	O2 (字节 / 周期)	Ofast (字节 / 周期)
1	IAR 用户侧	52 / 2556	56 / 1451	56 / 2727
2	IAR 运行库	44 / 491	48 / 509	48 / 553
3	MDK	58 / 1214	58 / 1500	58 / 1505
4	Eclipse	60 / 2891	100 / 956	100 / 982

说明:

- (1) “IAR 用户侧”一行的字节数取自 map 中的 itcm.ramfunc_init, 即 ItcmData_Init 实际拷贝的字节数。
- (2) 表中 Ofast 列在 IAR 下对应优化等级 High 且偏向速度 (High+Speed); IAR 无字面 Ofast, 映射关系

见第 4.2 节。

- (3) IAR 用户侧以 init 块为准；该块可能略大于函数正文，原因包括对齐以及链接器生成的其它内容。
- (4) 优化等级会改变生成代码的体积，因此同一环境下不同优化等级的字节数可以不同。
- (5) 用户应以本工程最新 map 复核；ITCM 待拷贝字节数变化后，须重新评估启动时间。

IAR 用户侧相对其它场景的延迟见表格 3。

表格 3 IAR 用户侧相对对照场景的延迟（用户侧 - 对照场景）

优化等级	相对 IAR 运行库	相对 MDK	相对 Eclipse
O0	+2065（约 258 μs）	+1342（约 168 μs）	-335（约 -42 μs）
O2	+942（约 118 μs）	-49（约 -6 μs）	+495（约 62 μs）
Ofast	+2174（约 272 μs）	+1222（约 153 μs）	+1745（约 218 μs）

结论要点：

- 相对 IAR 运行库装载，在本次测试 ITCM 装载量（约 52~56 字节，见表格 2 序号 1）下，用户侧装载的相对延迟约为 900~2200 周期，即约 0.12~0.27 ms（按 8 MHz 估算）。
- 若工程几乎无 ITCM 待拷贝内容（例如简单 GPIO 翻转例程），IAR 与 MDK 的 CycleCount 可能仅差数十周期；这与「ISR→ITCM 且确有装载负载」场景不可直接类比。
- ITCM 待拷贝字节数增大时，用户侧与运行库路径的启动耗时通常都会上升；请以本工程 map 中的实际段长重新评估（装载量与周期见表格 2）。
- 在本次测试较小装载量下，IAR≈Ofast 的 CycleCount 可高于 O2，解读见第 6.2 节。

4.6 数据解读边界

- 跨工具链的 CycleCount 包含各家 C 库与启动路径差异，不能简单解读为拷贝算法效率的高低。
- ITCM 待拷贝量接近为空时的微小差异，**不能**用来否定 ISR→ITCM 场景下采用用户侧装载的必要性。
- 表格 2、表格 3 对应同一次板测矩阵；具体数值随 ITCM 段大小、优化选项与库版本会变化，应以同条件复测为准。

5 何时可用运行库装载以及如何配置

5.1 适用条件

在同时满足下列条件时，可改用 IAR 运行库装载（“__itcm_init_mode__ = 1”）：

- 没有任何中断向量可达路径调用位于 ITCM 的函数（含间接调用）。
- ITCM 中的函数仅由 main 或线程上下文调用，或工程根本不使用 ITCM 代码段。

出现下列情况时，应保持 SDK 默认的用户侧装载：

- ISR，或任何被向量表引用的 handler，会调用 ITCM 中的函数。

注意：

- （1）在 ISR→ITCM 场景下误启用运行库装载，可能导致运行异常或偶发故障。
- （2）修改装载方式后，必须重新全编译，并做功能与启动回归。

5.2 改为运行库装载的步骤

1. 打开工程实际使用的 g32r430xb_flash.icf（可为 SDK Device 路径下的脚本，或工程目录中的副本）。
2. 将 __itcm_init_mode__ 由 0（用户侧装载，默认）改为 1（运行库装载）。例如把：

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

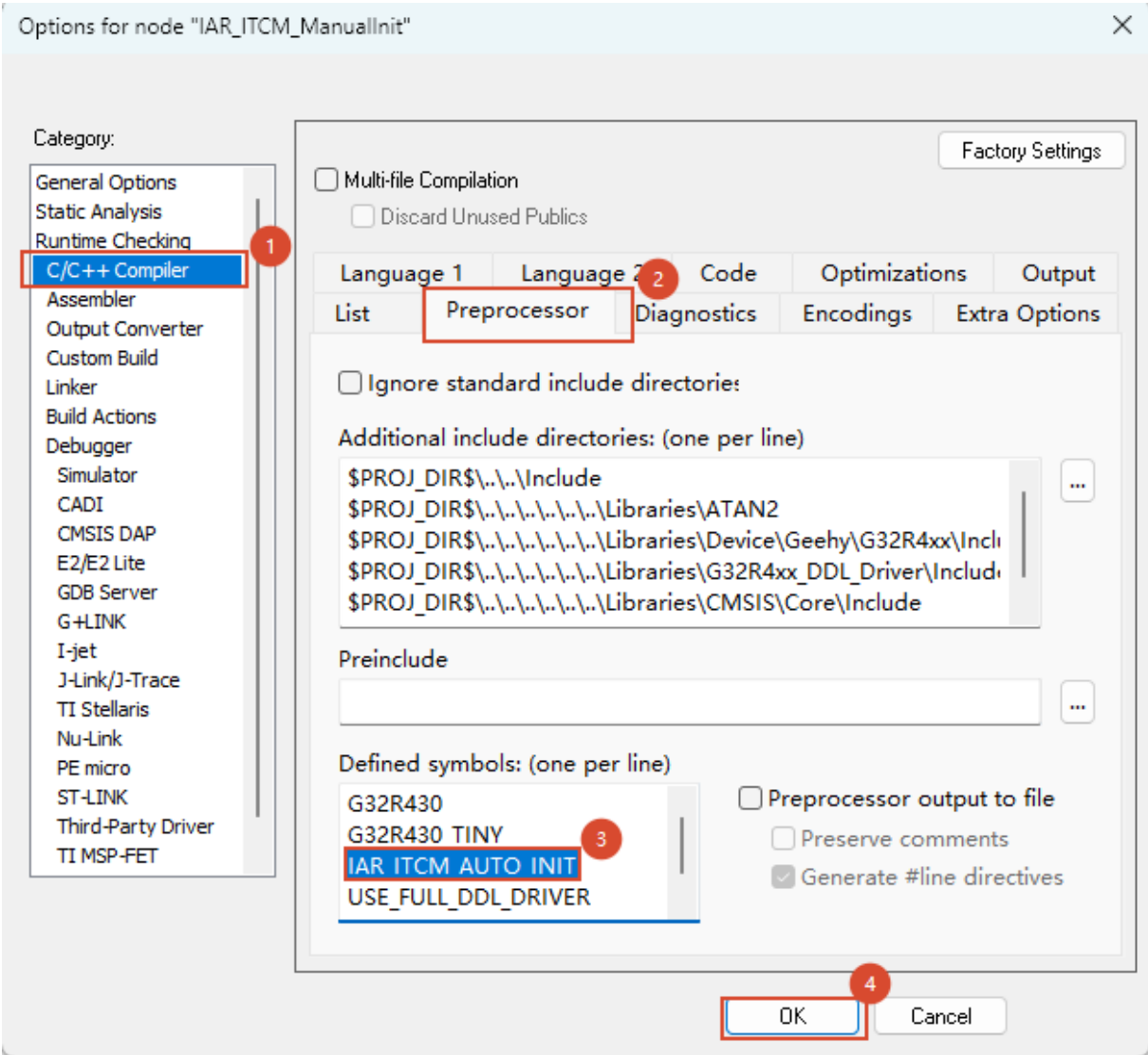
修改为：

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

3. 在 IAR 工程选项中增加预定义宏 IAR_ITCM_AUTO_INIT，使 Device startup 跳过 ItcmData_Init()，避免与 C 库自动拷贝重复或冲突。操作如图 1 所示（对应图中①～④）：
 - 在 Category 中选择 **C/C++ Compiler**
 - 打开 **Preprocessor** 页签
 - 在 **Defined symbols**（每行一个宏定义）中新增 IAR_ITCM_AUTO_INIT
 - 单击 **OK** 确认

如图 1 所示，在 IAR 工程选项中添加预定义宏 IAR_ITCM_AUTO_INIT。

图 1 在 IAR 工程选项中添加预定义宏 IARITCM_AUTOINIT



4. 执行 Rebuild、下载并运行。确认：

- 若仍有 ITCM 函数，其地址位于 ITCM 区间；
- 业务功能与中断行为符合预期。

用户侧与运行库配置对照见表格 4。

表格 4 用户侧 / 运行库装载配置检查清单

检查项	用户侧 (默认)	运行库
__itcm_init_mode__	0	1
工程宏 IAR_ITCM_AUTO_INIT	不定义	必须定义
ItcmData_Init	调用	跳过

检查项	用户侧 (默认)	运行库
ITCM 段装载责任方	startup 用户侧拷贝	IAR C 库 initialize by copy
ISR→ITCM	推荐	不推荐

注意:

__itcm_init_mode__ 与 IAR_ITCM_AUTO_INIT 必须成对修改, 不可只改其中一项。

5.3 恢复为用户侧装载

1. 将 __itcm_init_mode__ 恢复为 0。
2. 从工程预定义宏中删除 IAR_ITCM_AUTO_INIT。
3. Rebuild 并回归验证。

6 其它说明

6.1 MDK 与 Eclipse

Keil MDK (scatter-load) 与 Eclipse/Clang (copy table) 在「向量表 ISR 调用 ITCM 函数」场景下, 一般不存在与 IAR 相同的 initialize by copy 限制。这两类工程按 SDK 提供的分散加载脚本或链接脚本完成 RW/ITCM 装载即可。跨工具链的启动周期差异主要来自 C 库与拷贝实现, 解读时参见第 4.6 节。

6.2 优化与拷贝实现

正式 SDK 默认的 ItcmData_Init 为**字节拷贝**。不建议仅将优化等级改为 Ofast, 作为缩短拷贝耗时的唯一手段; 在本次测试较小装载量下, Ofast 对小拷贝循环未必总能缩短拷贝耗时。

若用户工程 ITCM 待拷贝内容较大、且启动时间成为瓶颈, 可在保持正确装载顺序的前提下, 自行评估将 ItcmData_Init 改为**按字 (或按对齐宽度) 拷贝**, 并配合较高优化等级 (如 O2 / High), 以降低拷贝开销; 修改后须做功能与启动回归。默认交付仍以 SDK 字节拷贝实现为准。

6.3 其它注意

- 向量表重映射、二次 Boot、IAP 多映像等场景下的 ITCM 段归属与装载时机需单独评估, 超出本文范围。
- 本文测时与功能验证基于 Flash 启动 + ISR→ITCM 对照工程完成; 用户工程请按本章检查清单配置, 并结合自身 ITCM 段大小复测启动时间。

7 参考文献

《IAR 在 TCM/RAM 中运行中断服务程序的问题》：

<https://mp.weixin.qq.com/s/jRuLkSzGWEQ1mB-Z65Mpw>

8 版本历史

文件版本历史见表格 5。

表格 5 文件版本历史

日期	版本	变更说明
2026.7	1.0	● 初始版本

声明

本手册由珠海极海半导体有限公司（以下简称“极海”）制订并发布，所列内容均受商标、著作权、软件著作权相关法律法规保护，极海保留随时更正、修改本手册的权利。使用极海产品前请仔细阅读本手册，一旦使用产品则表明您（以下称“用户”）已知悉并接受本手册的所有内容。用户必须按照相关法律法规和本手册的要求使用极海产品。

1、权利所有

本手册仅应当被用于与极海所提供的对应型号的芯片产品、软件产品搭配使用，未经极海许可，任何单位或个人均不得以任何理由或方式对本手册的全部或部分内容进行复制、抄录、修改、编辑或传播。

本手册中所列带有“®”或“™”的“极海”或“Geehy”字样或图形均为极海的商标，其他在极海产品上显示的产品或服务名称均为其各自所有者的财产。

2、无知识产权许可

极海拥有本手册所涉及的全部权利、所有权及知识产权。

极海不应因销售、分发极海产品及本手册而被视为将任何知识产权的许可或权利明示或默示地授予用户。

如果本手册中涉及任何第三方的产品、服务或知识产权，不应被视为极海授权用户使用前述第三方产品、服务或知识产权，也不应被视为极海对第三方产品、服务或知识产权提供任何形式的保证，包括但不限于任何第三方知识产权的非侵权保证，除非极海在销售订单或销售合同中另有约定。

3、版本更新

用户在下单购买极海产品时可获取相应产品的最新版的手册。

如果本手册中所述的内容与极海产品不一致的，应以极海销售订单或销售合同中的约定为

准。

4、信息可靠性

本手册相关数据经极海实验室或合作的第三方测试机构批量测试获得，但本手册相关数据难免会出现校正笔误或因测试环境差异所导致的误差，因此用户应当理解，极海对本手册中可能出现的该等错误无需承担任何责任。本手册相关数据仅用于指导用户作为性能参数参照，不构成极海对任何产品性能方面的保证。

用户应根据自身需求选择合适的极海产品，并对极海产品的应用适用性进行有效验证和测试，以确认极海产品满足用户自身的需求、相应标准、安全或其它可靠性要求；若因用户未充分对极海产品进行有效验证和测试而致使用户损失的，极海不承担任何责任。

5、合规要求

用户在使用本手册及所搭配的极海产品时，应遵守当地所适用的所有法律法规。用户应了解产品可能受到产品供应商、极海、极海经销商及用户所在地等各国有关出口、再出口或其它法律的限制，用户（代表其本身、子公司及关联企业）应同意并保证遵守所有关于取得极海产品及/或技术与直接产品的出口和再出口适用法律与法规。

6、免责声明

本手册由极海“按原样”（as is）提供，在适用法律所允许的范围内，极海不提供任何形式的明示或暗示担保，包括但不限于对产品适销性和特定用途适用性的担保。

极海产品并非设计、授权或担保适合用于军事、生命保障系统、污染控制或有害物质管理系统中的关键部件，亦非设计、授权或担保适合用于在产品失效或故障时可导致人员受伤、死亡、财产或环境损害的应用。

如果产品未标明“汽车级”，则表示不适用于汽车应用。如果用户对产品的应用超出极海提供的规格、应用领域、规范，极海不承担任何责任。

用户应该确保对产品的应用符合相应标准以及功能安全、信息安全、环境标准等要求。用户对极海产品的选择和使用负全部的责任。对于用户后续在针对极海产品进行设计、使用的过程中

所引起的任何纠纷，极海概不承担责任。

7、责任限制

在任何情况下，除非适用法律要求或书面同意，否则极海和/或以“按原样”形式提供本手册及产品的任何第三方均不承担损害赔偿 responsibility，包括任何一般、特殊因使用或无法使用本手册及产品而产生的直接、间接或附带损害（包括但不限于数据丢失或数据不准确，或用户或第三方遭受的损失），这涵盖了可能导致的人身安全、财产或环境损害等情况，对于这些损害极海概不承担责任。

8、适用范围

本手册的信息用以取代本手册所有早期版本所提供的信息。

©2026 珠海极海半导体有限公司 – 保留所有权利