GigaDevice Semiconductor Inc.

GD32F5xx 系列双 bank 切换

应用笔记

AN195

1.0 版本

（2024 年 4 月）

目录

图索引

表索引

1. 前言

本文是专为 GD32F5xx 双 bank Flash 产品提供，介绍了双 bank 代码更新及切换启动的功能。本应用笔记总共分为两部分来讲述，第一部分介绍了双 bank 切换的实现原理，第二部分介绍了软件实现方法。

- 开发板：GD32F527I-EVAL板
- 调试器：J-Link或GD-Link
- IDE：Keil 5.35

2. 双 bank 切换原理

GD32F5xx 闪存结构分为 4MB 双块、2MB 双块、1MB 单块、512KB 单块结构，每种结构均有 bank1 拓展闪存（Bank1_Ex），bank1 拓展闪存地址从 0x08400000 开始且操作方式与 bank1 相同。当为 GD32F5xx 双块结构时，前 2048KB 容量在第 0 片闪存（Bank0）中，后续的容量在第 1 片闪存（Bank1 和 Bank1_Ex）中。MCU 执行指令零等待区域最大支持到前 2048K 字节空间（在闪存大小小于 2048KB 时，闪存全片执行指令零等待），在此范围外，CPU 读取指令存在较长延时。本文使用的是 4MB 双块产品。

SYSCFG_CFG0 寄存器提供了 FMC_SWP 位，该位控制主闪存存储器的 bank0 和 bank1 的地址映射切换功能。该位为 0 时，主闪存存储器的 bank0 被映射到地址 0x08000000，主闪存存储器的 bank1 被映射到地址 0x08100000（2M 的是交换 0x08100000，4M 的是交换 0x08200000）。该位为 1 时，主闪存存储器的 bank1 映射到地址 0x0800 0000，主闪存存储器的 bank0 映射到地址 0x08100000（2M 的是交换 0x08100000，4M 的是交换 0x08200000）。注意在 SYSCFG 中设置 FMC_SWP 位将交换总线矩阵中的 bank0 和 bank1 逻辑地址，但不影响原始擦除地址。

选项字节中提供了 BB 位，该位为 0 时（出厂值），当 MCU 配置从主存储块启动时，从 bank0 启动。该位置 1 时，当 MCU 配置从主存储块启动时，若 bank1 无启动程序，从 bank0 启动，若 bank1 有程序，则从 bank1 启动。内部实现原理为：MCU 会在从主闪存存储器启动时，系统 bootloader 检测 BB 位是否置位，若 BB 位置位，则将 FMC_SWP 位置位，并检测 bank1 有无代码，若 bank1 有代码，则从 bank1 启动（逻辑地址仍为 0x08000000）。

选择字节中同时提供了 NWA 位，该位为 0 时，选择零等待区域为 bank1。该位置 1 时（出厂值），选择零等待区为 bank0。该位仅在电源复位后生效，且仅在 4MB 双块产品中有效。通过配置该位可灵活配置零等待区，提高代码执行效率。

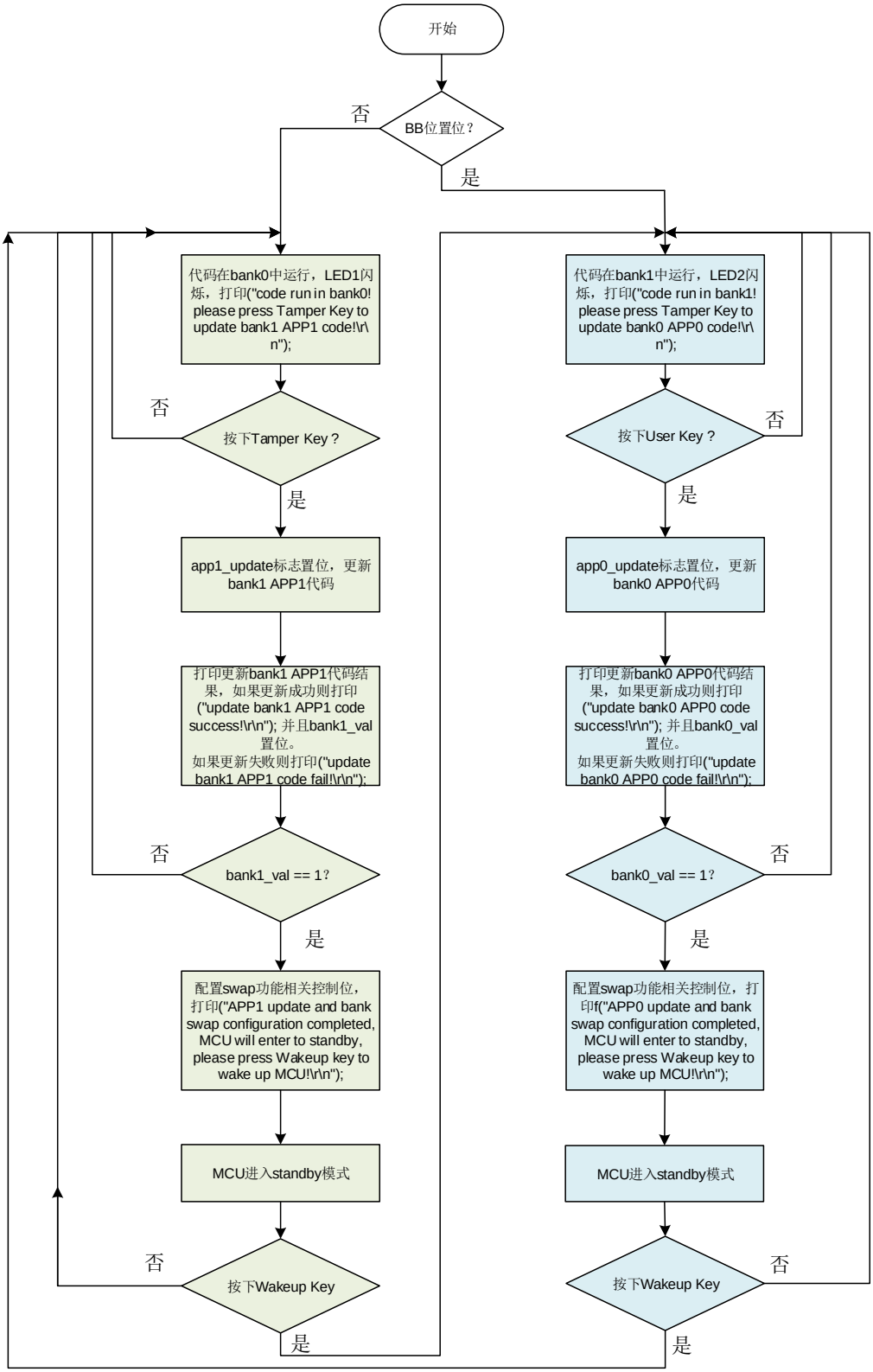
根据以上 GD32F5xx 的特性，GD32F5xx 可以实现在 bank0 运行代码时，更新 bank1 代码，待 bank1 代码更新完成后，通过配置选项字节 BB 位及 NWA 位，断电复位后实现切换至 bank1 代码运行的功能。在 bank1 运行代码时，更新 bank0 代码，待 bank0 代码更新完成后，通过配置选项字节 BB 位及 NWA 位，断电复位后实现切换至 bank0 代码运行的功能。

3. 软件实现

3.1. 软件流程

在此软件实现方案中，预先将 bank0 APP0 和 bank1 APP1 的两套代码存储在外部 Nand Flash 中。软件检测到对应的按键按下后，将 Nand Flash 中的代码搬运到 bank0 或 bank1。代码搬运完成后，软件配置选项字节 BB 位及 NWA 位，并通过使 MCU 进入 standby 后唤醒操作，使得选项字节 BB 位及 NWA 位生效。具体实现流程如*图 3-1. 软件流程*所示。

图 3-1. 软件流程图


开始
BB位置位？
否
是
代码在bank0中运行，LED1闪烁，打印("code run in bank0! please press Tamper Key to update bank1 APP1 code!\r\n");
按下Tamper Key？
否
是
app1_update标志置位，更新bank1 APP1代码
打印更新bank1 APP1代码结果，如果更新成功则打印("update bank1 APP1 code success!\r\n"); 并且bank1_val置位。
如果更新失败则打印("update bank1 APP1 code fail!\r\n");
bank1_val == 1？
否
是
配置swap功能相关控制位，打印("APP1 update and bank swap configuration completed, MCU will enter to standby, please press Wakeup key to wake up MCU!\r\n");
MCU进入standby模式
按下Wakeup Key
否
是
代码在bank1中运行，LED2闪烁，打印("code run in bank1! please press Tamper Key to update bank0 APP0 code!\r\n");
按下User Key？
否
是
app0_update标志置位，更新bank0 APP0代码
打印更新bank0 APP0代码结果，如果更新成功则打印("update bank0 APP0 code success!\r\n"); 并且bank0_val置位。
如果更新失败则打印("update bank0 APP0 code fail!\r\n");
bank0_val == 1？
否
是
配置swap功能相关控制位，打印f("APP0 update and bank swap configuration completed, MCU will enter to standby, please press Wakeup key to wake up MCU!\r\n");
MCU进入standby模式
按下Wakeup Key
否
是

3.2. 实验步骤及现象

1. 使用宏 APP0、APP1 分别编译出对应的 app0.bin 和 app1.bin 文件。app0.bin 和 app1.bin 分别对应 bank0 和 bank1 的代码。编译时请注意 Linker 选项使用 Template\Keil_project 路径下提供的 Project.sct 文件，*图 3-3. Keil 工程 Linker 选项*。在 Keil 工程 User 选项中配置工程编译完成后生成 bin 文件命令，见*图 3-4. Keil 工程 User 选项*。路径需选择本地 Keil 安装路径。

图 3-2. Keil 工程预编译宏定义

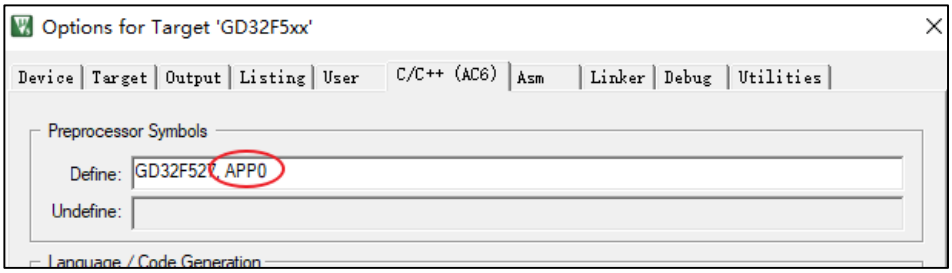


图 3-3. Keil 工程 Linker 选项

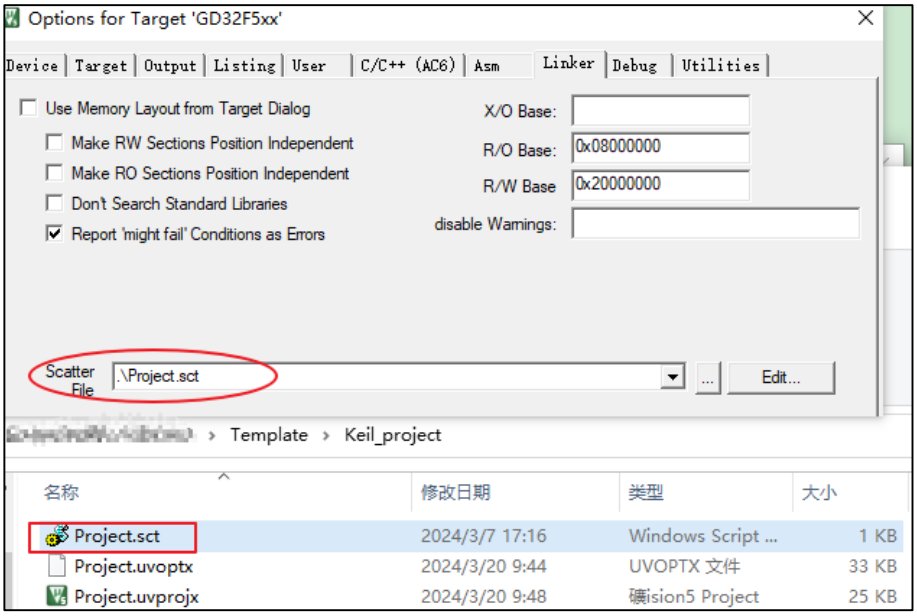
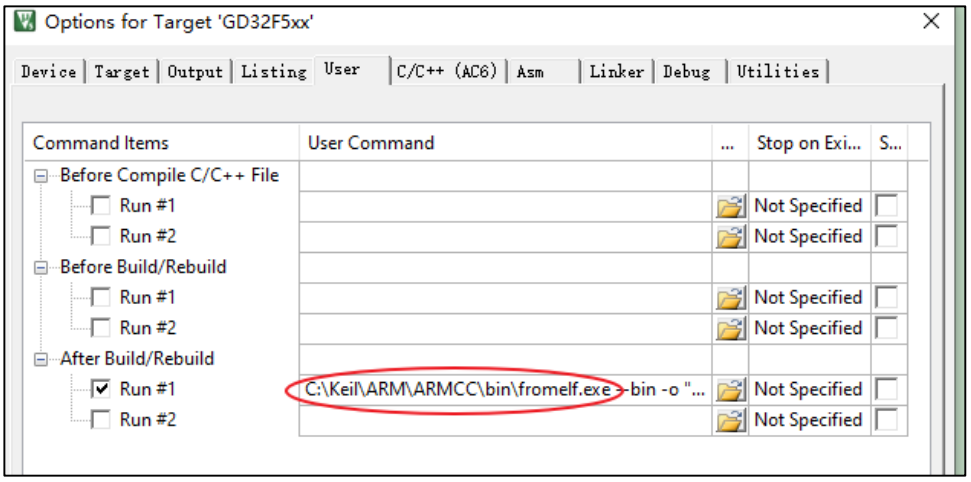
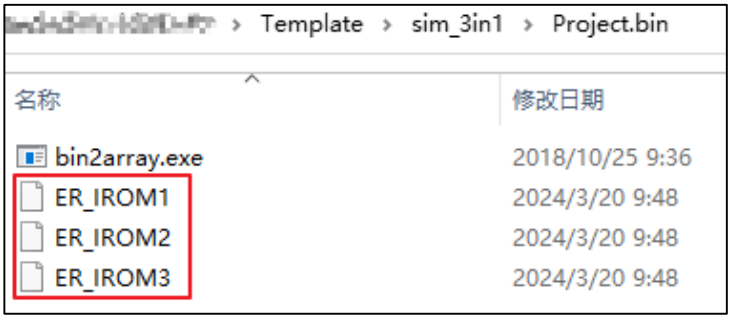


图 3-4. Keil 工程 User 选项



2. 工程编译完成后，在 Template\sim_3in1\Project.bin 文件夹内生成 3 段 bin，分别为 ER_IROM1、ER_IROM2、ER_IROM3。

图 3-5.编译生成 Project.bin 文件夹



当使用的预编译宏为 APP0，编译完工程后将 ER_IROM1 名称改为 app0.bin，并使用 bin2array.exe 软件将 app0.bin 转为 app0.txt 文件，将 app0.txt 中数组放入 app0.h 文件的 app0_code[]数组中，见*表 3-1. app0.bin 定义*。同样方法生成 app1.bin，并放入 app1.h 文件的 app1_code[]数组中，见*表 3-2. app1.bin 定义*。

图 3-6.生成 app0.bin 文件

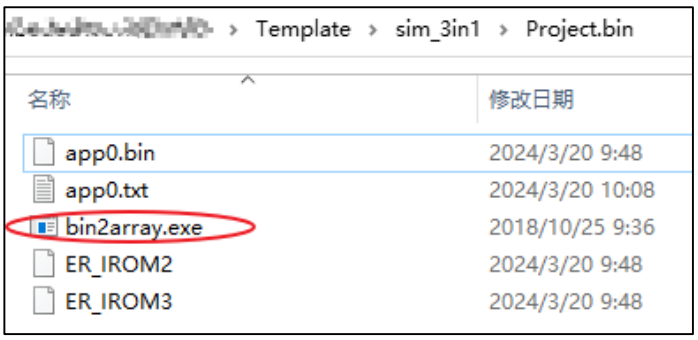


表 3-1. app0.bin 定义

const uint8_t app0_code[]__attribute__((used))__attribute__((section ("APP0_Array")))=

```
{0xB0,0x0C,0x00,0x20,0xF5,0x01,0x00,0x08,0xC5,0x07,0x00,0x08,0xBD,0x07,0x00,0x08,
…};
```

表 3-2. app1.bin 定义

```
const uint8_t app1_code[]__attribute__((used))__attribute__((section ("APP1_Array")))=
{0xB0,0x0C,0x00,0x20,0xF5,0x01,0x00,0x08,0xC5,0x07,0x00,0x08,0xBD,0x07,0x00,0x08,
…};
```

3. 使用分散加载的方式将 app0.bin 和 app1.bin 文件加载到 Nand Flash 中。这样操作可以在工程代码下载时，同时烧写主闪存及 Nand Flash。

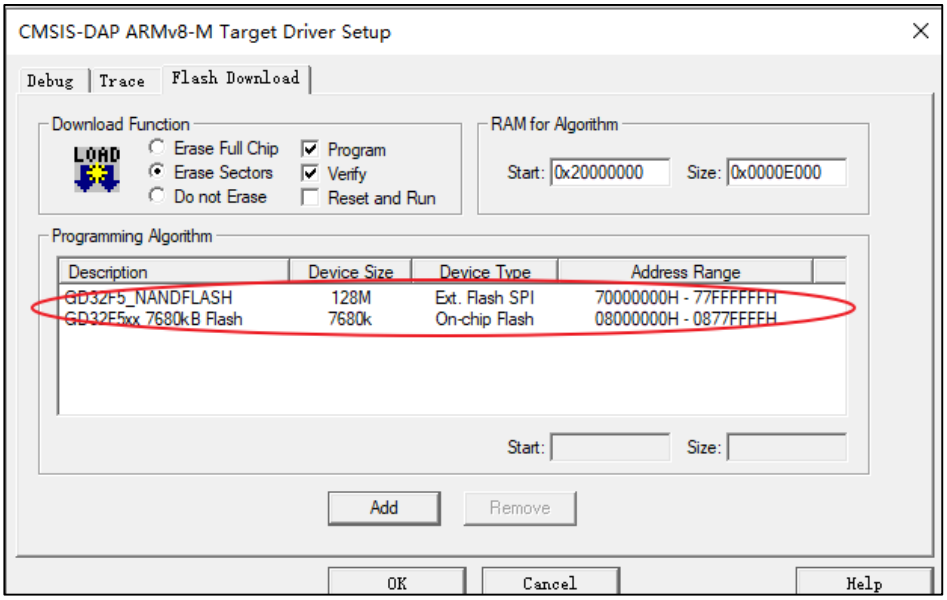
表 3-3. 分散加载文件

```
; *************************************************************
; *** Scatter-Loading Description File generated by uVision ***
; *************************************************************

LR_IROM1 0x08000000 0x00780000  {      ; load region size_region
  ER_IROM1 0x08000000 0x00780000  {  ; load address = execution address
   *.o (RESET, +First)
   *(InRoot$$Sections)
   .ANY (+RO)
   .ANY (+XO)
  }
  RW_IRAM1 0x20000000 0x00080000  {  ; RW data
   .ANY (+RW +ZI)
  }
}

LR_IROM2 0x70000000 0x2000000  {
  ER_IROM2 0x70000000 0x2000000  {  ; load address = execution address
   *(APP0_Array)
  }
}

LR_IROM3 0x72000000 0x6000000  {
  ER_IROM3 0x72000000 0x6000000  {  ; load address = execution address
   *(APP1_Array)
  }
}
```

4. 完成以上操作后，将宏定义选择为 APP0，编译并将代码下载到主闪存及 Nand Flash 中。此处需要预先制作 GD32F5_NANDFLASH 下载算法，并将下载算法放置在 Keil 安装路径下，此例中为 C:\Keil_v535\ARM\Flash。在 Keil “Options->Debug->Settings->Flash Download”选项中添加 GD32F5xx Flash 下载算法和 GD32F5_NANDFLASH 下载算法，见*图 3-7. Keil 工程下载算法配置*。请注意下载代码前请保持选项字节 BB 位及 NWA 位为

出厂默认值，及 BB 位为 0，NWA 位为 1。若下载代码时遇到错误，请先全片擦除后再下载。

图 3-7. Keil 工程下载算法配置



5. 代码下载完成后，复位 MCU，代码开始运行。串口输出运行信息及操作提示。

   运行 bank0 代码时，LED1 闪烁，串口输出“code run in bank0! please press Tamper Key to update bank1 APP1 code!”。

   用户按下 Tamper 键时，软件更新 bank1 APP1 代码。更新完成后，串口输出“update bank1 APP1 code success! APP1 update and bank swap configuration completed, MCU will enter to standby, please press Wakeup key to wake up MCU!”。MCU 完成 bank 切换，进入 standby 模式。

   此时若用户按下 Wakeup 键，切换至 bank1 代码运行，串口输出“code run in bank1! please press User Key to update bank0 APP0 code!”，LED2 闪烁。

   用户按下 User 键，软件更新 bank0 APP0 代码。更新完成后，串口输出“update bank0 APP0 code success! APP0 update and bank swap configuration completed, MCU will enter to standby, please press Wakeup key to wake up MCU! ”。MCU 完成 bank 切换，进入 standby 模式。

   此时若用户按下 Wakeup 键，MCU 切换至 bank0 代码运行，串口输出“code run in bank0! please press Tamper Key to update bank1 APP1 code!”，LED1 闪烁。

图 3-8. 测试结果

```
[2024-03-18 14:05:06.877]# RECV ASCII>
code run in bank0! please press Tamper Key to update bank1 APP1 code!


[2024-03-18 14:05:13.749]# RECV ASCII>
update bank1 APP1 code success!
APP1 update and bank swap configuration completed, MCU will enter to standby,
please press Wakeup key to wake up MCU!


[2024-03-18 14:05:17.612]# RECV ASCII>
code run in bank1! please press User Key to update bank0 APP0 code!


[2024-03-18 14:05:23.735]# RECV ASCII>
update bank0 APP0 code success!
APP0 update and bank swap configuration completed, MCU will enter to standby,
please press Wakeup key to wake up MCU!

[2024-03-18 14:05:26.589]# RECV ASCII>
code run in bank0! please press Tamper Key to update bank1 APP1 code!
```

4. 版本历史

表 4-1. 版本历史

版本号.	说明	日期
1.0	首次发布	2024 年 4 月 23 日

Important Notice

This document is the property of GigaDevice Semiconductor Inc. and its subsidiaries (the "Company"). This document, including any product of the Company described in this document (the “Product”), is owned by the Company under the intellectual property laws and treaties of the People’s Republic of China and other jurisdictions worldwide. The Company reserves all rights under such laws and treaties and does not grant any license under its patents, copyrights, trademarks, or other intellectual property rights. The names and brands of third party referred thereto (if any) are the property of their respective owner and referred to for identification purposes only.

The Company makes no warranty of any kind, express or implied, with regard to this document or any Product, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Company does not assume any liability arising out of the application or use of any Product described in this document. Any information provided in this document is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Except for customized products which has been expressly identified in the applicable agreement, the Products are designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only. The Products are not designed, intended, or authorized for use as components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, atomic energy control instruments, combustion control instruments, airplane or spaceship instruments, transportation instruments, traffic signal instruments, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or Product could cause personal injury, death, property or environmental damage ("Unintended Uses"). Customers shall take any and all actions to ensure using and selling the Products in accordance with the applicable laws and regulations. The Company is not liable, in whole or in part, and customers shall and hereby do release the Company as well as it’s suppliers and/or distributors from any claim, damage, or other liability arising from or related to all Unintended Uses of the Products. Customers shall indemnify and hold the Company as well as it’s suppliers and/or distributors harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of the Products.

Information in this document is provided solely in connection with the Products. The Company reserves the right to make changes, corrections, modifications or improvements to this document and Products and services described herein at any time, without notice.