S32K1xx系列MCU应用指南之EEPROM模块使用详解

版本管理

版本	修改日期	修改内容
V1.0	2019/7/9	初始版本，包含完整应用指南内容，内部 release for team review。
V1.1	2019/7/12	增加 SDK 中 flash 组件配置详解；优化各级标题，统一格式；增加 6.3 节 Flex RAM 启动加载示例及说明

S32K1xx 系列 MCU 应用指南之 EEPROM 模块使用详解

1. 介绍

NXP 提供的 MCU 有三种方法实现 EEPROM 的功能。一个是使用真正的 EEPROM，如 KEA64，具有 256 B EEPROM。优点是它不占用 Flash；但缺点是价格很贵。第二种方法是使用软件实现 EEPROM 功能，如 KEA8 和 KEA128。优点是便宜；但缺点是它占用 Flash，包括用于 EEPROM 实现的额外代码和用于模拟 EEPROM 的 Flash。第三种方法是使用固件实现 EEPROM 功能，如 S32K1xx。第三种方法的优点是不需要 Flash 存储用于 EEPROM 实现的额外代码。

本应用笔记主要介绍 S32K1xx EEPROM 功能特性和用法。 S32K1xx EEPROM 允许用户将片上某些 Flash 全部或部分配置为模拟 EEPROM。

有关本应用笔记中提到的任何参考资料的详细信息，请参阅 S32K1xx Reference Manual 和 Datasheet。

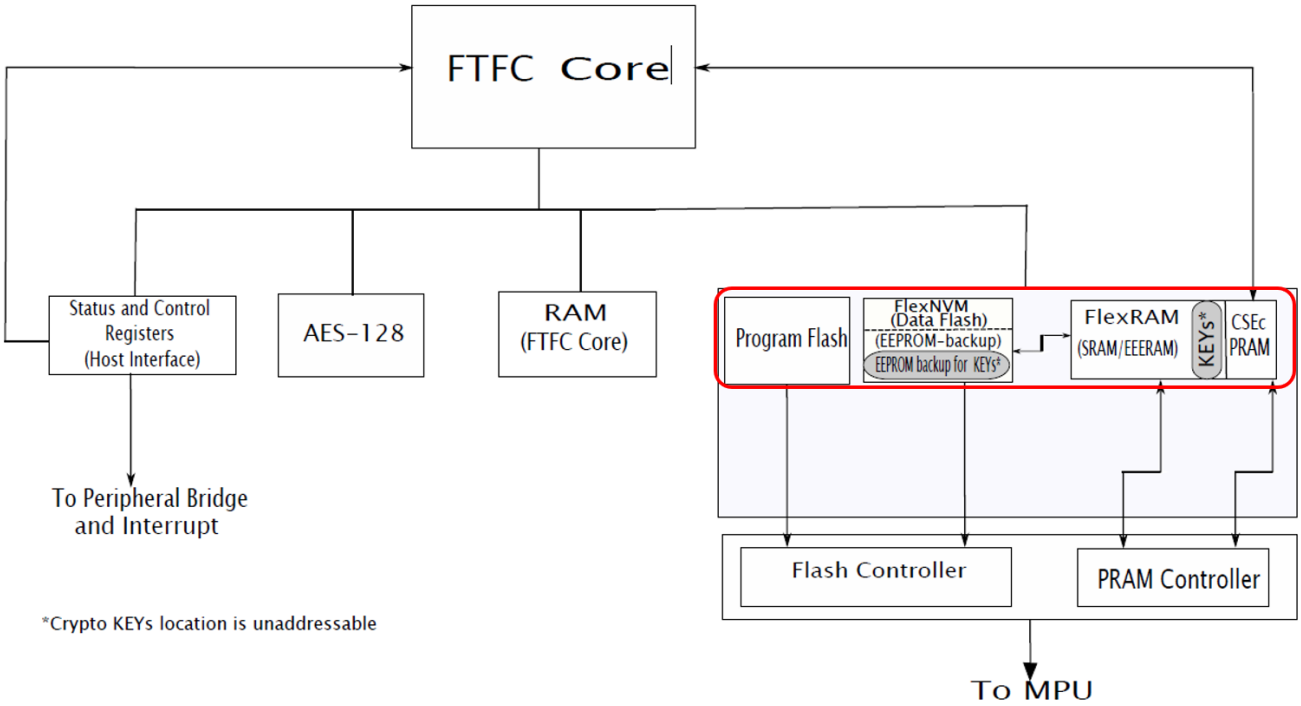
2. S32K1xx系列MCU的 EEPROM 模块介绍

2.1 S32K1xx系列MCU EEPROM模块功能特性

S32K1xx EEPROM 具有许多的功能，包括:

- 不需要客户开发 EEPROM 的驱动软件
  - 复位：EEPROM(E-Flash)记录的数据自动加载到 FlexRAM 中
  - 读取：直接从 FlexRAM 中读取记录，不涉及任何的 EEPROM 读操作；读操作包括：8bit，16bit 和 32bit
  - 写入：直接将数据写入 FlexRAM 的记录中，记录的数据自动同步到 EEPROM 中；写操作包括：8bit，16bit 和 32bit
- 利用循环均衡负载的方法：优化 EEPROM 的可靠性，提高擦写次数和使用寿命
- 支持快写模式
  - 利用新的快速写入模式 – 在断电之前写入少量数据，防止重要数据的丢失。
- 已经在生产中得到证实
  - 超过 150M 的用户使用 S32K1xx 系列和 Kinetis 系列 MCU EEPROM

2.2 P-Flash D-Flash和EEPROM的关系



如上图红框部分所示，FlexNVM 可以用作 EEPROM 备份 Flash（E-Flash）和 D-Flash；当 CSEc 使能时，EEPROM 必须使能，同时 EEPROM 会拿出部分空间用于存储 CSEc 的 Key。此时，EEPROM 的数据只能通过 FlexRAM 进行读写访问；CSEc 的 Key 只能通过 PRAM 进行读写访问。

2.3 S32K1xx系列MCU EEPROM 工作原理简介

S32K1xx EEPROM 由 RAM 模块（FlexRAM），E-Flash 模块和 EEPROM 状态机三部分组成。如果启用 EEPROM 功能后，FlexRAM 将成为访问 EEPROM 的内存；E-Flash 将作为真正备份和存储 EEPROM 数据的内存。针对 S32K11x 系列 MCU 的 EEPROM 的大小只有 0K 或 2K；S32K14x 系列的 MCU 的 EEPROM 的大小只有 0K 或 4K，即这个大小对应的就是 FlexRAM 的大小。FlexRAM 地址空间是用户访问所有 EEPROM 数据的地方。当访问 EEPROM 时，EEPROM 状态机会跟踪数据并将其作为数据记录备份，存储在作为 E-flash 的 FlexNVM 的某些地址中。通过使用大块 E-flash 用于少量 EEPROM 数据的备份，可使 S32K1xx EEPROM 实现具有极高的耐用性。EEPROM 状态机使用 72 位将数据从 EEPROM 备份到 E-Flash 中。其中 32 位用于存储数据，其他 40 位是关于数据的地址，状态和奇偶校验

信息。数据记录部分是需要写入和擦除的；这意味着如果 EEPROM 中的某个位置从未被访问过，则不会有数据记录。这有助于减少需要备份的数据量，从而提高 EEPROM 的使用寿命。

当 FlexNVM 配置为模拟 EEPROM 时，写入数据到 FlexRAM 时，会调用模拟的 EEPROM 文件系统来编程新的 EEPROM 数据，并以循环的方式将数据记录在模拟 EEPROM 中。根据需要，模拟的 EEPROM 文件系统可识别模拟的 EEPROM 备份扇区中正在被擦除的扇区（以备将来使用）和已被擦除的扇区。

当 EEPROM 的备份扇区被填满后，系统将之前保存最旧的数据扇区的 EEPROM 的记录数据逐渐复制到先前擦除的 EEPROM 备份扇区。当扇区复制完成时，将最旧数据的模拟 EEPROM 备份扇区标记为擦除，并保持为擦除状态。

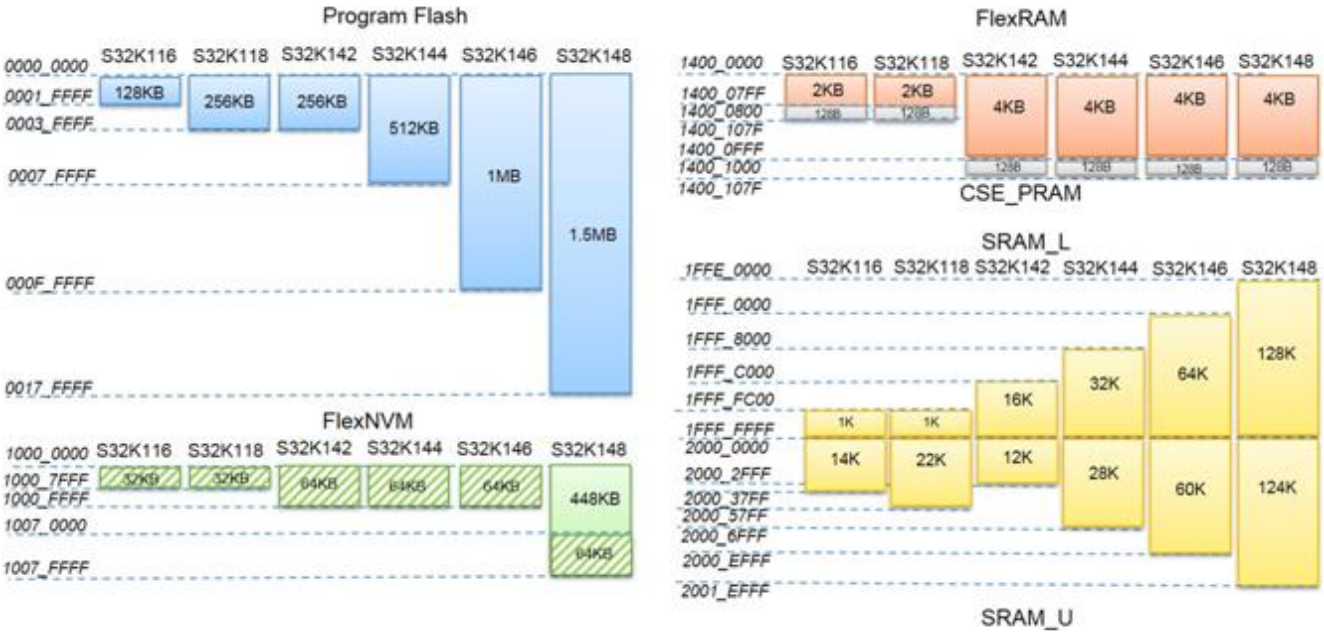
2.4 S32K1xx系列MCU EEPROM 使用案例

S32K1xx 有两个独立的闪存块，P-Flash 块和 FlexNVM 块。 P-Flash 旨在用作程序存储，但它也可用于存储数据。 FlexNVM 模块是一个可配置的闪存模块，可用作额外的闪存空间（D-Flash），也可用作支持模拟 EEPROM 功能（E-Flash）的备份存储器，或两者的组合。

NOTE

未用作EEPROM备份存储器（E-Flash）的FlexNVM部分称为D-Flash。D-Flash通常用于存储数据；但是与P-Flash一样，D-Flash实际上可用于存储指令。

下图显示了 S32K1xx 系列 MCU Memory map。 其中，S32K1xx 系列 MCU 的 P-Flash 的起始地址都是从 0x0000_0000 开始的；不同型号的 S32K1xx MCU 的 P-Flash 大小有所差别。D-Flash 的起始地址都是从 0x1000_0000 开始的。FlexRAM 的起始地址都是从 0x1400_0000 开始的。

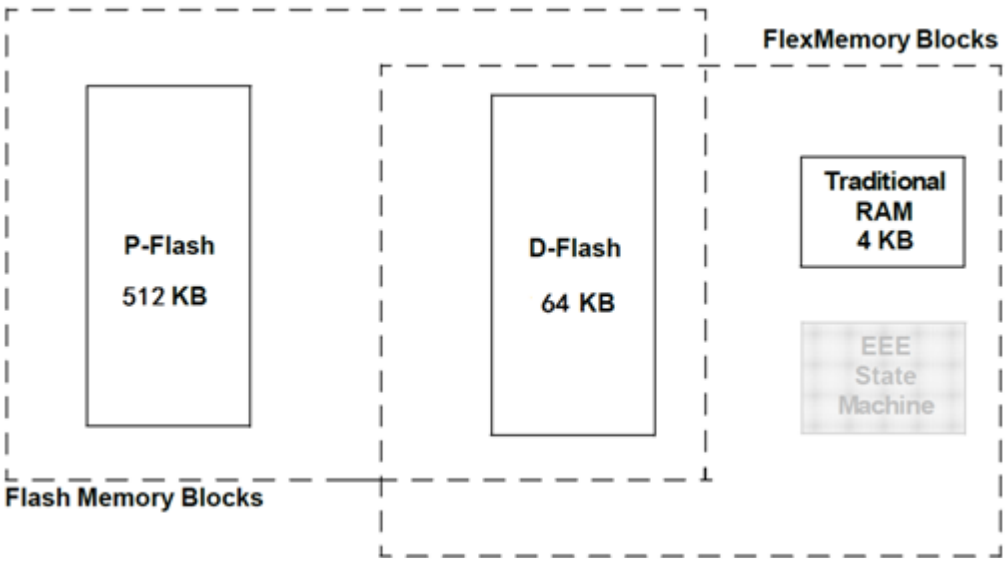


S32K148 需要特别注意:

其中 FlexNVM 模块与 448 kB 的 P-Flash 在同一个 partion。3.1.1.2 节软件配置描述了这些注意事项以及使用 S32K1xx 的 FlexNVM 映射为不同 EEPROM 配置的参考。

2.4.1 EEPROM和CSEc全部禁止

下图是S32K144的EEPROM和CSEc全部禁止即不使用EEPROM功能的内存分布框图。

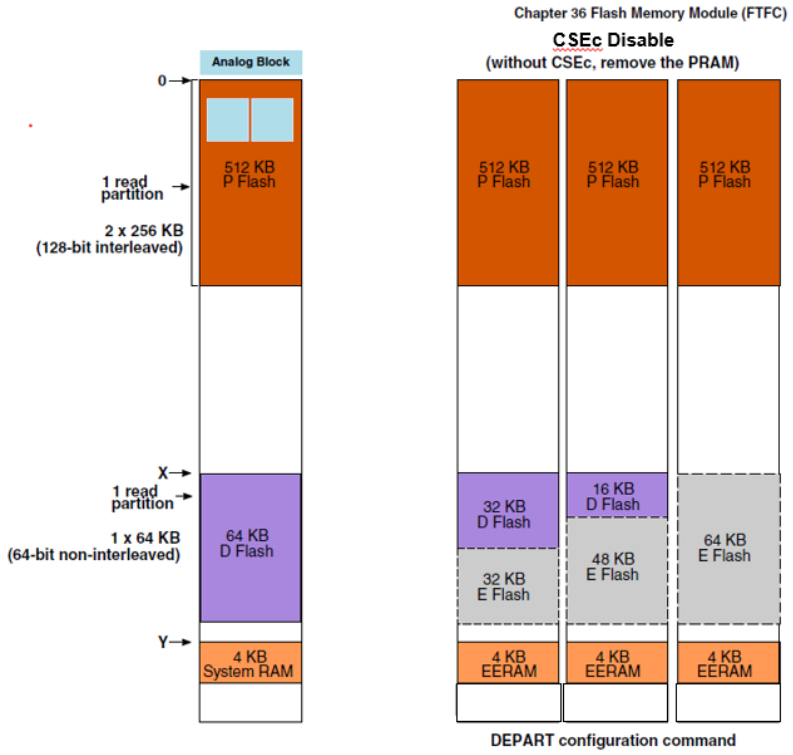


该配置的主要特点包括:

- 64kB FlexNVM 可用作 P-Flash 或 D-Flash，即 FlexNVM 可用于程序或数据存储。如果用作 P-Flash，由于 FlexNVM 不可缓存；因此，可能会出现性能下降。
- 4 kB FlexRAM 可用于传统的 SRAM，但它不像主 SRAM 那样具有 ECC，并且以 flash 的时钟速度运行。
- 由于未启用 EEPROM 机制，整个 FlexNVM 被分配为 D-Flash 空间。其中 EEPROM 状态机存在于设备中，但未激活。此时，FlexNVM 的使用特性与 datasheet 中指定的 P-Flash 存储器的特性相同。

2.4.2 只使能EEPROM

下图是只使能S32K144的EEPROM的内存分布。



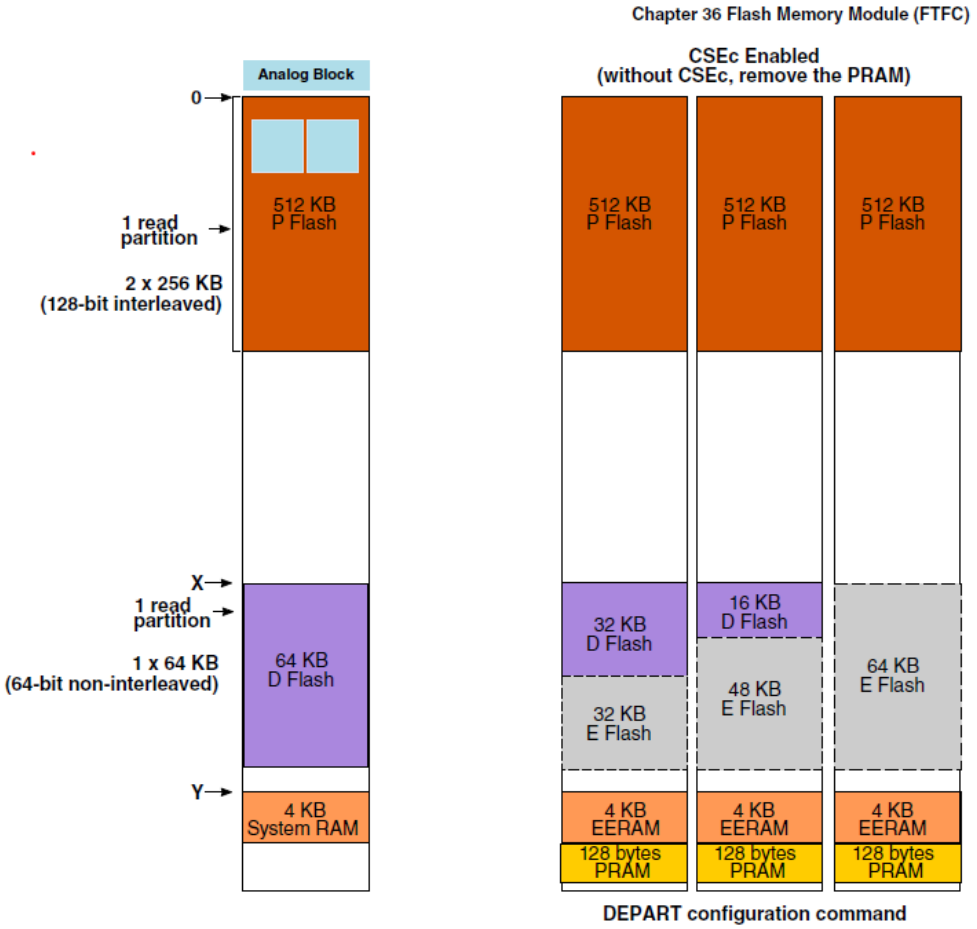
该配置的主要特点包括:

- 在此配置中，32/48/64 KB 的 FlexNVM 用作 E-Flash（EEPROM 备份），剩余的 32/16/0 KB 用作 D-Flash。
- E-Flash 未进行内存映射。即启用后无法通过任何方式直接访问此内存。

- EEPROM 数据的任何读取和写入都使用 4 kB EERAM 存储空间通过 8/16/32 位进行访问。

2.4.3 EEPROM和CSEc都使能

下图是S32K144的EEPROM和CSEc都使能时的内存分布。



该配置的主要特点包括:

- 在此配置中，32/48/64 KB 的 FlexNVM 用作 E-Flash（EEPROM 备份和密钥存储），32/16/0 KB 用作 D-Flash。
- E-Flash 未进行内存映射。即启用后无法通过任何方式直接访问此内存。
- 对 EEPROM 的数据读取和写入操作都只能使用 3.5 kB EERAM 存储空间通过 8/16/32 位进行访问。对 Key 的读取和写入操作都只能使用 128B 的 PRAM 进行访问。
- 要使用 CSEc 功能，必须启用 EEPROM，因为 CSEc 密钥保存在 E-Flash 中。

- CSEc 使能和 CSEc‘Key 大小都由 PRGPART 命令控制。根据 Key 的大小，从 EEPRAM 最后的地址分配空间(对应关系：1-5 个 Keys, RAM 128B; 1-10 个 Keys, RAM 256B; 1-20 个 Keys, RAM 512B)。即最多使用 512B 进行密钥存储，而 EEPROM 则使用剩余的 3.5KB EERAM 进行访问。执行 CSEc 操作时，此配置是必需的。

2.5 S32K1xx系列MCU EEPROM 配置注意事项

S32K1xx 允许许多不同的内存配置选项。EEPROM 数据的数量和用于备份 EEPROM 数据的 E-flash 存储器的数量都是可编程的。这允许用户在 EEPROM 的内存大小和使用寿命之间进行权衡。

有两个可编程选项用于定义系统的确切 EEPROM 内存使用情况。

1. EEPRAM 大小 - 这是存储所需 EEPROM 数据的总大小。对于 S32K14x，总 EEPROM 大小只能是 0 或 4 kB。对于 S32K11x 只能是 0 或 2 kB。
2. FlexNVM 分区 - 此参数定义 FlexNVM 用作 D-Flash 和用作 EEPROM 备份内存（E-Flash）的大小。如果使用 EEPROM，则必须将 FlexNVM 的至少 32 kB（S32K14x）或 24 kB（S32K11x）分配为 E-Flash。为了让 EEPROM 获得最大的使用寿命，那么整个 FlexNVM 都可用作 E-Flash。FlexEVM 的使用寿命受 EEPROM 备份内存大小和 EEPRAM 大小之间的关系影响。有关 FlexNVM 可靠性规范的详细信息，请参见 S32K1xx datasheet。

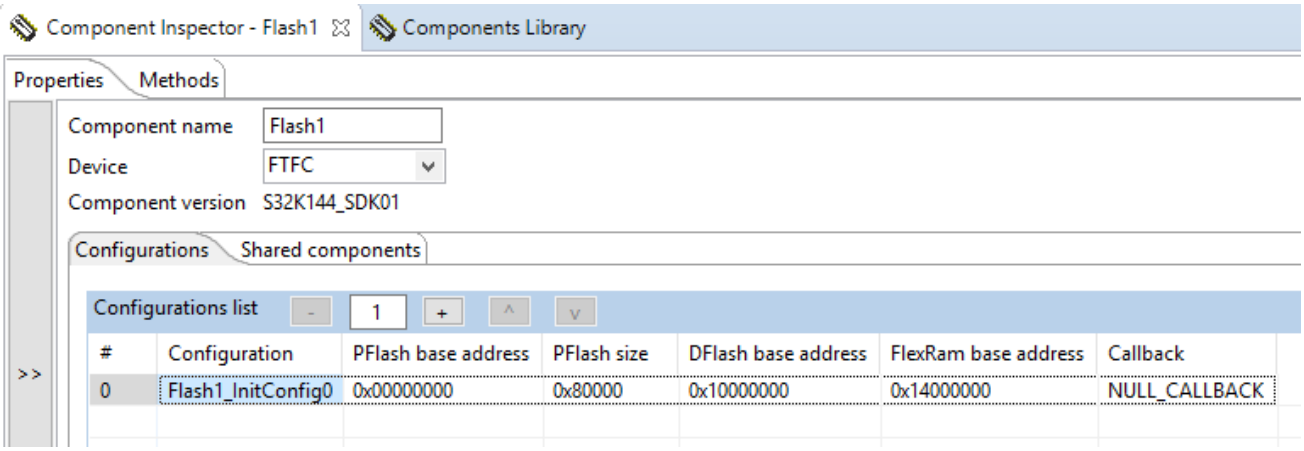
3. S32K1xx系列MCU EEPROM使用详解

3.1 S32K1xx SDK中flash驱动组件使用详解

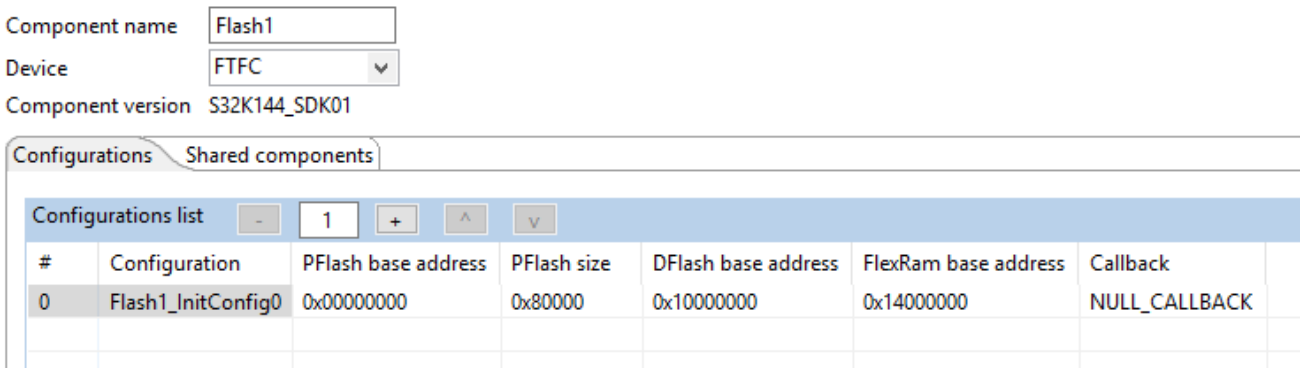
由 2.2 节可知：P-Flash，D-Flash 和 EEPROM 都属于 FTFC 模块，而 S32K1xx SDK 的 flash 组件提供 FTFC 的配置和操作 API 函数。下面将详细介绍该组件的使用。

3.1.1 S32K1xx SDK中flash组件配置

在组件视窗(Component Inspector)中 flash 组件的属性(Properites)配置界面如下：



在配置(Configurations)一栏中，用户可以配置如下参数：



Configuration list：用户可以通过“+”/“-”添加和移除一个或多个配置选项。

Configuration：定义 Flash 初始化配置选项的变量名，用户可自行更改。

PFlash base address：指 P-Flash 的起始地址；从 2.4 节可知：S32K1xx 系列 MCU 的 P-Flash 的起始地址都是从 0x0000_0000 开始的。

PFlash size：指 P-Flash 的大小；不同型号的 S32K1xx MCU 的 P-Flash 大小有所差别，用户需根据具体型号的 MCU 进行设置该配置选项。

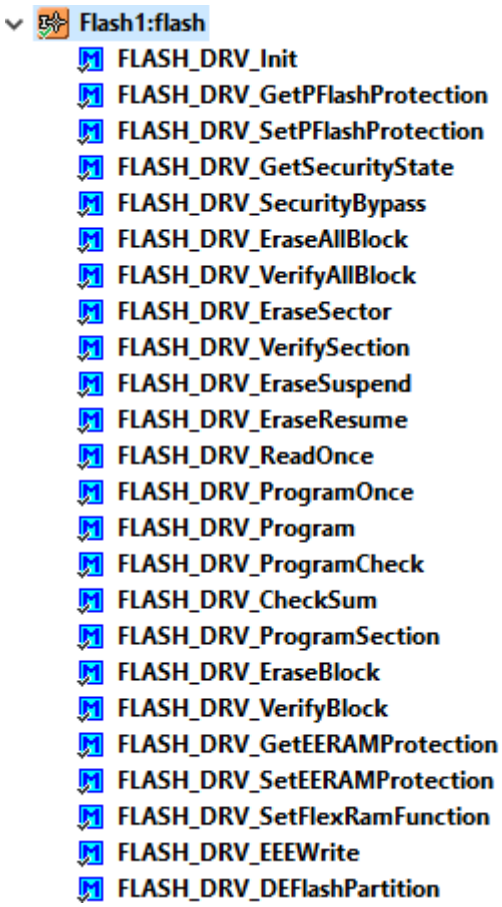
DFlash base address：指 D-Flash 的起始地址；从 2.4 可知：S32K1xx 系列 MCU 的 D-Flash 的起始地址都是从 0x1000_0000 开始的。

FlexRam base address：指 FlexRAM 的起始地址；从 2.4 可知：S32K1xx 系列 MCU 的 FlexRAM 的起始地址都是从 0x1400_0000 开始的。

Callback：指等待 Flash 操作命令完成时调用的回调函数，该函数名用户可自行定义，且需要自己实现。默认是空的。

3.1.2 S32K1xx SDK中flash组件的API说明

在 S32K1xx SDK 的 flash 组件中提供了 FTFC 模块配置和读写的若干 API 函数，其中比较重要的 API 函数功能如下：



其中与 EEPROM 操作相关的函数如下：

FLASH_DRV_Init()：Flash 模块初始化 API 函数，根据用户配置初始化 Flash 模块；

FLASH_DRV_DEFlashPartition()：对 D-Flash 进行分区，具体细节请参见 3.2 节；

FLASH_DRV_GetEERAMProtection()：获取当前 FlexRAM 的哪些 EEPROM 部分受到编程和擦除操作的保护。形参 **protectStatus** 返回 EEPROM 保护寄存器的当前值。每个位对应保护状态为 EEPROM 总使用量的 1/8。最低有效位对应于 EEPROM 的最低地址区域。最高有效位对应于 EEPROM 的最高地址区域，依此类推。 **protectStatus** 的某一 bit 为 0：此区域受到保护。**protectStatus** 的某一 bit 为 1：此区域未受保护；

FLASH_DRV_SetEERAMProtection()：设置 FlexRAM 的哪些 EEPROM 部分受到编程和擦除操作的保护。原理与 **FLASH_DRV_GetEERAMProtection()**一致；

FLASH_DRV_SetFlexRamFunction()：设置 FlexRAM 的功能，具体细节请参见 3.3 节 FlexRAM 的功能；

FLASH_DRV_EEEWrite()：通过 FlexRAM 往 EEPROM 写数据。该函数有以下限制：

- 如果第二个形参 destination 地址是 32bit 对齐，并且第三个形参大于 32bit 即 4Byte，那么以 32bit 的写入将被调用
- 如果第二个形参 destination 地址是 16bit 对齐，并且第三个形参大于 16bit 即 2Byte，那么以 16bit 的写入将被调用
- 如果第二个形参 destination 地址是 8bit 对齐，并且第三个形参大于 8bit 即 1Byte，那么以 8bit 的写入将被调用
- 如果 EEPROM 被设置成快写模式，第二个形参 destination 地址必须是 32bit 对齐，并且第三个形参必须被 4 整除

3.2 S32K1xx系列MCU EEPROM 分区详解

要使用 EEPROM 功能，必须对 FlexNVM 的内存进行分区。分区过程告诉状态机将使用多少 EEPROM 内存以及将使用多少 FlexNVM 来备份 EEPROM 即 E-Flash。FlexNVM 有一个特殊的分区命令，用于配置 EEPROM。分区命令用于对 EEPROM 的两个参数即 E-Flash 的大小和 EEPROM 的大小进行编程。这两个参数被编程到 FlexNVM 的特殊位置。由于这是一个非易失性存储位置，因此在设备的整个生命周期中只需要进行一次分区。在启动分区命令之前，FlexNVM 和 D-Flash IFR(Nonvolatile information register)必须处于擦除状态。建议在工厂编程中，将新设备的分区做为第一步。

NOTE

原则上只需要进行一次分区。如果重新对FlexNVM进行分区，则记录的数据将全部丢失，并且将无法保证EEPROM获得预期的使用寿命。

NOTE

如果设备被大规模擦除，则分区信息，EEPROM数据和EEPROM位置信息都将丢失。当使用EEPROM，同时设备上也启用安全性(security)时，强烈建议使用后门密钥(backdoor key)。该后门密钥允许临时禁用安全性而无需执行批量擦除(mass erased)的方法。原

因是：如果使用批量擦除，则不能保证EEPROM的使用寿命。在使用任何批量擦除之前需要启用CSEc去执行DBG_AUTH命令。

3.2.1 S32K1xx系列MCU EEPROM分区命令

分区命令将 FlexNVM 分割为 D-Flash，E-Flash 或两者的组合，并初始化 FlexRAM。有关详细信息，请参阅 S32K1xx's reference manual。下表显示了分区命令所需的参数。

表 1 分区命令参数

FCCOB Number	FCCOB Contents [7:0]
0	0x80 (PGMPART)
1	CSEc Key Size
2	SFE
3	FlexRAM load during reset option (only bit 0 used): 0 - FlexRAM loaded with valid EEPROM data during reset sequence 1- FlexRAM not loaded during reset sequence
4	EEPROM Data Set Size Code
5	FlexNVM Partition Code

3.2.1.1 分区命令参数

对 D-Flash 分区操作命令通过配置 FTFC 模块中的 FCCOB 寄存器完成。对于分区命令，需要六个参数：

- FCCOB0 定义了所需的命令。 0x80 表示 PGMPART 命令（分区命令）。
- FCCOB1 和 FCCOB2 用于 CSEc 配置。在 AN5401（CSEc 安全模块）中深入介绍了这两个参数。如果仅使用 EEPROM 功能，这两个值可以设置为 0x00。
- FCCOB3（仅使用 bit 0）配置 FlexRAM 是否在复位期间加载 EEPROM 数据。如果该位清零（0x00），则 FlexRAM 在复位期间自动加载 EEPROM 的数据。如果置 1（0x01），则 FlexRAM 在复位期间不加载 EEPROM 的数据，这意味着复位后它将作为传统 RAM 运行，直到调用 flash 组件的 FlexRAM_Set() API 函数配置 FlexRAM 为正常 EEPROM 或快速写 EEPROM 功能。建议：如果为了缩短启动时间，可以将其配置成 0x01；但通常情况下设置为 0x00。
- FCCOB4 表示 EEPROM 数据大小。根据 FlexRAM 大小，此选项只有两个不同的值。

表 2 EEPROM data size

EEERAMSIZE value (FCCOB4[3:0])	EEPROM data size (Bytes)
0xF	0
0x3[1]	2K
0x2[2]	4K

1．S32K11x 仅有 2KB 的有效 FlexRAM

2．S32k14x 仅有 4KB 的有效 Flex RAM

NOTE

当FlexNVM分区代码设置为无EEPROM操作时，EEPROM数据大小必须设置为0（即FlexNVM配置为仅用作D-Flash）。

- FCCOB5 指示如何在数据闪存（D-Flash）和模拟 EEPROM 备份存储器（E-Flash）之间拆分 FlexNVM 模块。此选项有不同的值，具体取决于 FlexNVM 大小。下表显示了不同的配置。

表 3 32K FlexNVM(S32K11x) 分区

FlexNVM Partition Code (FCCOB5[3:0])	Data flash Size (Kbytes)	EEPROM-backup Size (Kbytes)
0x0	32	0
0x3	0	32
0x8	0	32
0x9	8	24
0xB	32	0

表 4 64K FlexNVM(S32K142/4/6) 分区

FlexNVM Partition Code DEPART (FCCOB5[3:0])[1]	Data flash Size (KB)	EEPROM-backup Size (KB)
0000	64	0
0011	32	32
0100	0	64
1000	0	64
1010	16	48
1011	32	32
1100	64	0

表 5 64K FlexNVM(S32K148) 分区

FlexNVM Partition Code DEPART (FCCOB5[3:0])[1]	Data flash Size (KB)	EEPROM-backup Size (KB)
0000	DFlash size (512 KB)	0
0100	DFlash size - EEPROM size (512 KB - 64 KB = 448 KB)	64
1111	DFlash size (512 KB)	0

1. FCCOB5[7:4] = 0000

在启动（launch）分区命令之前，D-Flash IFR(Nonvolatile information register)必须处于擦除状态。启动分区代码后，E-Flash 的大小和 EEPROMSIZE 值保存在 D-Flash IFR 区域，并且在系统复位期间，将 D-Flash IFR 区域加载到 FTFC_FCFG1 寄存器的 DEPART 字段中。有关 D-Flash IFR 请参考 S32K1xx Reference Manual 36.4.3 节 Data flash 0 IFR map。

NOTE

SIM_FCFG1 [DEPART]字段用于保存当前的FlexNVM分区代码，因此建议在启动分区命令之前，用户检查此字段并验证FlexNVM尚未分区（DEPART字段为0xF表示尚未分区设备）以避免分区命令错误。

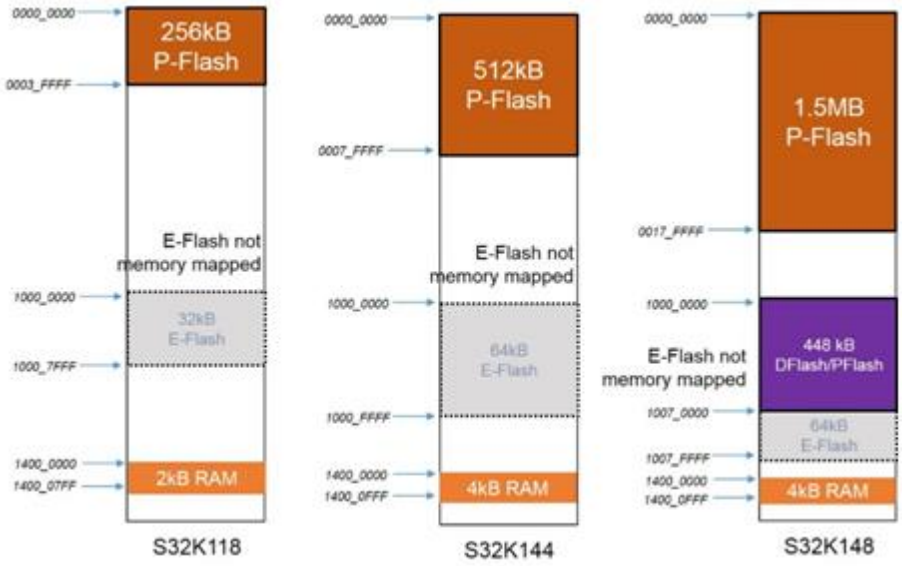
3.2.1.2 分区命令使用示例

S32K1xx 的 FlexNVM 有多种 EEPROM 大小可供选择。以下显示了 S32K1xx 设备的不同分区命令和 FlexNVM 配置。请注意，用于 E-Flash 的内存，用户不可直接访问，只能通过 FlexRAM 进行访问。

NOTE

FCCOBx的偏移量与FCCOB的索引号不同，这意味着FCCOB0不是第一个寄存器FCCOB数组。有关更多详细信息，请参阅Reference Manual中的FCCOB偏移说明。

- 如下图所示是：将所有的 FlexNVM 都配置为 E-Flash（32K E-Flash on S32K11x, 64K E-Flash on S32K14x）。



SDK 配置 & code：

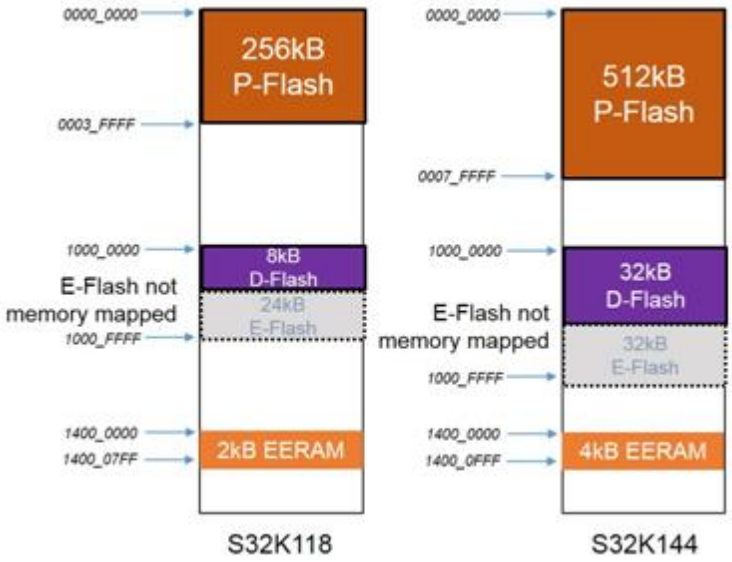
下图是以 S32K144 为例，对 Flash 的 SDK 组件配置选项的各个参数配置，其他型号 MCU 的配置可能有细微差别，具体细节请参考 2.4 节。

Configurations list - 1 + ^ v

#	Configuration	PFlash base address	PFlash size	DFlash base address	FlexRam base address	Callback
0	Flash_InitConfig0	0x00000000	0x80000	0x10000000	0x14000000	NULL_CALLBACK

```
/* Configure FlexRAM as EEPROM and FlexNVM as EEPROM backup region,
  * DEFlashPartition will be failed if the IFR region isn't blank.
  * Refer to the device document for valid EEPROM Data Size Code
  * and FlexNVM Partition Code. For example on S32K11x:
  * - EEPROMDataSizeCode = 0x03u: EEPROM size = 2 Kbytes
  * - DEPartitionCode = 0x08u: EEPROM backup size = 32 Kbytes */
#if S32K11x_DEVICES
FLASH_DRV_DEFlashPartition(&flashSSDConfig, 0x03u, 0x08u, 0x0u, false,
true);
#elif S32K14x_DEVICES
/* For example on S32K14x:
  * - EEPROMDataSizeCode = 0x02u: EEPROM size = 4 Kbytes
  * - DEPartitionCode = 0x08u: EEPROM backup size = 64 Kbytes */
FLASH_DRV_DEFlashPartition(&flashSSDConfig, 0x02u, 0x08u, 0x0u, false,
true);
#endif
```

- 如下图所示是：将 FlexNVM 分配为 D-Flash & E-Flash 组合的内存分布图（8K D-Flash & 24K E-Flash on S32K11x, 32K D-Flash & 32K E-Flash on S32K14x）；注意：在 S32K148 不能这样分配。



SDK 配置 & code：

Configurations list - 1 + ^ v

#	Configuration	PFlash base address	PFlash size	DFlash base address	FlexRam base address	Callback
0	Flash_InitConfig0	0x00000000	0x80000	0x10000000	0x14000000	NULL_CALLBACK

```
/* Configure FlexRAM as EEPROM and FlexNVM as EEPROM backup region,
  * DEFlashPartition will be failed if the IFR region isn't blank.
  * Refer to the device document for valid EEPROM Data Size Code
  * and FlexNVM Partition Code. For example on S32K11x:
  * - EEPROMDataSizeCode = 0x03u: EEPROM size = 2 Kbytes
  * - DEPartitionCode = 0x09u: EEPROM backup size = 24 Kbytes */
#if S32K11x_DEVICES
FLASH_DRV_DEFlashPartition(&flashSSDConfig, 0x03u, 0x09u, 0x0u, false,
true);
#elif S32K14x_DEVICES
/* For example on S32K14x:
  * - EEPROMDataSizeCode = 0x02u: EEPROM size = 4 Kbytes
  * - DEPartitionCode = 0x03u: EEPROM backup size = 32 Kbytes */
FLASH_DRV_DEFlashPartition(&flashSSDConfig, 0x02u, 0x03u, 0x0u, false,
true);
#endif
```

3.3 S32K1xx系列MCU FlexRAM 设置详解

Set FlexRAM 功能命令用于更改 FlexRAM 的功能：

- 当未对模拟 EEPROM 进行分区时，FlexRAM 通常用作传统 RAM。
- 当使能模拟 EEPROM 分区时，FlexRAM 通常用于存储 EEPROM 数据。

下表显示了 FlexRAM 配置命令所需的参数。有关详细信息，请参 S32K1xxReference Manual 中 FTFC 章节。

表 6 Set FlexRAM Function command

FCCOB Number	FCCOB Contents [7:0]
0	0x81 (SETRAM)
1	FlexRAM Function Control Code (see Table 36-71)
2	Reserved
3	Reserved
4	Number of FlexRAM bytes allocated for EEPROM quick writes [15:8]
5	Number of FlexRAM bytes allocated for EEPROM quick writes [7:0]
Returned Values	
5	Brown-out (BO) Detection Codes • 0x00 - No EEPROM issues detected • 0x01 - BO detected before completing EEPROM quick write maintenance • 0x02 - BO detected before completing EEPROM quick writes • 0x04 - BO detected during normal EEPROM write activity
6	Number of EEPROM quick write records requiring maintenance [15:8]
7	Number of EEPROM quick write records requiring maintenance [7:0]
8	EEPROM sector erase count [15:8]
9	EEPROM sector erase count [7:0]

3.3.1 设置FlexRAM功能的命令分类

设置 FlexRAM 功能命令用于配置 FlexRAM 功能和一些其他快速写入设置。根据控制代码值的使用情况，命令会执行不同的操作。

- FCCOB0 定义了所需的命令。 0x81 表示 SETRAM 命令（设置 FlexRAM 功能命令）。
- FCCOB1 定义了 FlexRAM 功能控制代码用于执行不同的操作。下表显示了不同的控制代码以及每个执行的操作：
- 根据选择的 FlexRAM 功能控制代码，配置剩余的 FCCOB 参数。

设置 FlexRAM 功能的命令可以随时启动(launch)。因此，用户可以在某些例程中将 FlexRAM 作为传统的 RAM，然后切换回某些 E-RAM（用作 EEPROM 接口的 FlexRAM），并且此过程不会丢失 EEPROM 的数据。

表 7 FlexRAM 功能选项

FlexRAM Function Control Code	Action
0xFF	Make FlexRAM available as RAM: • Clear the FCNFG[RAMRDY] and FCNFG[EEERDY] flags • Write a background of ones to all FlexRAM locations • Set the FCNFG[RAMRDY] flag
0xAA	Complete interrupted EEPROM quick write process • Clear the FCNFG.EEERDY flag (and FCNFG.RAMRDY flag) • Complete maintenance on interrupted EEPROM quick write operations • Set the FCNFG.EEERDY flag
0x77	EEPROM quick write status query • Clear the FCNFG.EEERDY flag (and FCNFG.RAMRDY flag) • Report emulated EEPROM status • Set the FCNFG.EERDY flag
0x55	Make FlexRAM available for EEPROM quick writes • Clear the FCNFG.EERDY flash (and FCNFG.RAMRDY flag) • Enable the emulated EEPROM system for EEPROM quick writes • Set the FCNFG.EEERDY flag
0x00	Make FlexRAM available for emulated EEPROM: • Clear the FCNFG[RAMRDY] and FCNFG[EEERDY] flags • Write a background of ones to all FlexRAM locations • Copy-down existing EEPROM data to FlexRAM • Set the FCNFG[EEERDY] flag

3.3.1.1 设置FlexRAM功能为传统RAM

如果需要将 FlexRAM 用作传统 RAM，则必须在控制代码字段中填写 0xFF，其他 FCCOB 字段不使用。

FTFC 模块将清除 FCNFG [EEPROMRDY]和 FCNFG [RAMRDY]标识，将整个 FlexRAM 的内容都填 1；并设置 FCNFG [RAMRDY]标识，指示 FlexRAM 已准备好用作传统 RAM，可以对 FlexRAM 进行正常的读写访问。请注意，FlexRAM 比 SRAM 慢，并且它们之间也不是连续地址。

NOTE

EPROT(EEPROM Region Protect)寄存器的状态不会阻止FlexRAM 被覆盖。

下面的代码段将显示：**FlexRAM_RAMInit()**是将 FlexRAM 配置为传统 RAM；**FlexRAM_RAMMain()**实现：当 FlexRAM 成功设置为 RAM 功能时，对 FlexRAM 进行写操作。

SDK code：

```
static flash_eeprom_status_t gs_EEPROMStatus;
uint32_t* const gs_FlexRAMStartAdd = (uint32_t*)0x14000000u;
void FlexRAM_RAMInit(void)
{
  gs_EEPROMStatus.brownOutCode = 0u;
  gs_EEPROMStatus.numOfRecordReqMaintain = 0u;
  gs_EEPROMStatus.sectorEraseCount = 0u;
  FLASH_DRV_SetFlexRamFunction(&flashSSDConfig, EEPROM_DISABLE, 0u,
&gs_EEPROMStatus);
/* FlexRAM is ready to be used as traditional RAM */
  while((FTFx_FCNFG & FTFx_FCNFG_RAMRDY_MASK) != 0x02u);
}

void FlexRAM_RAMMain(void)
{
  *gs_FlexRAMStartAdd = 0x100;
}
```

3.3.1.2 设置FlexRAM 功能为普通模拟EEPROM(No quick writes)

如果将 FlexRAM 用于普通的模拟 EEPROM 操作，则必须在控制代码字段中填写 0x00，其他 FCCOB 字段不使用。

FTFC 将清除 FCNFG [RAMRDY]和 FCNFG [EEPROMRDY]标识，并将整个 FlexRAM 的内容都填 1；然后自动从 EEPROM 备份记录空间复制现有的 EEPROM 数据到 FlexRAM。当完成 EEPROM 复制后，FCNFG [EEPROMRDY]标识置 1；然后可以正常读取和写入 FlexRAM。

下面代码段显示：**FlexRAM_NomEEPROMInit()**是将 FlexRAM 初始化为普通模拟 EEPROM。当初始化成功后，然后对 FlexRAM 进行写操作。

NOTE

对于启用CSEc的MCU，FlexRAM 为普通模拟EEPROM功能是默认的FlexRAM模式。因此，无需启动Set FlexRAM控制代码设置为0x00的命令。

SDK code：

```
void FlexRAM_NomEEPROMInit(void)
{
  gs_EEPROMStatus.brownOutCode = 0u;
  gs_EEPROMStatus.numOfRecordReqMaintain = 0u;
  gs_EEPROMStatus.sectorEraseCount = 0u;
  FLASH_DRV_SetFlexRamFunction(&flashSSDConfig,
                               EEPROM_ENABLE,
                               0u,
                               &gs_EEPROMStatus);
  while ((FTFC-> FCNFG & FTFC_FCNFG_EEPROMRDY_MASK) == 0);
}
void FlexRAM_NomEEPROMMain(void)
{
  uint8_t buffer[4] = {0x01u, 0x02u, 0x03u, 0x04u};
  uint8_t value = 0xFFu;
  FLASH_DRV_EEPROMWrite(&flashSSDConfig,
                        flashSSDConfig.EERAMBase,
                        4u,
                        &buffer[0]);
  FLASH_DRV_EEPROMWrite(&flashSSDConfig,
                        flashSSDConfig.EERAMBase + 1u,
                        1u,
                        &value);
}
```

3.3.1.3 设置FlexRAM 功能为快速写入(quick writes)模式的模拟EEPROM

如果需要将 FlexRAM 用于 EEPROM 快速写入模式，则必须在控制代码字段中使用 0x55；同时需要用 FCCOB4 和 FCCOB5 字段指示用于快速写入的字节量。这 2 个字段代表 16 位值（FCCOB4 [15：8]和 FCCOB5 [7：0]）。可以从 16 到 512（0x0010 到 0x0200）之间的值中选择字节数，但是选择的字节数必须可被 4 整除，否则 FlexRAM 功能命令将失效，且 FSTAT [ACCERR]标识将置 1。其他 FCCOB 寄存器不使用。

NOTE

每次写入只允许32位快速写入，其他写入方式则会产生访问错误。

当将 FlexRAM 用于 EEPROM 快速写入时，FlexRAM 的内容将保留原样，并准备模拟 EEPROM 系统以进行快速写入操作。设置 FCNFG 寄存器中的 EEPROMRDY 标识后，模拟 EEPROM 系统已准备好进行快速写入。此时，可以对 FlexRAM 进行读写访问，但每次写入 FlexRAM 的 32 位将调用 EEPROM 快速写入操作。 CCIF 和 EEPROMRDY 标识将在每次 32 位写操作时取消置位，当本次 EEPROM 快速写入操作完成后置位。在最后 32 位写入 FlexRAM（由 FCCOB4 和 FCCOB5 的内容指定）之后，模拟 EEPROM 系统同时为最后一次写入（~150μs）创建数据记录，然后再在整个快速写入的数据块上进行 EEPROM 维护操作。在所有 EEPROM 快速写入维护操作完成之前，CCIF 和 EEPROMRDY 标识将保持无效。

下面代码段显示：**FlexRAM_EEPROMQuickWriteInit()**是将 FlexRAM 初始化为快速写入模式的模拟 EEPROM，同时将快速写入的字节长度配置为 16 个字节。当初始化成功后，然后对 FlexRAM 进行 16 字节的快速写入操作。

SDK code：

```
void FlexRAM_EEPROMQuickWriteInit(void)
{
  gs_EEPROMStatus.brownOutCode = 0u;
  gs_EEPROMStatus.numOfRecordReqMaintain = 0u;
  gs_EEPROMStatus.sectorEraseCount = 0u;
  FLASH_DRV_SetFlexRamFunction(&flashSSDConfig,
                               EEPROM_QUICK_WRITE,
                               16u,
                               &gs_EEPROMStatus);
  while ((FTFC-> FCNFG & FTFC_FCNFG_EEPROMRDY_MASK) == 0);
}

void FlexRAM_EEPROMQuickWriteMain(void)
{
  uint8_t buffer[16] = {0};
  uint8_t index = 0;
  for(index = 0; index < 15; index++)
  {
      buffer[index] = index +1;
  }

  FLASH_DRV_EEPROMWrite(&flashSSDConfig,
                        flashSSDConfig.EERAMBase,
                        16u,
                        &buffer[0]);
  }
```

快速写入模式的原理请参考 3.4 节。

NOTE

在切换到另一个FlexRAM功能之前，必须完成快速写入。如果用户想要开始新的快速写入操作，而另一个快速写入操作正在进行时，则同样适用；在开始新快速写入操作开始之前必须完成当前的快速写入操作（无法中止正在进行的操作）。

3.3.1.4 设置FlexRAm功能为查询EEPROM写入状态

如果需要使用 FlexRAM 查询写入状态，则必须在控制代码字段中使用 0x77。其他 FCCOB 字段不用于启动(launch)命令，但是，其中一些字段用于从写入操作中检索状态。

命令启动(launch)后，FCNFG [EEPROMRDY]标识清零，FTFC 将询问模拟 EEPROM 系统并报告 EEPROM 清理要求（在 FCCOB6 和 FCCOB7 上），EEPROM 扇区擦除计数（在 FCCOB8 和 FCCOB9 上），以及 FCCOB5 将根据下表中显示的条件保存掉电检测代码。

表 8 掉电检测代码

Brownout code	Condition
0x00	No EEPROM issues.
0x04	If EEPROM normal write activity is interrupted by a reset or loss of power before finalizing the record.
0x02	If EEPROM quick write activity is interrupted by a reset before writing all quick write records.
0x01	If the interruption occurs after all quick writes requested have been written but before quick write maintenance has completed.

在 EEPROM 状态加载到 FCCOB 寄存器之前，CCIF 和 EERDY 标识将保持无效。

下面代码段显示： **FlexRAM_QueryWriteStatusInit()**是将 FlexRAM 初始化为查询 EEPROM 写入状态功能。然后，通过下面的三个 API 去获取 EEPROM 上一次写操作的状态。

如果在最后一次快速写入完成之前发生复位(reset)，则掉电代码将设置为 0x02，这意味着先前的写入将被忽略，并且更新将不会生效，将最后的记录保留原样。如果在所有快速写入完成之后发生复位（但是维护操作没有完成），掉电代码将设置为 0x01，这些记录可以通过使用 FlexRAM 快速写入中断的完成命令进行恢复。

如果在正常写入操作完成之前发生复位，则掉电代码将设置为 0x04，这意味着将忽略先前的写入，并且不会使更新生效。

SDK code：

```
void FlexRAM_QueryWriteStatusInit(void)
{
  gs_EEPROMStatus.brownOutCode = 0u;
  gs_EEPROMStatus.numOfRecordReqMaintain = 0u;
  gs_EEPROMStatus.sectorEraseCount = 0u;
  FLASH_DRV_SetFlexRamFunction(&flashSSDConfig, EEE_STATUS_QUERY, 0u,
&gs_EEPROMStatus);
}

uint8_t FlexRAM_GetBrownourCode(void)
{
  return gs_EEPROMStatus.brownOutCode;
}

uint16_t FlexRAM_GetWriteMaintain(void)
{
  return gs_EEPROMStatus.numOfRecordReqMaintain;
}

uint16_t FlexRAM_GetSectorEraseCount(void)
{
  return gs_EEPROMStatus.sectorEraseCount;
}
```

3.3.1.5 设置FlexRAM功能为继续完成上一次被中断的EEPROM快速写入操作

如果需要使用 FlexRAM 来完成之前被中断了的 EEPROM 写过程，则必须在控制代码字段中使用 0xAA。其他 FCCOB 字段不使用。

启动(launch)命令后，FCNFG [EEPROMRDY]标识清零，FTFC 将处理之前未完成的所有 EEPROM 快速写入操作。如果 EEPROM 快速写入操作未完成写入所有请求的记录，则将清除写入的记录，从而有效地恢复以前有效的记录。如果写入了所有 EEPROM 快速写入记录但未完成维护操作，则将在剩余的快速写入记录上完成维护操作。 CCIF 和 FCNFG [EEPROMRDY]标识将在所有 EEPROM 维护操作完成之前，保持无效状态。

下面代码片段显示：**FlexRAM_ComInterruptInit()**是将 FlexRAM 初始化为完成上一次被中断的 EEPROM 快速写入操作的功能。

SDK code：

```
void FlexRAM_ComInterruptInit(void)
{
  gs_EEPROMStatus.brownOutCode = 0u;
  gs_EEPROMStatus.numOfRecordReqMaintain = 0u;
  gs_EEPROMStatus.sectorEraseCount = 0u;
  FLASH_DRV_SetFlexRamFunction(&flashSSDConfig,
  EEE_COMPLETE_INTERRUPT_QUICK_WRITE, 0u, &gs_EEPROMStatus);
}
```

NOTE

在对FlexRAM进行操作时，有两个标识需要注意。

RAMRDY(RAM Ready)：

该标识指示FlexRAM的当前状态。RAMRDY标识的状态通常由 Set FlexRAM Function命令控制。在复位期间，如果FlexNVM模块被分区为模拟EEPROM，并且在复位期间从模拟EEPROM中加载数据到FlexRAM，则清除RAMRDY标识；如果FlexNVM模块未被分区为模拟EEPROM，可选择在复位期间不加载数据到FlexRAM，则把RAMRDY标识置1。如果运行Program Partition命令将FlexNVM块分区为模拟EEPROM，则清除RAMRDY标识。当完成擦除所有块命令或不安全的擦除所有块命令或在FTFC外部触发的擦除全部等操作执行后，RAMRDY标识置1。

0b - FlexRAM不适用于传统的RAM访问

1b - FlexRAM仅作为传统RAM；写入FlexRAM不会触发EEPROM操作。

EEERDY:

该标识指示EEPROM备份数据是否已复制到FlexRAM，因此可用于读取访问判断。在复位序列期间，EEERDY标识在CCIF = 0时保持清零，仅在FlexNVM模块被分区用于模拟EEPROM时置1。

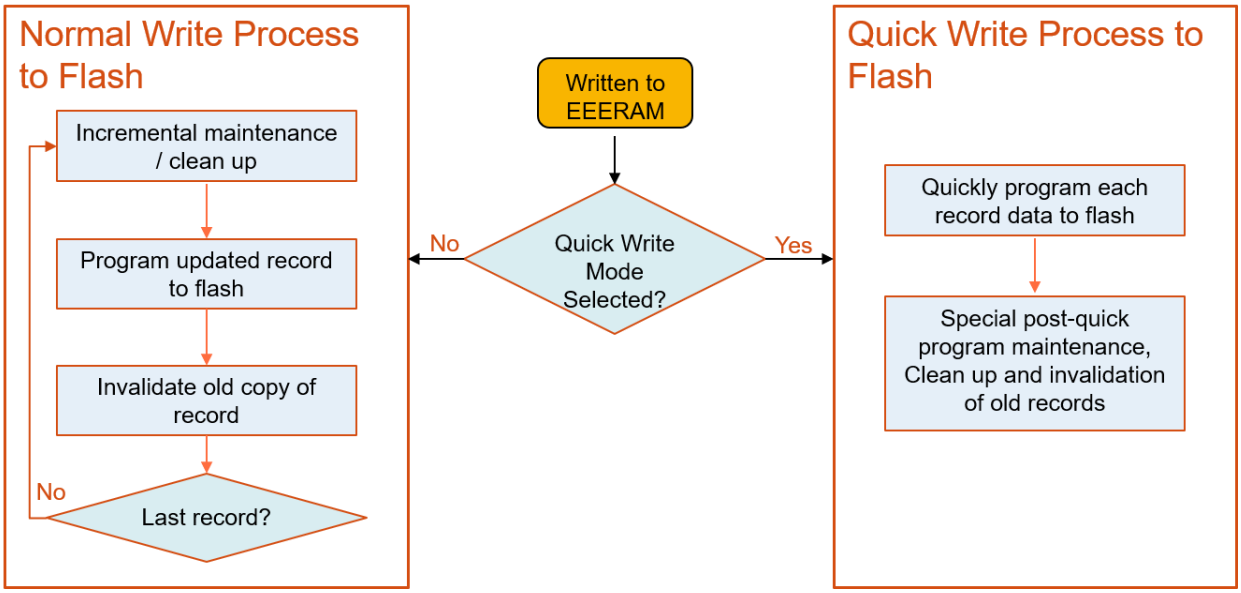
0b - FlexRAM不适用于模拟EEPROM操作

1b - FlexRAM可用于EEPROM操作，其中：（1）从FlexRAM读取先前在模拟EEPROM模式下写入FlexRAM的数据。（2）写入启动EEPROM操作以将写入的数据存储在FlexRAM和EEPROM备份中。

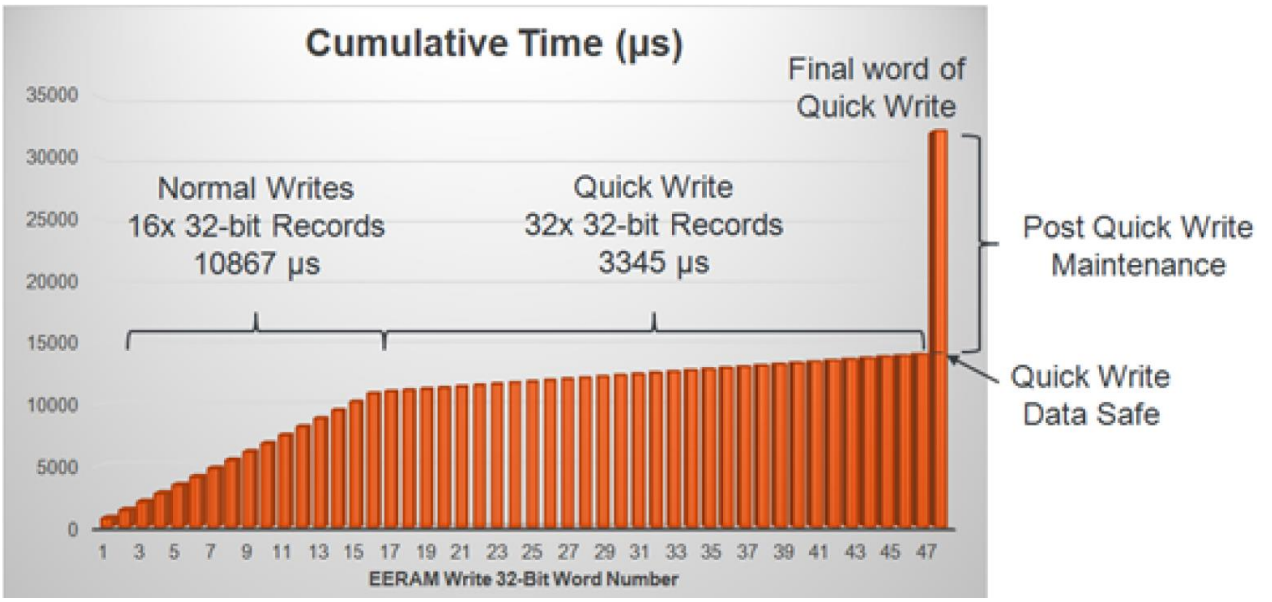
3.4 S32K1xx系列MCU EEPROM 快速写入模式原理介绍

快速写入模式以写入数据优先，例如在即将掉电之前写入数据。如有必要，可在复位后再执行 EEPROM 记录维护操作。在快速写入的维护操作完成之前，无法进行正常写入操作。所有请求的字节必须完成编程操作才能使它们有效。如果在写入最后一个字节之前写入操作因复位或掉电而中断，则所有写入均无效且最近的一次完整写入操作的记录数据有效。

下图显示了 E-Flash 的正常写入和快速写入过程的流程对比。正常写入过程在每次写入记录数据时执行维护操作。快速写入模式尽可能快地写入记录数据，将维护推迟到后面。如果在维护操作期间掉电，则此次快速写入模式可实现记录数据完全恢复。



与正常写入模式相比，快速写入模式时间提高了 66%（不加上执行维护操作时间）。下图显示了快速写入模式与正常写入模式的写入时间对比。



注：EEPROM 记录是在现有数据（不是预先擦除的数组）上编写的。

NOTE

这些数字仅供参考，不能作为绝对数字。

快速写入过程花费较少的时间将所有记录写入 E-Flash，但总时间（考虑快速写入后维护操作的时间）可能比正常写入时间长。快速写入的优点是：一旦写入最后的快速写入数据（EEPROM 到达上图中的快速写入数据安全点）任何掉电事件都不会导致数据丢失，因为后面可以进行维护。但是，如果在写入最后一个数据之前发生掉电，则不会保存新数据。

如果在快速写入维护操作完成之前发生掉电情况，则可以通过发出 Set FlexRAM 命令完成之前被中断的快速写入过程，在下一个复位期间恢复维护操作。

有时候快速写入看起来比正常写入慢，这取决于 EEPROM 系统中发生的后台清理过程。由于记录状态更新和压缩过程在两种模式下都不同，因此快速写入可能需要比正常写入更长的时间。但是，快速写入比正常写入的优势在于，编程新数据的速度更快，因为它首先将数据保存到之前被清理过的 E-Flash，所以当发生由于掉电导致维护操作未完成时，用户可以在下次上电复位时恢复剩余的维护操作，从而恢复出掉电前快速写入的数据。

3.5 S32K1xx系列MCU EEPROM 命令错误处理

对于与 EEPROM 功能（分区命令和设置 FlexRAM 功能）相关的每个 flash 命令，都存在导致命令失败的错误条件。具体请参考 S32K1xx Reference Manual 中的分区命令错误处理和

设置 FlexRAM 功能命令错误处理表，显示了这些条件以及每个条件的受影响标识。为确保避免这些错误情况，建议在每次启动这些命令时检查错误标记。

NOTE

FSTAT寄存器的访问错误（ACCER）和flash保护违规（FPVIOL）都会导致这些标识被置1；一旦这些标识被置1，将会阻止启动(launch)任何更多命令或写入FlexRAM操作，直到这些标识被清0（通过向其写入1）。

3.6 S32K1xx系列MCU EEPROM ECC 错误处理

ECC 错误由 EEE 状态机自动处理。单比特错误会被自动修正。在出现双比特错误期间，不会导致总线故障和硬件故障。在复制期间，如果双比特错误在有效记录中，则跳过该记录。如果此有效记录是一个 EEERAM 位置的唯一有效记录，则读取的位置返回 1。因此，建议不用 0xFFFF_FFFF 初始化 EEPROM 数据，从而可以识别记录是 ECC 错误还是初始化数据。

3.7 S32K1xx系列MCU EEPROM 复位启动过程

复位后，将自动加载在分区过程中写入的 EEPROM 配置选项。如果启用了 EEPROM，则状态机在系统引导(boot)期间从 E-Flash 加载 EEPROM 数据到 FlexRAM。将数据从 E-flash 复制到 FlexRAM 所需的时间可能会有所不同，具体取决于配置的 EEPROM 大小和需要解析的备份 E-flash 的数量；请参考 7.5 节。 FTFC_FCNFG [EEPROMRDY]标识会被清除，直到 EEPROM 数据的加载完成将被置位；因此在尝试访问 FlexRAM 中的 EEPROM 数据之前，软件必须等待 EEPROMRDY 标识置位。如果需要中断驱动选项而不是软件轮询，可以使用 CCIF 中断而不是轮询 EEPROMRDY。

如果复位期间 FlexRAM 加载选项字段设置为 0x01，则 EEPROM 数据将不会自动复制到 FlexRAM，FlexRAM 在 MCU 复位后用作传统 RAM。当需要时，用户必须发出 Set FlexRAM 命令将 FlexRAM 功能更改为 EEPROMRAM（Normal 或 Quick Writes 模式）。当 CSEc 功能启用时，用户调用程序分区命令时，必须将 FlexRAM 自动加载选项字段设置为 0x00，否则将导致分区失败。

关于 S32K1xx 系列 MCU EEPROM 在复位启动过程时，从 E-Flash 加载带有 EEPROM 数据到 FlexRAM 的验证请参考第 6 章提供的 Demo 工程 3。

在复位序列之后轮询 FTFC_FCNFG [EEPROMRDY]和/或 FTFC_FCNFG [RAMRDY]是获得当前 FlexRAM 模式的一个好习惯。

3.8 S32K1xx系列的MCU EEPROM读和写操作限制

通过访问 FlexRAM 地址空间来读取和写入 EEPROM 数据。 EEPROM 空间从 FlexRAM 的开头分配。可寻址空间是 FlexRAM 基址（0x1400_0000），直到编程的 EEPROM 大小。

由于 EEPROM 数据是通过 FlexRAM 访问的，因此数据可以任意大小，字节，字或长字读写。尽管可以使用任何大小访问，但用于备份 EEPROM 数据的记录使用字大小（4 个字节）的数据字段。这意味着能够以字节写入，但它的使用会降低效率。所以建议用户在写入数据时，调用 SDK 提供的默认 API **FLASH_DRV_EEEWrite()** ，且每次以 4 个字节对齐写入。

NOTE

在HSRUN模式（112 MHz）和VLPR（4MHz）模式下，CSEc（安全）或EEPROM写入/擦除将触发错误标识。所以MCU需要切换到RUN模式（80 Mhz）才能执行CSEc（安全）或EEPROM写/擦除。

3.8.1 EEPROM 写操作限制

将数据写入 EEPROM 空间时，会用到启动（launch）EEPROM 操作，将数据存储到 E-flash 存储器中。由于这是一个 flash 程序操作，软件必须检测 CCIF 位以确定在写入 EEPROM 之前是否还有其他 flash 操作正在进行中。由于不允许在同一个 flash 块中进行多个并发写入和读写操作，因此在 EEPROM 写入完成之前不允许访问 EEPROM 或 D-flash 空间。检查 CCIF 标识以确定以前的 Flash 或 EEPROM 命令是否完成。

NOTE

P-flash存储器是一个完全独立的逻辑块，因此在EEPROM写入正在进行时，可以继续对P-flash进行正常读取访问。但这种方式不适用于S32K148器件中与E-Flash位于同一存储器模块中的最后448 kB P-Flash。

EEPROM 写入不一定是顺序位置，实际上用户可以写入任何有效位置或多次写入同一地址。

NOTE

除非设备处于RUN模式，否则无法执行EEPROM写入。对于HSRUN / VLPR，仅允许EEPROM读取。

3.8.2 EEPROM 读操作限制

读取 EEPROM 时，数据由 FlexRAM 提供，因此不会触发 flash 操作。但是，在 EEPROM 写入过程中不允许进行 EEPROM 的读取。在允许软件继续执行之前，软件必须在写操作后等待 CCIF 置位。这样，EEPROM 写入时软件需要特殊处理，但 EEPROM 读取不需要任何特殊处理。这种方法的另一个优点是：如果用户有多个 EEPROM 读取访问且两者之间没有 EEPROM 写入操作，则不需要额外的延迟或标识检查。但是 EEPROM 读取的特殊情况必须考虑，即复位后首次访问 EEPROM。对于复位后第一次读取 EEPROM，必须对 EEPROMRDY 位进行检测，以确保状态机已完成从 E-flash 到 FlexRAM 的初始数据加载。如果系统启动时间很长，这可以保证在第一次读取 EEPROM 之前，初始数据加载有时间完成；然后可能不需要在第一次读取之前检测 EEPROMRDY 标识。但是，建议客户在第一次读取 EEPROM 访问之前检测 EEPROMRDY 位更为安全。

NOTE

某些CSEc引导选项（例如，MAC引导选项）可能会阻止设置FCNFG [EEPROMRDY]标识，直到引导过程正确完成才不会阻止该标识的设置。

4. S32K1xx系列MCU EEPROM 性能

S32K1xx 的模拟 EEPROM 除了提供灵活性和耐久性之外，S32K1xx 的 EEPROM 还比典型的 EEPROM 快。

4.1 S32K1xx EEPROM 快速性

传统的外部 EEPROM 通常需要大约 5 ms（最长的编程时间）。相比之下，S32K1xx 的 EEPROM 可以在 1.5 ms 最坏情况下擦除和写入。也可以通过将 0xFF 写入 EEPROM 数据位置来预擦除 EEPROM。预擦除位置有助于缩短编程时间，因为它可确保不需要擦除周期。预擦除数据位置的典型编程时间约为 100μs。此功能允许在时间紧迫的情况下快速记录数据。一个典型的用例：当检测到即将进入低功耗时，需要存储故障数据或操作信息的数据。

此外，在检测到掉电的情况下，使用快速写入模式进行数据记录可确保关键数据可以在较短的时间内写入 E-Flash，而不是正常的写入模式。

4.2 S32K1xx EEPROM 使用寿命

由于 FlexNVM 分区代码的选择会影响设备的耐用性和数据保留特性，因此根据应用要求选择最佳的分区代码/ EEERAM 大小至关重要。

如下图所示是 S32K1xx 的耐久性能，更多细节请参考 S32K1xx datasheet 和 AN11983.

Symbol	Description	Min.	Typ.	Max.	Unit	Notes
When using as Program and Data Flash						
$t_{nvmretp1k}$	Data retention after up to 1 K cycles	20	—	—	years	1
$n_{nvmcycp}$	Cycling endurance	1 K	—	—	cycles	2, 3
When using FlexMemory feature : FlexRAM as Emulated EEPROM						
$t_{nvmretee}$	Data retention	5	—	—	years	4
$n_{nvmwree16}$	Write endurance • EEPROM backup to FlexRAM ratio = 16	100 K	—	—	writes	5, 6, 7
$n_{nvmwree256}$	• EEPROM backup to FlexRAM ratio = 256	1.6 M	—	—	writes	

为了实现 S32K1xx datasheet 中所示的指定 w / e 使用寿命周期，E-Flash 大小必须至少是 FlexRAM（EEERAM）大小的 16 倍。如果用户想要分配少于 64kB 的 E-Flash，例如 S32K14x 中的 48kB 器件，EEERAM 尺寸必须小于 4kB（3kB 即分配 4KB 大小的 EEERAM，但是实际使用的 EEERAM 只有 3KB）才能达到 datasheet 中所示的使用寿命。 （48kB / 3KB = 16）。

如果需要更长的使用寿命，使用更少的记录将增加 RAM 与 NVM 的比率，从而增加使用寿命。

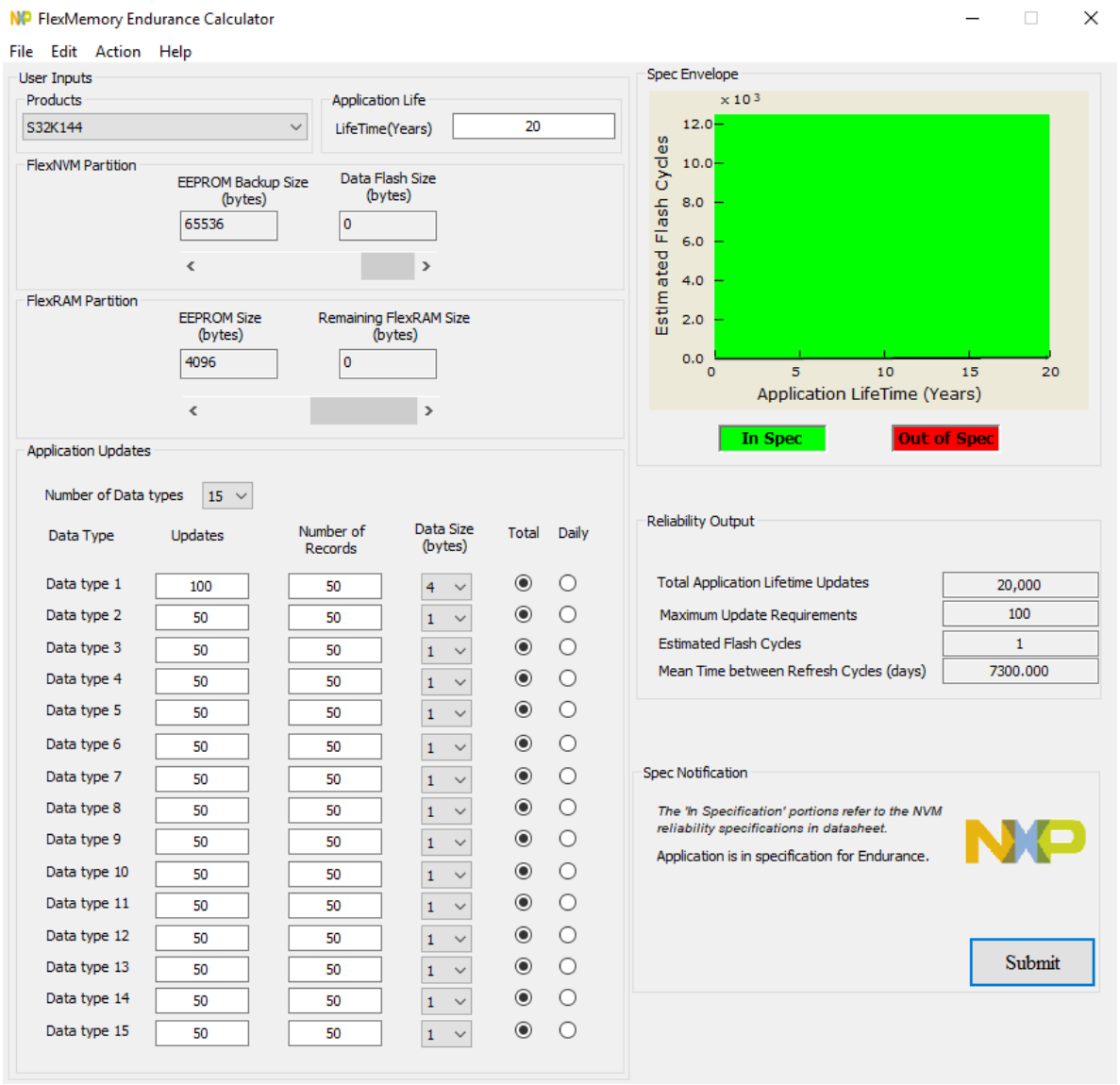
NOTE

将RAM与NVM的比率提高到16以上的方法是：在应用程序中使用较少的RAM地址。如果用户对于S32K14x器件，在总共4 kB的FlexRAM中真正只写入2 kB的数据，那么RAM和NVM的比率将从1:16变为1:32，使使用寿命加倍。

恩智浦提供了一种工具，用于估算 FlexNVM 上执行的写/擦除周期数，每个刷新 FlexNVM 周期之间的平均时间，以及用户应用所需的 EEPROM 周期数最差情况。您可以从恩智浦网站（FlexMemory Endurance Calculator）下载该工具，以根据用户自己的应用要求计算这些值。

4.2.1 FlexMemory Endurance Calculator介绍

如下图是：FlexMemory Endurance Calculator 的整个界面。关于该工具的使用请参考：菜单栏 Help 中的 User Guide。



- **User Inputs**

由上图可知，User Inputs 主要是选择 MCU 型号；Flex NVM 使用寿命；Flex NVM 分区大小；Flex RAM 分区大小。用户可根据自己需求进行设置。

- **Application Updates**

对于该系统，用户可以为该系统最多指定 15 类数据，从 Data Type 1 到 Data Type 15。每种数据类型的 Data Size 大小只有 3 种（1B, 2B 和 4 B）。

- Number of Data types：指定数据类型的数量。
- Updates：有效值是 1 到 4294967295 之间的整数。如果选择“Total”，则它反映在应用程序的整个生命周期内执行的更新总数。否则，如果选择“Daily”，它反映每日更新的频率。
- Number of records：有效值是 1 到 4294967295 之间的整数。它提供每种数据类型在 E-Flash 中的记录数量。
- Data Size：指定每种数据类型的数据大小（以字节为单位）。只有三种数据大小：1 字节，2 字节或 4 字节。

- **Spec Envelope**

Application Lifetime 与所需的 FlexNVM 写/擦除周期的图形。绿框区域表示应用程序在规范范围内，红色区域表示应用程序超出规范。它是由：

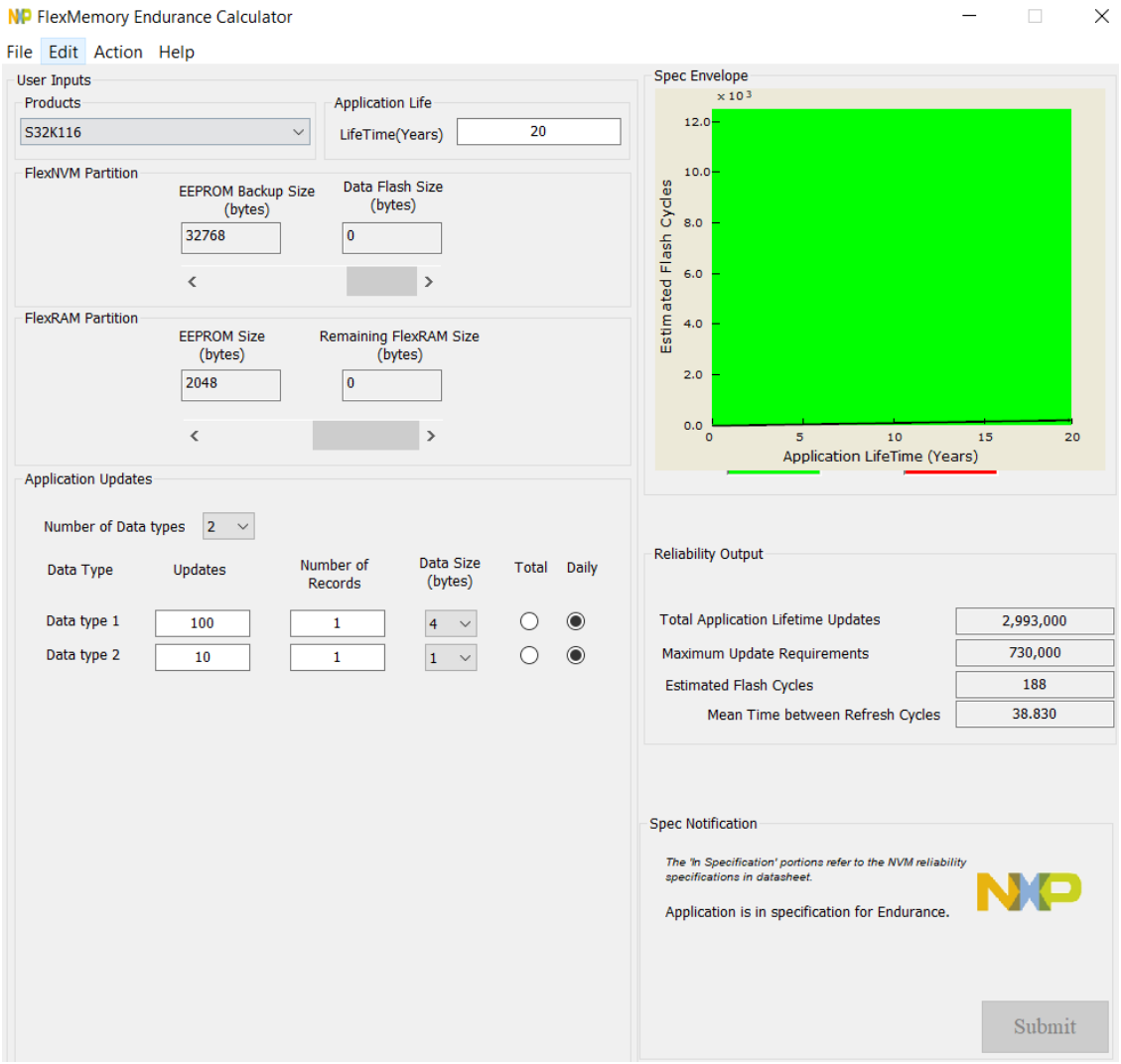
- 应用程序生命周期或超过 20 年。
- 估计闪存周期或超过 EEPROM 备份的循环耐久性。

- **Reliability Outputs**
  - Total Application Lifetime Updates：反映了整个应用程序生命周期内所有字节的总更新次数。
  - Maximum Update Requirement：是整个应用程序生命周期内所有数据类型的最大更新次数。
  - Estimated Flash Cycles：指定在 FlexNVM 上实际执行的“写入/擦除”次数。
  - Mean Time between Refresh（每天刷新的平均时间）：显示每个写入/擦除周期之间的平均时间。例如，在 10 年的时间内需要 3,650 次刷新的应用，则平均每天将有一个刷新。

4.2.2 FlexMemory Endurance Calculator使用示例

下面将以汽车仪表中的总里程，平均油耗为例说明该工具的使用。首先假设总里程为 Data type1，字节大小为 4 字节；平均油耗为 Data type2，字节大小为 2 字节。两中数据类型的记录数量都为 1。总里程以汽车行驶 1 公里记录一次；平均油耗以汽车行驶 10 公里记录一次。假设家用轿车平均每天行驶 100 公里，那么总里程每天更新 100 次，平均油耗每天更新 10 次。

从下图可以看出：总里程和平均油耗在 20 年中，更新总次数为 2993000 次，在 FlexNVM 上实际执行的“写入/擦除”次数为 188 次；下图的曲线在 20 年内并没有超出 FlexNVM 限制的擦写次数。



5. S32K1xx EEPROM 掉电检测及数据恢复

EEPROM 状态机包括：检测是否有任何 EEPROM 数据未完全编程的逻辑。此功能在文档中称为掉电检测，但名称并不完全准确。在写入过程中由于掉电或任何复位而检测到 EEPROM 数据未完全编程的任何情况都被视为相同。如果在 EEPROM 写入过程中发生复位，则数据可能会损坏。 EEPROM 状态机测试易受损坏的 EEPROM 数据记录，以查找可能未完全编程的值。如果检测到不完整的记录，则状态机将这部分记录数据标记为损坏，并在下一次 EEPROM 写入期间，将其替换为相关联的 EEPROM 地址的先前有效数据。这可确保如果 EEPROM 的写入因任何原因而中断，用户还可获得最后一次正确写入 EEPROM 的值。写入过程根据复位发生的时间，此值可以是先前值或新值，但不会获得损坏的值。

如果在复位期间，MCU 已经被分区，并且有有效的 EEPROM 数据用于加载到 FlexRAM，则复位期间将使用先前写入的有效数据替换受损数据。通过 AN11983 的第 3.2.1.4 节，可知：可以通过查询命令返回欠压代码来确定 EEPROM 的写入状态。

其中数据恢复过程取决于写入的模式：

正常写入模式：

正常写入模式为每个被写的记录区域执行维护/清理操作。因此，如果在正常写入操作期间发生复位或掉电，则记录区域将标记为无效，并且不会进行更新。

快速写入模式：

快速写入模式尽可能快地写入记录，然后将维护等操作推迟到以后。如果在写入最后一个字节之前被中断，则所有写入都不起作用且不会进行更新。但是如果在维护期间快速写入操作被中断（即所有写入已完成），则掉电代码将设置为 0x01，并且可以稍后完成维护（通过使用 FlexRAM 命令完成中断的快速写入过程）。

以下代码片段展示了恢复数据的过程：首先读取上一次 EEPROM 写入操作的掉电代码，如果掉电代码为 0x01 时，表示之前写入的数据可以被恢复。然后，将 FlexRAM 设置为完成被中断的快速写入操作的功能，即可完成数据的恢复。

```
BrownourCode = FlexRAMTest_GetBrownourCode();
if(0x1u == BrownourCode)
{
       BrownourCode = 0u;
  FlexRAMTest_ComInterruptInit();
  while (((FTFC-> FSTAT & FTFC_FSTAT_CCIF_MASK) == 0) && \

      ((FTFC-> FCNFG & FTFC_FCNFG_EEERDY_MASK) == 0));
}
```

用户可以查看设置 FlexRAm 功能为查询 EEPROM 写入状态。了解有关掉电检测的更多信息。为了避免掉电时 EEPROM 备份数据丢失，必须遵循硬件设计指南中所示的硬件电路设计建议，以确保在完成任何 EEPROM 备份序列时系统不掉电。如果使用更大的电源储能电容（Bulk Capacitor）就会面临更少的电压不足的情况。

6. S32K1xx系列MCU EEPROM demo工程

本节重点介绍 EEEPROM 可能涉及的不同场景以及处理这些场景的方法。本节提供了有关不同情况的简要指南，例如 EEEPROM 初始化和掉电处理。

6.1 S32K1xx EEPROM 分区推荐流程及Demo工程

EEEPROM 分区命令在产品生命周期中使用一次，下图显示了 FlexNVM 分区的推荐流程图。它显示了在 MCU 上电复位后，首先检测分区信息，确保正确的分区操作。然后如何发出 SetFlexRAM 命令将 FlexRAM 配置为多种模式，例如正常/快速写入模式，查询快速写入状态或完成被中断的快速写入。

MCU 每次复位时，首先读取分区信息。分区信息通常有流程图中所描述的三种情况，根据不同的情况，然后执行下一步操作。具体代码请参考 **S32K1xx_SDK_RTM3_0_FlexNVMPartition** 示例工程。

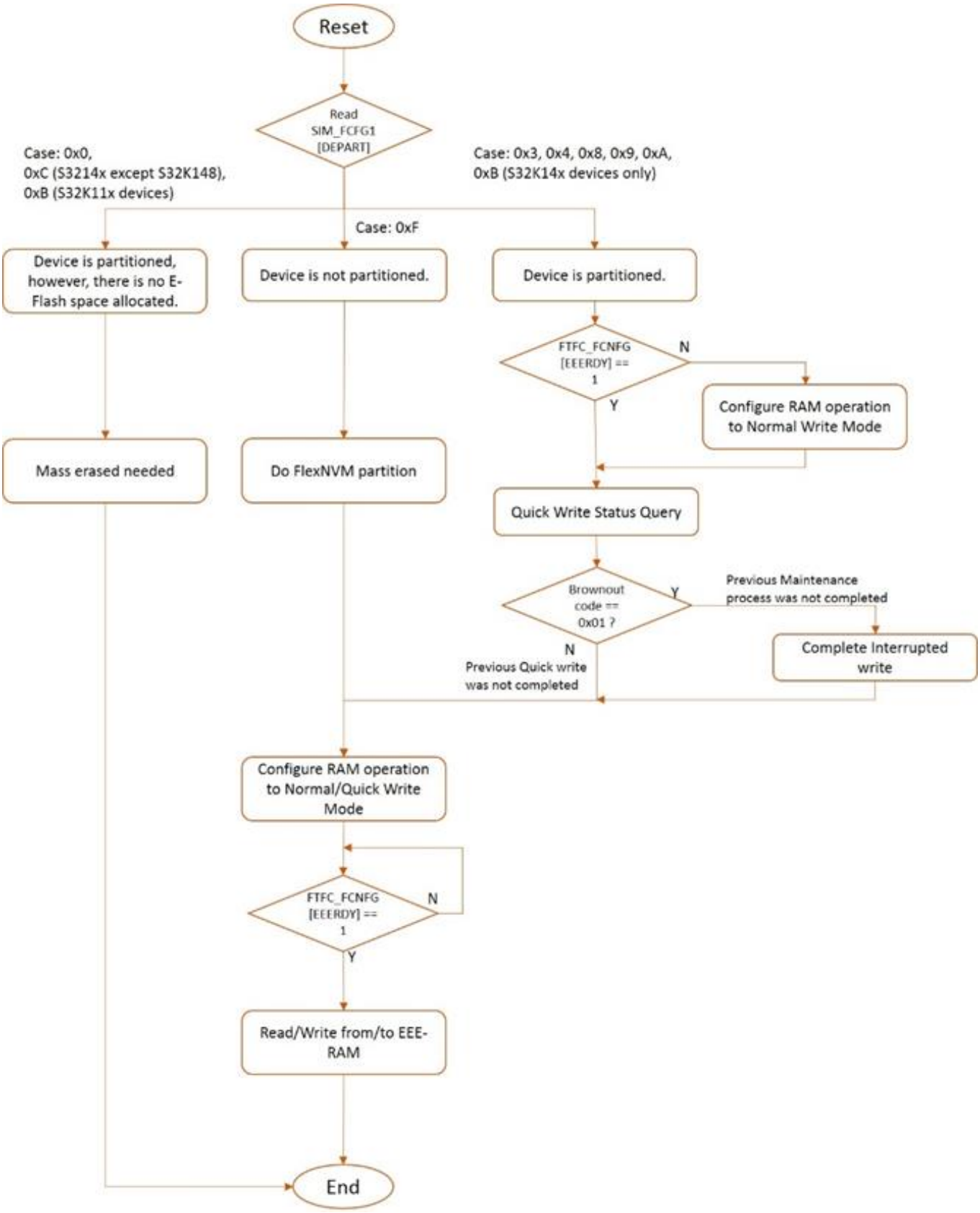
6.2 S32K1xx EEPROM防数据丢失推荐流程及Demo工程

为防止数据在写入 EEEPROM 时发生掉电，而导致数据存储失败。S32K1xx 系列 MCU 专门为 EEPROM 提供了快速写入模式：由于快速写入(Quick write)模式旨在用于在掉电事件之前保存关键状态的记录，用户可以根据系统需要，在正常写入和快速写入之间进行切换。为了保证所有记录都在掉电情况发生之前写入 FlexRAM，强烈建议在系统检测到低电压事件(比如 MCU 的 LVW 中断)时，配置使能并使用 EEEPROM 的快速写模式。

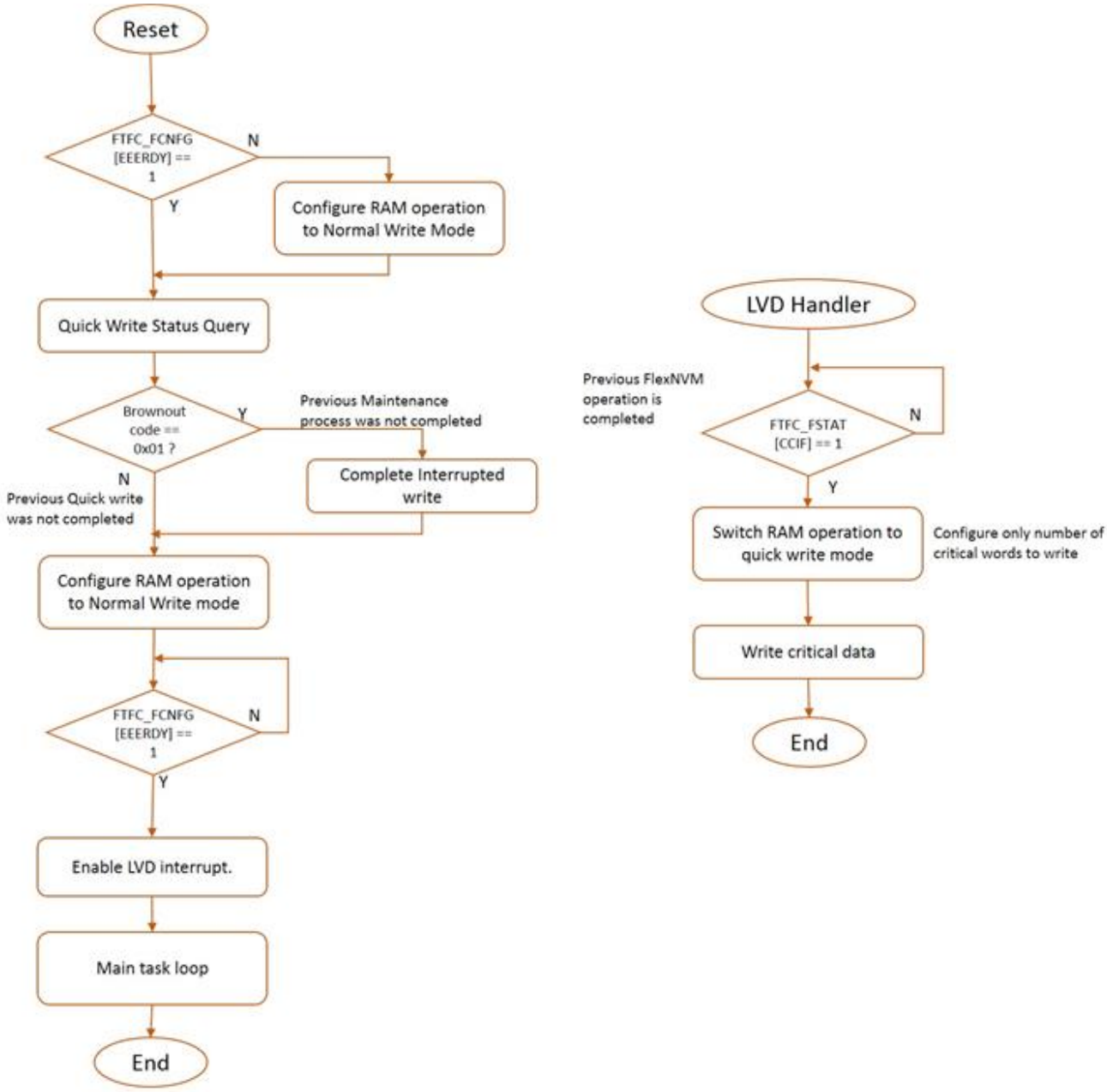
以下流程图显示了使用低压检测（LVD）中断程序处理快速写入过程的方法。通过低电压检测中断处理，能够有效的保证数据在 MCU 掉电之前安全的写入 EEPROM 中。在低电压检测中断处理函数中，首先调用 Set FlexRAM 命令将 FlexRAM 设置成 Quick Write 模式，同时设置好需要存储的字节数量；然后将需要存储的数据写入 EEPROM 中。最后退出低电压检测中断处理函数。当 MCU 复位时，首先将 Flex RAM 设置为检测快写状态模式，检测上一次写入 EEPROM 的状态，如果状态码为 0x01 表示上次写入 EEPROM 被中断的数据能够恢复。通过将 Flex RAM 的设置为完成中断的功能，即可恢复上次写入 EEPROM 且被中断的数据。

系统硬件设计必须保证 MCU VDD 的供电，直到写完最后一个字。所需时间可以从 datasheet 中获得，它依赖于写入字节数和每次 32 位写入所需的时间。

代码请参考 **S32K1xx_SDK_RTM3_0_FlexNVMQuickWrite** 示例工程。其中，该 Demo 工程是以 S32K1xx 系列 MCU 自动的 LVD 检测模块实现，而市面上实际的汽车电子产品中自带的 SBC 带有 LVD 检测功能。该 Demo 工程不能直接使用，需用户更具自己产品的特性进行修改。



S32K1xx EEPROM 分区推荐流程



S32K1xx EEPROM 防数据丢失推荐流程

6.3 S32K1xx EEPROM启动加载示例

由于 S32K1xx EEPROM 的配置不同，MCU 上电复位后，对 EEPROM 的加载有所不同，**S32k144_SDK_RTM3_0_EEPROMStartup** 示例工程给出了所有对 EEPROM 启动加载的验证程序。

TaskPartition_FlexRAMNoLoadMain()：验证的是将 D-Flash 分区为 16K 的 D-Flash 和 48K 的 E-Flash，同时禁止 CSEc，MCU 复位时不自动加载 E-Flash 的数据到 FlexRAM 中。其

中，在 **FlexRAM_NomEEPROMMain2()**函数中除了对 EEPROM 进行了写操作，同时也对剩余的 D-Flash 进行了写操作。用户直接在 main()调用即可，然后通过提示信息进行操作。通过验证结果可以得出：MCU 上电复位后，FlexRAM 的值是随机的；然后，通过将 FlexRAM 设置成 **Normal Write** 或者 **Quick Write** 后，E-Flash 的数据将自动加载到 FlexRAM 中。

TaskPartition_FlexRAMLoadMain()：验证的是将 D-Flash 分区为 0K 的 D-Flash 和 64K 的 E-Flash，同时禁止 CSEc，MCU 复位时自动加载 E-Flash 的数据到 FlexRAM 中。用户直接在 main()调用即可，然后通过提示信息进行操作。验证结果可以得出：MCU 上电复位后，E-Flash 的数据自动加载到 FlexRAM 中。

TaskPartition_FRAMLoadCSEcENMain()：验证的是将 D-Flash 分区为 0K 的 D-Flash 和 64K 的 E-Flash，同时使能 CSEc，MCU 复位时不自动加载 E-Flash 的数据到 FlexRAM 中。用户直接在 main()调用即可，然后通过提示信息进行操作；验证结果可以得出：MCU 上电复位后，E-Flash 的数据自动加载到 FlexRAM 中。

TaskPartition_FRAMNoLoadCSEcENMain()：验证的是将 D-Flash 分区为 16K 的 D-Flash 和 48K 的 E-Flash，同时使能 CSEc，MCU 复位时不自动加载 E-Flash 的数据到 FlexRAM 中。通过验证结果可以得出：该分区方法无法成功分区；所以用户在分区时使能了 CSEc，必须将 FlexRAM 的加载设置为自动加载。

代码请参见 **S32K1xx_SDK_RTM3_0_FlexNVMPEUsage** 示例工程。

7. S32K1xx系列MCU EEPROM常见问题(FAQ)

7.1 S32K1xx EEPROM 分区原则

EEPROM 的分区只能进行一次。如果为了不同的配置重新对 EEPROM 进行分区，会导致记录的数据丢失，并且无法保证获得预期的耐久性。

7.2 S32K1xx Mass Erase对EEPROM的影响

如果 MCU 被 Mass erased，则分区信息，EEPROM 数据和 EEPROM 位置信息将全部丢失。当 MCU 启用 EEPROM，且使能 security 时，强烈建议使用后门密钥。该后门密钥允许临时禁用 security 而无需执行批量擦除的方法。如果 MCU 被被 Mass erased，则不再能保证 EEPROM 的耐久性。如果 MCU 启用了 CSEc 功能，在任何批量擦除之前需要执行 DBG_AUTH 命令。

7.3 S32K1xx 系列MCU EEPROM 操作时间参数

Symbol	Description[1]		S32K142		S32K144		S32K146		S32K148			
			Typ	Max	Typ	Max	Typ	Max	Typ	Max	Unit	Notes
t_{setram}	Set FlexRAM Function execution time	Control Code 0xFF	0.08	—	0.08	—	0.08	—	0.08	—	ms	3
		32 KB EEPROM backup	0.8	1.2	0.8	1.2	0.8	1.2	—	—		
		48 KB EEPROM backup	1	1.5	1	1.5	1	1.5	—	—		
		64 KB EEPROM backup	1.3	1.9	1.3	1.9	1.3	1.9	1.3	1.9		
t_{eewr8b}	Byte write to FlexRAM execution time	32 KB EEPROM backup	385	1700	385	1700	385	1700	—	—	µs	3,4
		48 KB EEPROM backup	430	1850	430	1850	430	1850	—	—		
		64 KB EEPROM backup	475	2000	475	2000	475	2000	475	4000		
$t_{eewr16b}$	16-bit write to FlexRAM execution time	32 KB EEPROM backup	385	1700	385	1700	385	1700	—	—	µs	3,4
		48 KB EEPROM backup	430	1850	430	1850	430	1850	—	—		
		64 KB EEPROM backup	475	2000	475	2000	475	2000	475	4000		
$t_{eewr32bers}$	32-bit write to erased FlexRAM location execution time	—	360	2000	360	2000	360	2000	360	2000	µs	
$t_{eewr32b}$	32-bit write to FlexRAM execution time	32 KB EEPROM backup	630	2000	630	2000	630	2000	—	—	µs	3,4
		48 KB EEPROM backup	720	2125	720	2125	720	2125	—	—		
		64 KB EEPROM backup	810	2250	810	2250	810	2250	810	4500		
$t_{quickwr}$	32-bit Quick Write execution time: Time from CCIF clearing (start the write) until CCIF	1st 32-bit write	200	550	200	550	200	550	200	1100	µs	4,5,6
		2nd through Next to Last (Nth-1) 32-bit write	150	550	150	550	150	550	150	550		
	setting (32-bit write complete, ready for next 32-bit write)	Last (Nth) 32-bit write (time for write only, not cleanup)	200	550	200	550	200	550	200	550		
$t_{quickwrClnup}$	Quick Write Cleanup execution time	—	—	(# of Quick Writes) * 2.0	—	(# of Quick Writes) * 2.0	—	(# of Quick Writes) * 2.0	—	(# of Quick Writes) * 2.0	ms	7

7.4 S32K1xx EEPROM startup时间测试

时间测试包括：EEPROM 里面有 0K 的记录，有 1K 的记录；有 2K 的记录；有 4K 的记录，上电复位后，将数据从 E-Flash 复制到 FlexRAM 所需的时间。

测试方法如下所示：在 Reset_Handler 中，先将 PortD0 端口初始化，然后将该端口输出为高，然后等待 EERDY flag 被置 1 时，翻转 PortD0。这段时间可近视为：MCU 上电复位后，将数据从 E-Flash 复制到 FlexRAM 所需的时间。具体实现代码如下：

NOTE

为了测试不同的时间，每次需要手动修改main.c中EEPROMStartupTime_Write_NK的数据大小。每次数据写入EEPROM后，需要重新上电，通过示波器去测量这个时间。

EEPROMStartupTime.c

```
void EEPROMStartupTime_EEPROMTime(void)
{
    if(0xf != ((SIM->FCFG1 & SIM_FCFG1_DEPART_MASK) >>
SIM_FCFG1_DEPART_SHIFT))
    {
        while ((FTFC-> FCNFG & FTFC_FCNFG_EEERDY_MASK) == 0);
    }
    GPIO_LEDBlueToggle();

}
```

Main.c

```
EEPROMStartupTime_EEPROMInit();

EEPROMStartupTime_Write256BytesInit();

while(1)
{
    if(1u == Key_SW2())
    {
        EEPROMStartupTime_Write_NK(DataSize_1K);
    }
}
```

Startup_S32K144.S

```
Reset_Handler:
    cpsid   i               /* Mask interrupts */

/* Init Toggle GPIO */
    bl      GPIO_Init

    /* GPIO Output high */
    bl      GPIO_OutPutHigh

    bl EEPROMStartupTime_EEPROMTime
    bl      GPIO_Toggle

    bl      GPIO_Toggle
```

GPIO.c

```
void GPIO_Init(void)
{
    PCC-> PCCn[PCC_PORTD_INDEX] = PCC_PCCn_CGC_MASK; /* Enable clock
for PORT D */
    PTD->PDDR |= 1<<0;            /* Port D0:  Data Direction= output
*/
    PORTD->PCR[0] =  0x00000100;  /* Port D0:  MUX = ALT1: GPIO
function */

}
void GPIO_Toggle(void)
{
    *(uint32_t *)0x400FF0CC = 0x01u;
    //PTD->PTOR |= 1<<0;                /* Toggle output on port D0
(blue LED) */

}
void GPIO_OutPutHigh(void)
{
    *(uint32_t *)0x400FF0CC = 0x01u;  /* Output 0 on port D0 (blue LED)
*/
}
```

EEPROM 里面有 0K 的记录时，时间：396us（指 MCU 在汇编代码中，初始化通用寄存器之前，直到 EEREDY 置 1 等待的时间）

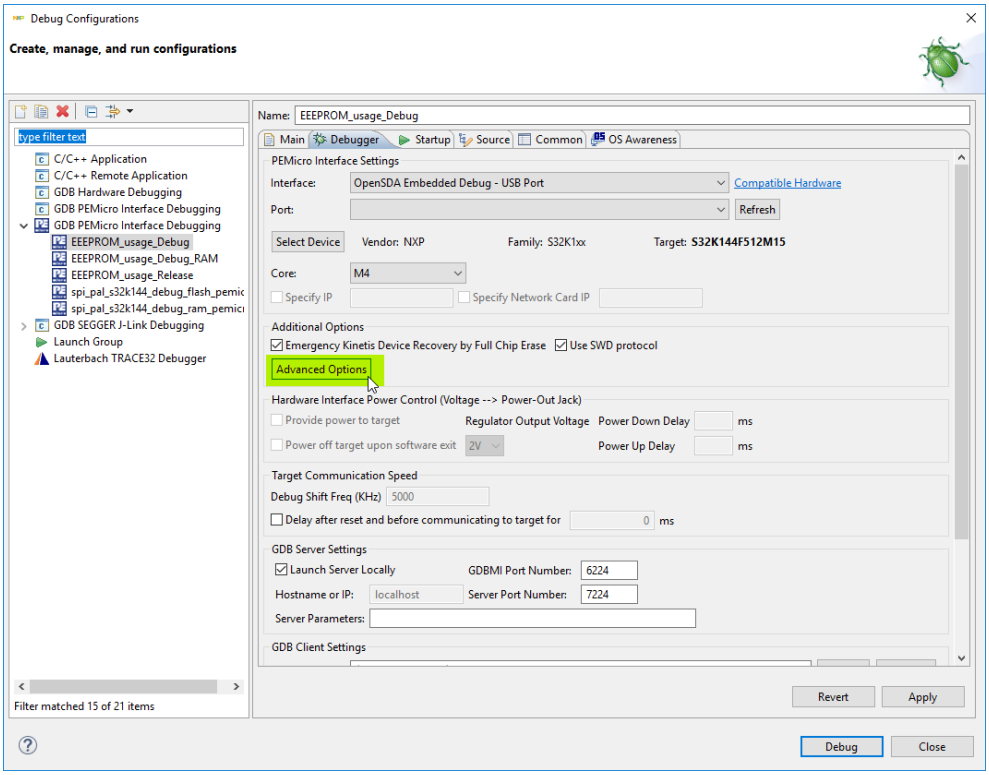
EEPROM 里面有 1K 的记录时，时间：403us（指 MCU 在汇编代码中，初始化通用寄存器之前，直到 EEREDY 置 1 等待的时间）

EEPROM 里面有 2K 的记录时，时间：420us（指 MCU 在汇编代码中，初始化通用寄存器之前，直到 EEREDY 置 1 等待的时间）

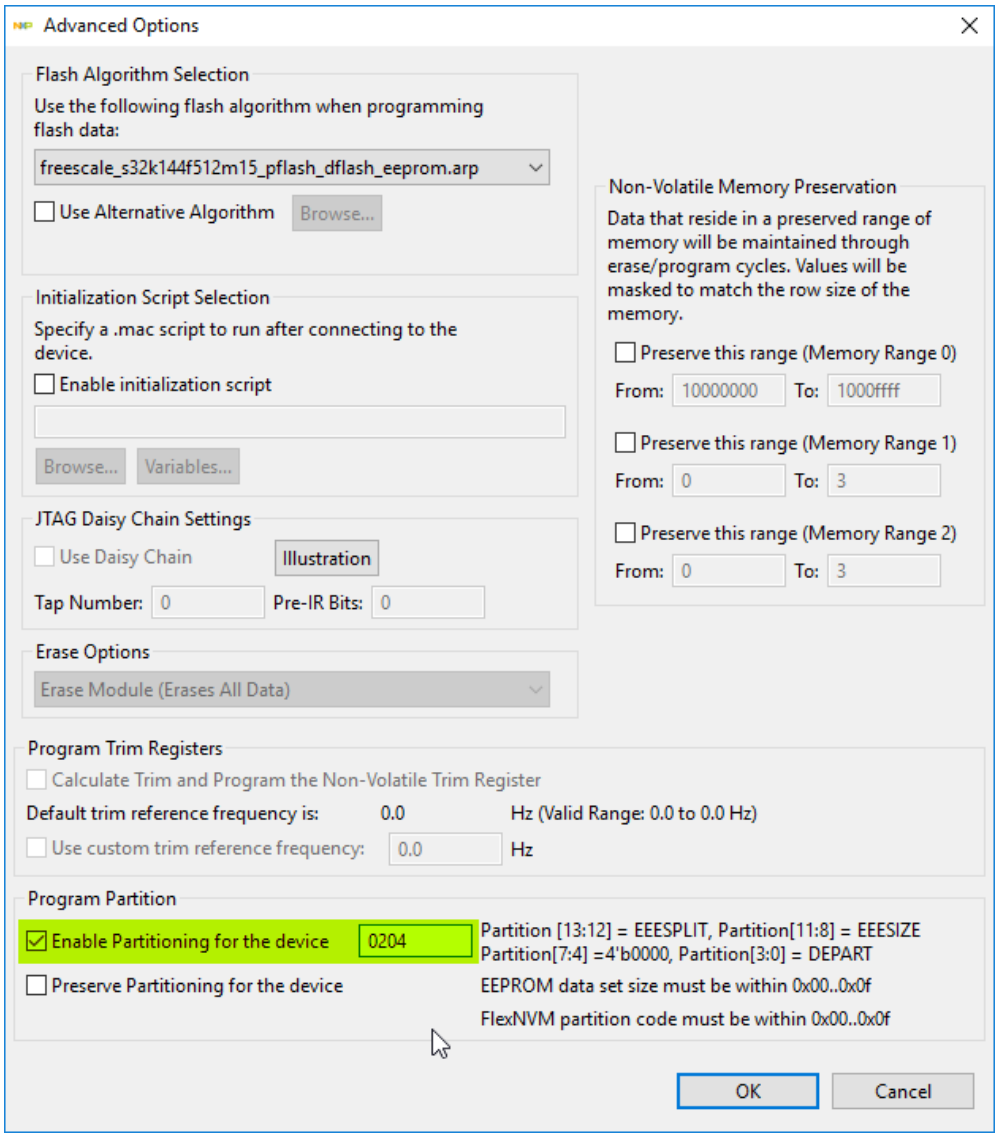
EEPROM 里面有 4K 的记录时，时间：458us（指 MCU 在汇编代码中，初始化通用寄存器之前，直到 EEREDY 置 1 等待的时间）

7.5 如何通过PE Micro分区EEPROM和将一个值写入S32K EEPROM?

用户可以使用 PE Micro 分区功能，并通过__attribute__说明符将你的数据添加到 FlexRAM 中（已由 PE Micro 划分为模拟 eeprom）。用户可以在调试配置 -> Advanced Options（高级选项）中找到分区选项，如下图所示。



用户可以通过 S32K1xx reference manual 找到更多关于 EEPROMSIZE 和 DEPART 的细节。其中 0204（0000_0010_0000_0100）即 EEPROMSIZE=0x2(4K)；DEPART=0x4(0K Dflash, 64K EEPROM)。其中，Preserve Partitioning for the device 选项的作用是：防止在用 PE 工具下载代码时，将之前的分区信息擦除掉。



然后修改应用工程的 Linker 文件，即在 Linker 文件中增加 flexram section，如下：

```
/* Specify the memory areas */
MEMORY
{
  /* Flash */
  m_interrupts          (RX)  : ORIGIN = 0x00000000, LENGTH = 0x00000400
  m_flash_config        (RX)  : ORIGIN = 0x00000400, LENGTH = 0x00000010
  m_text                (RX)  : ORIGIN = 0x00000410, LENGTH = 0x0007FBF0

  m_flexram             (RW)  : ORIGIN = 0x14000000, LENGTH = 0x00001000
  /* SRAM_L */
  m_data                (RW)  : ORIGIN = 0x1FFF8000, LENGTH = 0x00008000

  /* SRAM_U */
  m_data_2              (RW)  : ORIGIN = 0x20000000, LENGTH = 0x00007000
}

/* Define output sections */
SECTIONS
{

.eeeprom :
{
  KEEP(*(.eeeprom))
} > m_flexram

  /* The startup code goes first into internal flash */
  .interrupts :
  {
    __VECTOR_TABLE = .;
    . = ALIGN(4);
    KEEP(*(.isr_vector))     /* Startup code */
    . = ALIGN(4);
  } > m_interrupts
```

同时在 EEEPROM_data_def.c 中增加用.eeprom section 修饰的变量 eeprom_data = 10;如下:

```
__attribute__ ((section(".eeeprom"))) int eeerom_data=10;

/*!
  \brief The main function for the project.
  \details The startup initialization sequence is the following:
 * - startup asm routine
 * - main()
*/
int main(void)
{
    int ret;
  /* Write your local variable definition here */

  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  #ifdef PEX_RTOS_INIT
    PEX_RTOS_INIT();                 /* Initialization of the selected RTOS. Mac
  #endif
  /*** End of Processor Expert internal initialization.                    ***/

    CLOCK SYS Init(g clockManConfigsArr, CLOCK MANAGER CONFIG CNT,
```

代码请参见 **S32K1xx_SDK_RTM3_0_FlexNVMPEUsage** 示例工程。

7.6 产线批量生产Flash编程的方法和步骤

为了提高生产效率并降低产线 MCU 编程环节的 EOS 和 ESD，往往需要使用专门的 Flash 批量编程器并在产线上开发专门的编程工装，甚至研发自动化产线，将 MCU 编程作为自动化产线的一个有机部分，由电脑/工控机通过 USB/串口/网口连接批量 Flash 编程器进行自动化控制和编程结果记录跟踪。有关产线批量生产 Flash 编程的方法和步骤，请参考：产线批量生产 Flash 编程的方法和步骤。

8. 参考文献

- S32 Design Studio IDE for ARM® based MCUs
- S32K1xx Data Sheet
- S32K1XXRM, S32K1xx Series Reference Manual
- S32K144EVB: S32K144 Evaluation Board
- S32K MCU 应用文档，软件和开发工具请参考 www.nxp.com/s32k
- Getting Started with CSEc Security Module (document AN5401)
- Using S32K1xx EEPROM functionality (document AN11983)
- Using S32K148 FlexNVM memory (document AN12003)
- Production Flash Programming Best Practices for Kinetis K- and L-series MCUs (document AN4835)