# RL78 族, R7F0C 系列

数据闪存库的比较和应用

R01AN1944CC0100
Rev.1.00
2013.11.05

## 要点

在瑞萨官网有两种不同类型的数据闪存访问库可以对数据闪存进行访问。

该应用说明将从资源和 API 函数两方面来比较数据闪存库和 EEPROM 仿真库。通过提供样例程序，用户可以正确地评估这两个库。

该应用笔记旨在提供用户更多的信息，以帮助用户选择适合的数据闪存访问库。

## 目标器件

R5F104xA, R5F104xC

R7F0C001, R7F0C002, R7F0C010

## 目录

# 1. 数据闪存库 – 比较和应用

有 3 种不同类型的数据闪存库(FDL)可以对 RL78 内置的数据闪存进行访问。

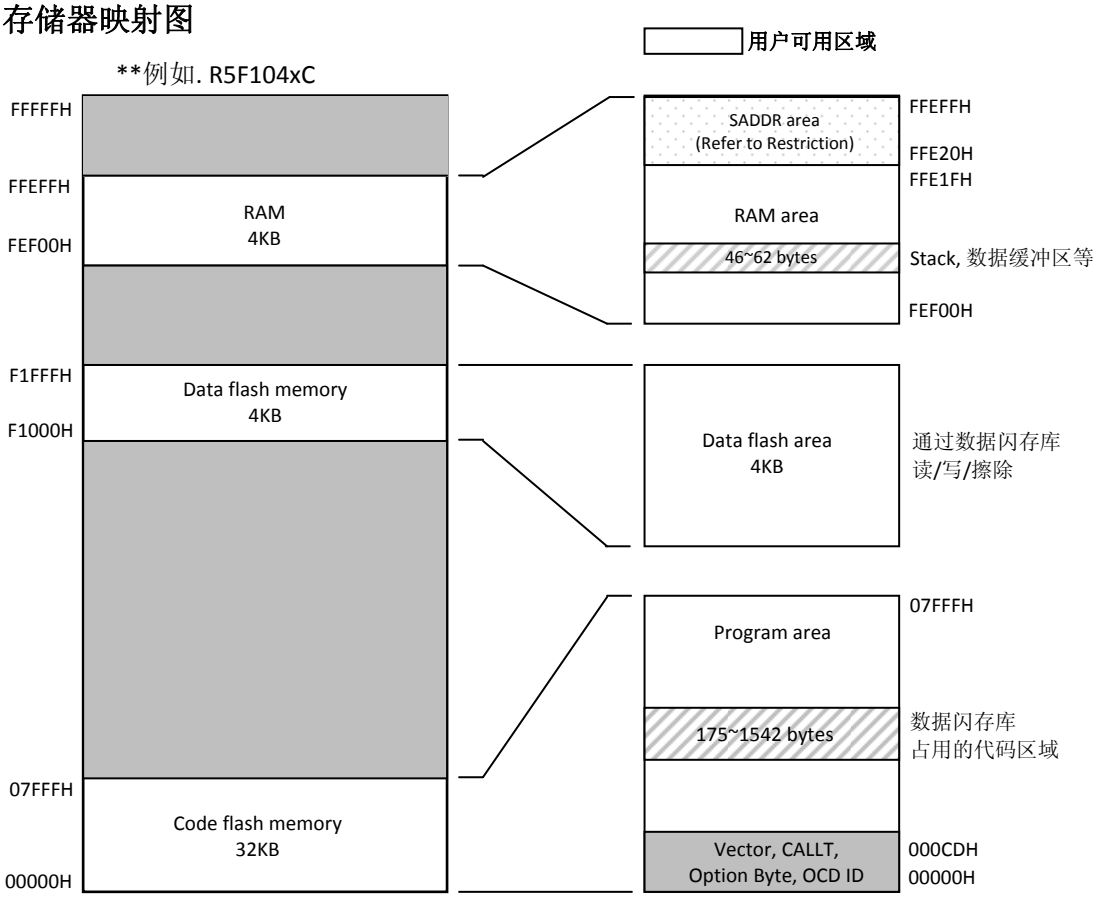
- 数据闪存访问库，T01
- 数据闪存访问库，T02 (Tiny)
- 数据闪存访问库，T04 (Pico)

下图从资源和功能两方面比较了这 3 种不同类型的库。

## 1.1 资源比较

<table>
<tr><th colspan="2">资源</th><th>T01</th><th>T02 (Tiny)</th><th>T04 (Pico)</th><th>描述 / 备注</th></tr>
<tr><td colspan="2">最大代码大小<br>代码 + 常量</td><td>1478 + 64<br>(1542 字节)</td><td>572 +10<br>(682 字节)</td><td>175 + 0<br>(175 字节)</td><td>分配在代码闪存区</td></tr>
<tr><td rowspan="6">工作<br>RAM</td><td>2.5KB RAM<br>R5F104xA<br>(G14)</td><td colspan="3">无限制</td><td rowspan="6">工作 RAM 是指被数据闪存访问库使用的 RAM 区。当使用 PFDL 时，每个设备有一些限制。<br>(细节请参考设备的用户手册)</td></tr>
<tr><td>4KB RAM<br>R5F104xC<br>(G14)</td><td colspan="3">无限制</td></tr>
<tr><td>1KB RAM<br>R7F0C001</td><td colspan="2">FFB00H~FFC89 (395 字节)</td><td>无限制</td></tr>
<tr><td>1.5KB RAM<br>R7F0C002</td><td colspan="2">FF900H~FFC80 (897 字节)</td><td>FF900H~FF9FF<br>(256 字节)</td></tr>
<tr><td>1.5KB RAM<br>R7F0C010</td><td colspan="2">FF900H~FFC80 (897 字节)</td><td>FF900H~FF9FF<br>(256 字节)</td></tr>
<tr><td>除上述设备外</td><td colspan="3">细节请参考用户硬件手册</td></tr>
<tr><td colspan="2">内部数据<br>(SADDR RAM)</td><td>2 字节</td><td>2 字节</td><td>FFE20H~FFEFFH<br>(224 字节)</td><td>SADDR RAM (FFE20H~FFEFFH) 是分配在 RAM 区的立即数访问区。</td></tr>
<tr><td colspan="2">最大 Stack 消耗</td><td>60 字节</td><td>56 字节</td><td>46 字节</td><td>除了 SelfRAM, SADDR RAM</td></tr>
<tr><td colspan="2">SADDR 区的限制</td><td colspan="3">Stack，数据缓冲区，函数的参数不能分配在该区域</td><td>地址 FFE20H~FFEFFH</td></tr>
</table>

## 1.2 存储器映射图



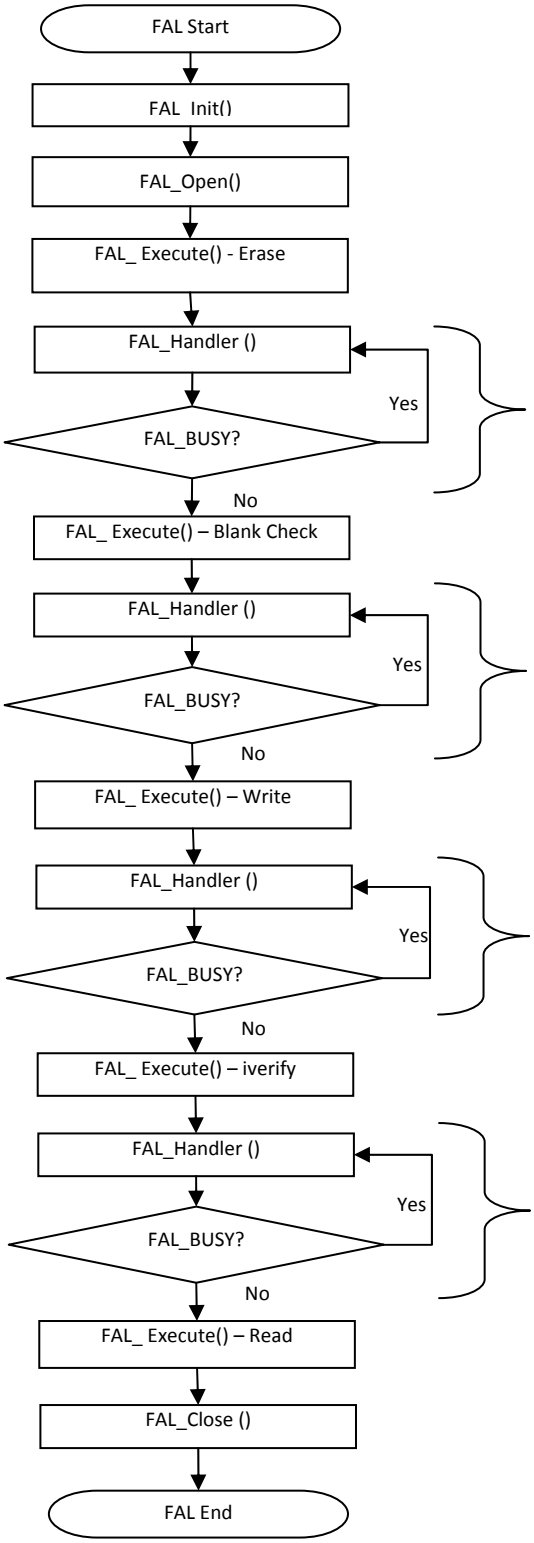

**注: 存储器映射图因不同的设备而不同，细节请参考设备的用户手册。

## 1.3 函数和 API

| 函数 \ 库 API | | T01 | T02 (Tiny) | T04 (Pico) | 描述 / 备注 |
|---|---|---|---|---|---|
| Init | | FAL_Init | FDL_Init | | 配置参数初始化内部数据 |
| Open | | FAL_Open | FDL_Open | PFDL_Open | 初始化 FDL 使用的 RAM 区，允许数据闪存时钟 |
| Close | | FAL_Close | FDL_Close | PFDL_Close | 结束 FDL 的操作，禁止数据闪存时钟 |
| Execute (Command) | Erase | FAL_Execute | FDL_Execute | PFDL_Execute | 触发并对数据闪存执行命令 |
| | BlankCheck | | | | |
| | Write | | | | |
| | IVerify | | | | |
| | Read | | | | |
| Handler | | FAL_Handler | FDL_Handler | PFDL_Handler | 检查数据闪存操作的当前状态并且驱使命令执行(状态检查处理) |
| Abort | | | FDL_Abort | | 放弃擦除命令 |
| StandBy | | | FDL_StandBy | | 使库进入待机模式(关掉数据闪存的硬件) |
| Wakeup | | | FDL_WakeUp | | 从待机模式唤醒库 |
| GetVersionString | | FAL_GetVersionString | FDL_GetVersionString | PFDL_GetVersionString | 获得 FDL 的版本信息 |

## 1.4 样例程序

### 1.4.1 程序流程图




****T01/T02/T04 的程序流程相同， 这里用 T01 的 API 作为样例。**

- T01 的 API 以 FAL_xxx()开头
- T02 的 API 以 FDL_xxx()开头
- T04 的 API 以 PFDL_xxx()开头

对于 API 的使用详解请参考数据闪存库的用户手册。

等待命令执行完成

执行状态显示在：

- T01/T02: request.status_enu
- T04: PFDL_Handler()的返回值

等待命令执行完成

执行状态显示在：

- T01/T02: request.status_enu
- T04: PFDL_Handler()的返回值

等待命令执行完成

执行状态显示在：

- T01/T02: request.status_enu
- T04: PFDL_Handler()的返回值

等待命令执行完成

执行状态显示在：

- T01/T02: request.status_enu
- T04: PFDL_Handler()的返回值

### 1.4.2 数据闪存访问库 T01

```
extern __far const fal_descriptor_t fal_descriptor_str;
fal_status_t my_fal_status_enu;
__near fal_request_t request;

/* initialization */
my_fal_status_enu = FAL_Init((const __far fal_descriptor_t*)&fal_descriptor_str );
if(my_fal_status_enu != FAL_OK)
        ErrorHandler();
FAL_Open();

/* erase block 0 */
request.index_u16 = 0x0000;
request.command_enu = FAL_CMD_ERASE_BLOCK;
FAL_Execute(&request);
while(request.status_enu == FAL_BUSY)
        FAL_Handler();
if(request.status_enu != FAL_OK)
        ErrorHandler();

/* blank check widx = 0 */
request.index_u16 = 0x0000;
request.command_enu = FAL_CMD_BLANKCHECK_WORD;
FAL_Execute(&request);
while(request.status_enu == FAL_BUSY)
        FAL_Handler();
if(request.status_enu != FAL_OK)
        ErrorHandler();

/* write patter 0x12345678 into the widx = 0 */
request.index_u16 = 0x0000;
request.data_u32 = 0x12345678;
request.command_enu = FAL_CMD_WRITE_WORD;
FAL_Execute(&request);
while(request.status_enu == FAL_BUSY)
        FAL_Handler();
if(request.status_enu != FAL_OK)
        ErrorHandler();

/* verify widx = 0 */
request.index_u16 = 0x0000;
request.command_enu = FAL_CMD_IVERIFY_WORD;
FAL_Execute(&request);
while(request.status_enu == FAL_BUSY)
        FAL_Handler();
if(request.status_enu != FAL_OK)
        ErrorHandler();

/* read value of widx = 0 */
request.index_u16 = 0x0000;
request.command_enu = FAL_CMD_READ_WORD;
FAL_Execute(&request);
if(request.status_enu != FAL_OK)
        ErrorHandler();

/* check whether the written pattern is correct */
if(request.data_u32 != 0x12345678)
        ErrorHandler();

FAL_Close();
```

### 1.4.3 数据闪存访问库 T02 (Tiny)

```
extern __far const fdl_descriptor_t fdl_descriptor_str;
fdl_status_t my_fdl_status_enu;
__near fdl_request_t request;
fdl_u08 buffer[5];

/* initialization */
my_fdl_status_enu = FDL_Init((__far fdl_descriptor_t*)&fdl_descriptor_str );
if(my_fdl_status_enu != FDL_OK)
        ErrorHandler();
FDL_Open();

/* request structure initialization */
request.index_u16 = 0x0000;
request.data_pu08 = (__near fdl_u08*) 0x0000;
request.bytecount_u16 = 0x0000;
request.command_enu = (fdl_command_t)0xFF;
request.status_enu = FDL_ERR_PARAMETER;

/* erase block 0 */
request.index_u16 = 0x0000;
request.command_enu = FDL_CMD_ERASE_BLOCK;
FDL_Execute(&request);
while(request.status_enu == FDL_BUSY)
        FDL_Handler();
if(request.status_enu != FDL_OK)
        ErrorHandler();

/*blank Check*/
request.index_u16 = 0x0000;
request.bytecount_u16 = 0x0005;
request.command_enu = FDL_CMD_BLANKCHECK_BYTES;
FDL_Execute(&request);
while(request.status_enu == FDL_BUSY)
        FDL_Handler();
if(request.status_enu != FDL_OK)
        ErrorHandler();

/* write pattern 0x123456789A to idx = 0 */
buffer[0] = 0x12;
buffer[1] = 0x34;
buffer[2] = 0x56;
buffer[3] = 0x78;
buffer[4] = 0x9A;
request.index_u16 = 0x0000;
request.data_pu08 = (__near fdl_u08*)&buffer[0];
request.bytecount_u16 = 0x0005;
request.command_enu = FDL_CMD_WRITE_BYTES;
FDL_Execute(&request);
while(request.status_enu == FDL_BUSY)
        FDL_Handler();
if(request.status_enu != FDL_OK)
        ErrorHandler();

/*iverify*/
request.index_u16 = 0x0000;
request.bytecount_u16 = 0x0005;
request.command_enu = FDL_CMD_IVERIFY_BYTES;
FDL_Execute(&request);
while(request.status_enu == FDL_BUSY)
        FDL_Handler();
if(request.status_enu != FDL_OK)
        ErrorHandler();

/* set initial values */
buffer[0] = 0xFF;
buffer[1] = 0xFF;
buffer[2] = 0xFF;
```

```
buffer[3] = 0xFF;
buffer[4] = 0xFF;
request.index_u16 = 0x0000;
request.data_pu08 = (__near fdl_u08*)&buffer[0];
request.bytecount_u16 = 0x0005;
request.command_enu = FDL_CMD_READ_BYTES;
FDL_Execute(&request);
if(request.status_enu != FDL_OK)
        ErrorHandler();

FDL_Close();
```

### 1.4.4 数据闪存访问库 T04 (Pico)

```
__far pfdl_descriptor_t pfdl_descriptor_str;
pfdl_status_t my_pfdl_status_enu;
__near pfdl_request_t request;
pfdl_u08 buffer[5];

/* initialization */
pfdl_descriptor_str.fx_MHz_u08          = FDL_FRQ;                    //CPU clock
pfdl_descriptor_str.wide_voltage_mode_u08 = FDL_VOL;                  //Voltage mode
my_pfdl_status_enu = PFDL_Open((__far pfdl_descriptor_t*)&pfdl_descriptor_str );
if(my_pfdl_status_enu != PFDL_OK)
        ErrorHandler();

/* erase block 0 */
request.index_u16   = 0;
request.command_enu = PFDL_CMD_ERASE_BLOCK;
my_pfdl_status_enu = PFDL_Execute(&request);
while(my_pfdl_status_enu == PFDL_BUSY)
        my_pfdl_status_enu = PFDL_Handler();
if(my_pfdl_status_enu != PFDL_OK)
        ErrorHandler();

/* Blank Check */
request.index_u16   = 0;
request.bytecount_u16 = 0x0005;
request.command_enu = PFDL_CMD_BLANKCHECK_BYTES;
my_pfdl_status_enu = PFDL_Execute(&request);
while(my_pfdl_status_enu == PFDL_BUSY)
        my_pfdl_status_enu = PFDL_Handler();
if(my_pfdl_status_enu != PFDL_OK)
        ErrorHandler();

/* write pattern 0x123456789A to idx = 0 */
buffer[0] = 0x12;
buffer[1] = 0x34;
buffer[2] = 0x56;
buffer[3] = 0x78;
buffer[4] = 0x9A;
request.index_u16 = 0x0000;
request.data_pu08 = (__near pfdl_u08*)&buffer[0];
request.bytecount_u16 = 0x0005;
request.command_enu = PFDL_CMD_WRITE_BYTES;
my_pfdl_status_enu = PFDL_Execute(&request);
while(my_pfdl_status_enu == PFDL_BUSY)
        my_pfdl_status_enu = PFDL_Handler();
if(my_pfdl_status_enu != PFDL_OK)
        ErrorHandler();

/* Iverify */
request.index_u16   = 0;
request.bytecount_u16 = 0x0005;
request.command_enu = PFDL_CMD_IVERIFY_BYTES;
my_pfdl_status_enu = PFDL_Execute(&request);
while(my_pfdl_status_enu == PFDL_BUSY)
        my_pfdl_status_enu = PFDL_Handler();
if(my_pfdl_status_enu != PFDL_OK)
        ErrorHandler();

/* set initial values */
buffer[0] = 0xFF;
buffer[1] = 0xFF;
buffer[2] = 0xFF;
buffer[3] = 0xFF;
buffer[4] = 0xFF;
/* Read to buffer[0]*/
request.index_u16 = 0x0000;
request.data_pu08 = (__near pfdl_u08*)&buffer[0];
request.bytecount_u16 = 0x0005;
request.command_enu = PFDL_CMD_READ_BYTES;
```

```
my_pfdl_status_enu = PFDL_Execute(&request);
if(my_pfdl_status_enu != PFDL_OK)
        ErrorHandler();
PFDL_Close();
```

## 1.5 比较和注意事项

| | T01 | T02 (Tiny) | T04 (Pico) |
|---|---|---|---|
| 读/写的最小字节数 | 4 字节 | 1 字节 | 1 字节 |
| 读/写的最大字节数 | 4 字节 | 4k 字节 | 4k 字节 |
| 地址对齐 | 4 字节 | 1 字节 | 1 字节 |
| API 函数的个数 | 6 | 9 | 5 |
| EEL 支持 | 支持 | 支持 | 不支持 |
| 命令执行状态 | 命令请求 status_enu | 命令请求 status_enu | PFDL_Execute()的返回值 |

## 1.6 函数最长执行时间比较

| 函数 (API) | | T01 | T02 (Tiny) | T04 (Pico) |
|---|---|---|---|---|
| Init | | 没有命令正在执行:<br>1758/fclk<br><br>命令正在后台执行:<br>2092/fclk + 60us | 1199/fclk | |
| Execute | Erase | 1259/fclk + 28us | 646/fclk | 536/fclk |
| | Blank Check | | | 484 / fclk |
| | Write | | | 549/fclk |
| | Iverify | | | 502/fclk |
| | Read | | 167/fclk +(17/fclk x 字节数) | 53/fclk+(17/fclk x 字节数) |
| Handler | | 974/fclk + 15us | 284/fclk + 15us | 251/fclk + 14us |
| Open | | 83/fclk + 12us | 27/fclk + 14us | 536/fclk |
| Close | | 没有命令正在执行:<br>42/fclk<br><br>命令正在后台执行:<br>388/fclk + 60us | 没有命令正在执行:<br>30/fclk<br><br>命令正在后台执行:<br>836/fclk + 444us | 823/fclk +443us |
| Standby | | | 305/fclk | |
| WakeUp | | | 32/fclk | |
| Abort | | | 350/fclk | |
| GetVersionString | | 14/fclk | 14/fclk | 10/fclk |

# 2. EEPROM 仿真库 – 比较和应用

在数据闪存库之上，用户可以使用 EEPROM 仿真库来访问数据闪存区，该方法类似于访问外部 EEPROM。通过使用这些库，不用管理对数据闪存写和擦除操作。

有两种不同的 EEPROM 仿真库可以对 RL78 数据闪存进行访问。

- EEPROM 仿真库, T01
- EEPROM 仿真库, T02 (Tiny)

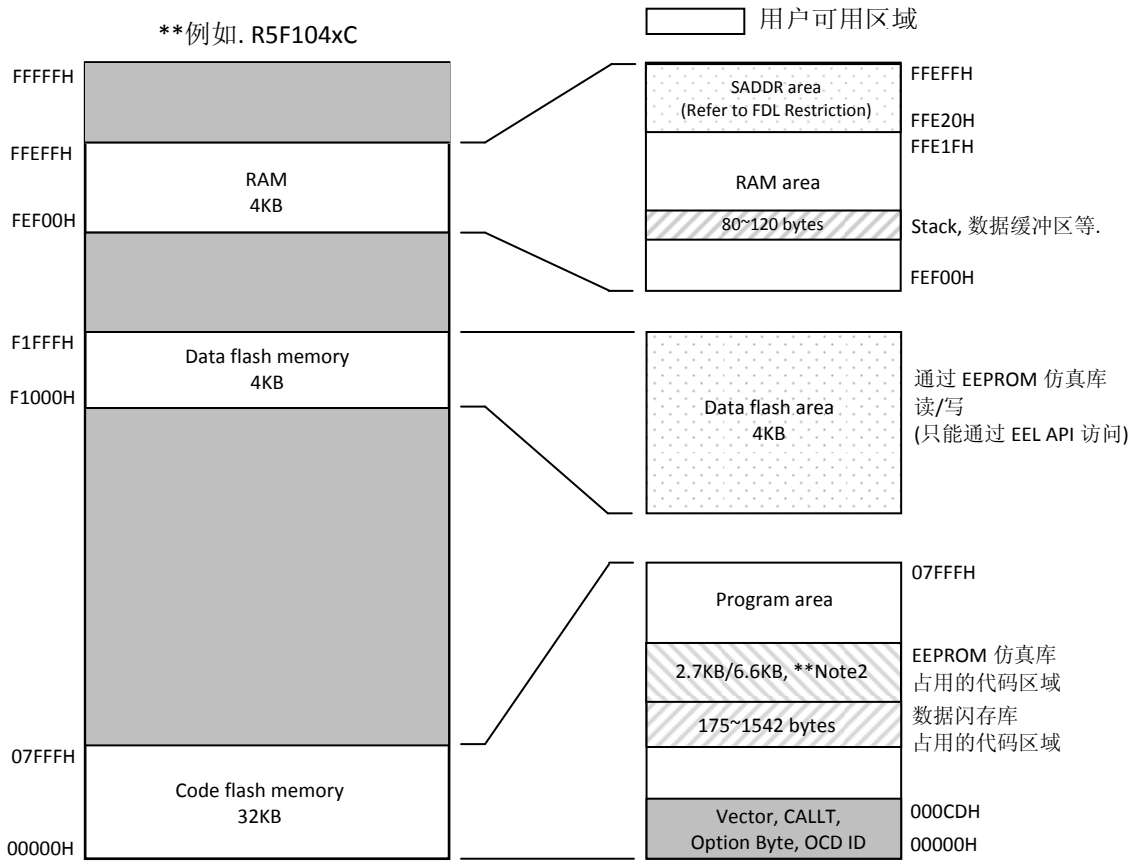
备注: EEPROM 仿真库没有 T04。

下图从资源和功能两方面比较了这 2 种不同类型的库。

## 2.1 资源比较

| 资源 | T01 | T02 (Tiny) | 描述 / 备注 |
|---|---|---|---|
| 最大代码大小<br>代码 + 常量 | 6.6k 字节+ (4 +( EEL 变量数)*4) | 2746 字节+ ( 4 + EEL 变量数) | 分配在代码闪存区 |
| 最小数据闪存 | 4 Block | 2 Block | 所需的最小数据闪存大小 (1KB/Block) |
| RAM | 1 字节 | - | 工作 RAM 是指 MCU 的 RAM 区。 |
| 内部数据<br>(SADDR RAM) | 9 字节 | 1 字节 | SADDR RAM (FFE20H~FFF1FH) 是分配在 RAM 区的立即数访问区。 |
| 最大 Stack 消耗<br>(包含 FDL) | <120 字节 | 80 字节 | 除了 SelfRAM, SADDR RAM |
| 所需的 FDL | FDL T01 | FDL T02 (Tiny) | 所需的数据闪存库 |

## 2.2 存储器映射图




**注 1: 存储器映射图因不同的设备而不同，细节请参考设备的用户手册。
注 2: 区域大小取决于总的变量数，下一章将计算 ROM 的大小。

## 2.3 所占用 ROM 的计算

EEPROM 仿真所占用的 ROM 区包含 FDL，EEL 和 EEL 描述符。FDL 和 EEL 的大小是固定的，但是 EEL 描述符的大小取决于应用。本章显示如何计算 ROM 大小（基于样例程序）。

### 2.3.1 EEL 描述符大小的计算

EEL 描述符大小取决于定义在 eel_descriptor.c 中的变量数，EEL T01 和 T02 的变量数是 8。

(1) **EEL T01 描述符**

```
__far const eel_u08 eel_descriptor[EEL_VAR_NO+1][4] =
{
/*  identifier        word-size (1...64)                 byte-size (1..255)         RAM-Ref. */
/*  ------------------------------------------------------------------------------------------ */
   (eel_u08)'a',     (eel_u08)((sizeof(type_A)+3)/4),    (eel_u08)sizeof(type_A),   0x01,  \
   (eel_u08)'b',     (eel_u08)((sizeof(type_B)+3)/4),    (eel_u08)sizeof(type_B),   0x01,  \
   (eel_u08)'c',     (eel_u08)((sizeof(type_C)+3)/4),    (eel_u08)sizeof(type_C),   0x01,  \
   (eel_u08)'d',     (eel_u08)((sizeof(type_D)+3)/4),    (eel_u08)sizeof(type_D),   0x01,  \
   (eel_u08)'e',     (eel_u08)((sizeof(type_E)+3)/4),    (eel_u08)sizeof(type_E),   0x01,  \
   (eel_u08)'f',     (eel_u08)((sizeof(type_F)+3)/4),    (eel_u08)sizeof(type_F),   0x01,  \
   (eel_u08)'x',     (eel_u08)((sizeof(type_X)+3)/4),    (eel_u08)sizeof(type_X),   0x01,  \
   (eel_u08)'z',     (eel_u08)((sizeof(type_Z)+3)/4),    (eel_u08)sizeof(type_Z),   0x01,  \
   (eel_u08)0x00,    (eel_u08)(0x00),                    (eel_u08)(0x00),           0x00,  \
};
```

上述 EEL T01 描述符的大小：(变量数 + 1) x 4 = 36 字节.

(2) **EEL T02 描述符**

```
__far const eel_u08 eel_descriptor[EEL_VAR_NO+2] =
{
  (eel_u08)(EEL_VAR_NO),       /* variable count   */  \
  (eel_u08)(sizeof(type_A)),   /* id=1             */  \
  (eel_u08)(sizeof(type_B)),   /* id=2             */  \
  (eel_u08)(sizeof(type_C)),   /* id=3             */  \
  (eel_u08)(sizeof(type_D)),   /* id=4             */  \
  (eel_u08)(sizeof(type_E)),   /* id=5             */  \
  (eel_u08)(sizeof(type_F)),   /* id=6             */  \
  (eel_u08)(sizeof(type_X)),   /* id=7             */  \
  (eel_u08)(sizeof(type_Z)),   /* id=8             */  \
  (eel_u08)(0x00),             /* zero terminator  */  \
}
```

上述 EEL T02 描述符的大小： (变量数 + 2) = 10 字节.

### 2.3.2 EEPROM 仿真的 ROM 大小

计算基于 2.3.1 节, 样例程序使用了 8 个 EEL 描述符。实际的大小取决于实际使用的描述符。

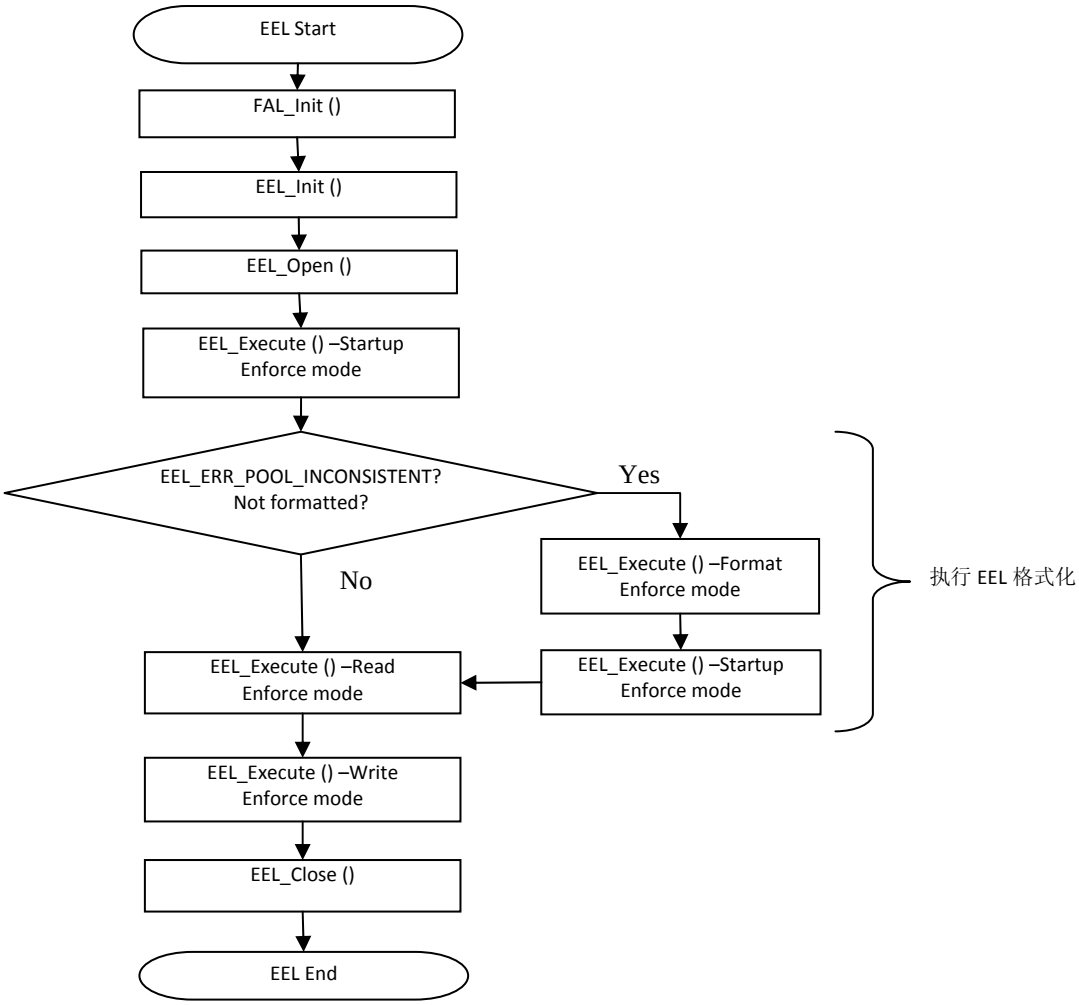
| 库\ 资源 | FDL | EEL | 描述符 (8 个变量) | 总共 ROM 大小 |
|---|---|---|---|---|
| T01 | 1542 字节 | 6.6K 字节 | 36 字节 | **8178 字节** |
| T02 (Tiny) | 682 字节 | 2746 字节 | 10 字节 | **3438 字节** |

## 2.4 EEL 函数和 API

| 函数 \ API | T01 | | T02 (Tiny) | | 描述 / 备注 |
|---|---|---|---|---|---|
| Init | EEL_Init | | EEL_Init | | 配置参数初始化内部数据 |
| Open | EEL_Open | | EEL_Open | | 打开 EEL |
| Close | EEL_Close | | EEL_Close | | 关闭 EEL |
| Execute (Command) | EEL_Execute | Startup | EEL_Execute | Startup | 触发并执行命令 |
| | | Shutdown | | Shutdown | |
| | | Clean Up | | | |
| | | | | Verify | |
| | | Format | | Format | |
| | | | | Refresh | |
| | | Write | | Write | |
| | | Read | | Read | |
| Handler | EEL_Handler | | EEL_Handler | | 检查当前状态并且驱使命令执行(状态检查处理) |
| TimeOut Count Down | EEL_TimerOut_CountDown | | | | 启动递减计数器 |
| Get Driver Status | EEL_GetDriverStatus | | EEL_GetDriverStatus | | 在发送请求前检查 EEL 驱动的内部状态 |
| Get Space | EEL_GetSpace | | EEL_GetSpace | | 检查当前块的剩余空间 |
| Get Version String | EEL_GetVersionString | | EEL_GetVersionString | | 获得 EEL 的版本信息 |

## 2.5 样例程序

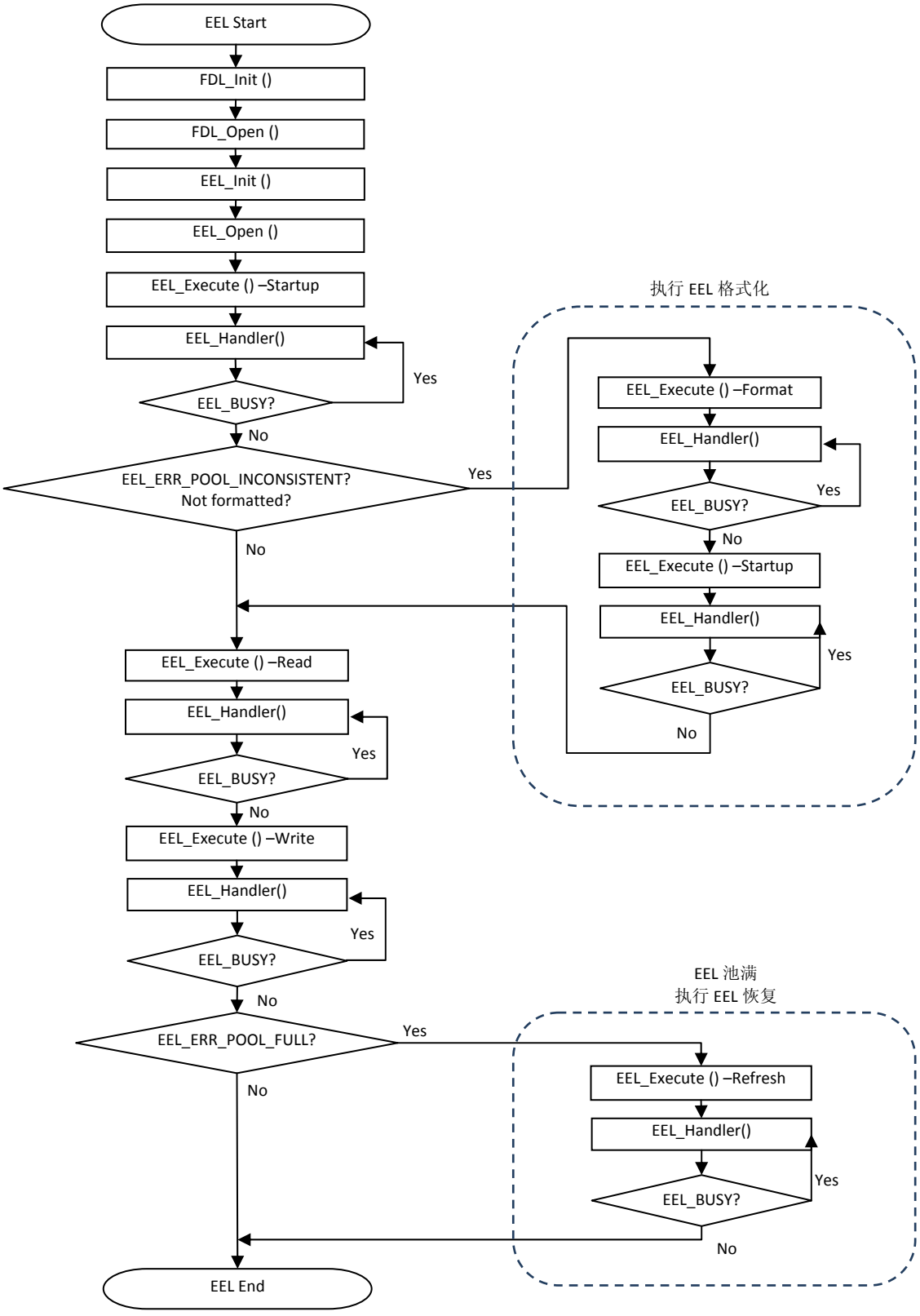
### 2.5.1 EEPROM 仿真库 T01 样例程序流程图



### 2.5.2 EEPROM 仿真库 T01

```
extern __far const fal_descriptor_t fal_descriptor_str;
fal_status_t my_fal_status_enu;
eel_request_t my_eel_request;

/*******Variable that match the data type defined in eel_user_types.h******/
eel_u08 A[2];

/* initialization */
my_fal_status_enu = FAL_Init((const __far fal_descriptor_t*)&fal_descriptor_str );
if(my_fal_status_enu != FAL_OK)
        ErrorHandler();
FAL_Open();

/* EEL init */
ret = EEL_Init();
if (ret != EEL_OK)
        ErrorHandler();
EEL_Open(); /* data flash clock starts here controlled */

/* specification of a time limited STARTUP request */
my_eel_request.command_enu = EEL_CMD_STARTUP;
my_eel_request.timeout_u08 = 255;
EEL_Execute(&my_eel_request);
if (my_eel_request.status_enu == EEL_ERR_POOL_INCONSISTENT)//Not format?
{
        /* specification of a time limited FORMAT request */
        my_eel_request.command_enu = EEL_CMD_FORMAT;
        my_eel_request.timeout_u08 = 0xFF;
        EEL_Execute(&my_eel_request);
        if (my_eel_request.status_enu != EEL_OK)
                ErrorHandler();

        /* specification of a time limited STARTUP request */
        my_eel_request.command_enu = EEL_CMD_STARTUP;
        my_eel_request.timeout_u08 = 255;
        EEL_Execute(&my_eel_request);
        if (my_eel_request.status_enu != EEL_OK)
                ErrorHandler();
}
else if (my_eel_request.status_enu != EEL_OK)
        ErrorHandler();

/* specification of a time limited READ request */
my_eel_request.address_pu08 = (eel_u08*)&A[0];      //data
my_eel_request.identifier_u08 = 'a';  //id
my_eel_request.command_enu = EEL_CMD_READ;
my_eel_request.timeout_u08 = 255;
EEL_Execute(&my_eel_request);
if (my_eel_request.status_enu != EEL_OK)
        ErrorHandler();

/* specification of a time limited WRITE request */
my_eel_request.address_pu08 = (eel_u08*)&A[0];      //data
my_eel_request.identifier_u08 = 'a';  //id
my_eel_request.command_enu = EEL_CMD_WRITE;
my_eel_request.timeout_u08 = 255;
EEL_Execute(&my_eel_request);
if (my_eel_request.status_enu != EEL_OK)
        ErrorHandler();

EEL_Close();
```

### 2.5.3 EEPROM 仿真库 T02 样例程序流程图



### 2.5.4 EEPROM 仿真库 T02 (Tiny)

```
extern __far const fdl_descriptor_t fdl_descriptor_str;
fdl_status_t my_fdl_status_enu;
__near fdl_request_t request;                :
/*******Variable that match the data type defined in eel_user_types.h******/
eel_u08 A[2];

/* initialization */
my_fdl_status_enu = FDL_Init((__far fdl_descriptor_t*)&fdl_descriptor_str );
if(my_fdl_status_enu != FDL_OK)
        ErrorHandler();
FDL_Open();

/* Ensure initialized request structure ------------------------------ */
my_eel_request_str.command_enu = EEL_CMD_UNDEFINED;
my_eel_request_str.identifier_u08 = 0;
my_eel_request_str.address_pu08 = 0;
my_eel_request_str.status_enu = EEL_ERR_PARAMETER;
my_refr_eel_request_str.command_enu = EEL_CMD_REFRESH;
my_refr_eel_request_str.identifier_u08 = 0;
my_refr_eel_request_str.address_pu08 = 0;
my_refr_eel_request_str.status_enu = EEL_ERR_PARAMETER;

/* It is assumed at this point, that the FDL has been initialized and opened correctly. */
my_eel_status = EEL_Init();
if (my_eel_status != EEL_OK)
        ErrorHandler();
EEL_Open();

/* Ensure initialized request structure ------------------------------ */
my_eel_request_str.command_enu = EEL_CMD_UNDEFINED;
my_eel_request_str.identifier_u08 = 0;
my_eel_request_str.address_pu08 = 0;
my_eel_request_str.status_enu = EEL_ERR_PARAMETER;
/* Initiate STARTUP command ----------------------------------------- */
my_eel_request_str.command_enu = EEL_CMD_STARTUP;
/* command cannot be rejected here as the library has just been opened */
EEL_Execute(&my_eel_request_str);
while (my_eel_request_str.status_enu == EEL_BUSY)
{
        EEL_Handler();
}
if (my_eel_request_str.status_enu == EEL_ERR_POOL_INCONSISTENT)
{
        /* Initiate FORMAT comamnd ------------------------------------- */
        my_eel_request_str.command_enu = EEL_CMD_FORMAT;
        /*command cannot be rejected here as the library has just been opened */
        EEL_Execute(&my_eel_request_str);
        while (my_eel_request_str.status_enu == EEL_BUSY)
        {
                EEL_Handler();
        }

        if (my_eel_request_str.status_enu != EEL_OK)
        {
                ErrorHandler();
        }

        /* EEL pool is formatted now, all previous variable content is lost */
        /* Ensure initialized request structure ------------------------- */
        my_eel_request_str.command_enu = EEL_CMD_UNDEFINED;
        my_eel_request_str.identifier_u08 = 0;
        my_eel_request_str.address_pu08 = 0;
        my_eel_request_str.status_enu = EEL_ERR_PARAMETER;
        /* Initiate STARTUP command ------------------------------------ */
        my_eel_request_str.command_enu = EEL_CMD_STARTUP;
        /*command cannot be rejected here as the library has just been opened */
        EEL_Execute(&my_eel_request_str);
        while (my_eel_request_str.status_enu == EEL_BUSY)
        {
                EEL_Handler();
        }
        if (my_eel_request_str.status_enu != EEL_OK)
```

```
        {
                ErrorHandler();
        }
}
else if (my_eel_request_str.status_enu != EEL_OK)
{
                ErrorHandler();
}

/* Initiate READ command ---------------------------------------------- */
my_eel_request_str.command_enu = EEL_CMD_READ;
my_eel_request_str.address_pu08 = (__near eel_u08*)&A[0];   //data buffer
my_eel_request_str.identifier_u08 = 1;       //id
do
{
        EEL_Handler();
        EEL_Execute(&my_eel_request_str);
}
while (my_eel_request_str.status_enu == EEL_ERR_REJECTED);
while (my_eel_request_str.status_enu == EEL_BUSY)
{
        EEL_Handler();
}
if (my_eel_request_str.status_enu != EEL_OK)
{
        ErrorHandler();
}

/* Initiate WRITE command --------------------------------------------- */
my_eel_request_str.command_enu = EEL_CMD_WRITE;
my_eel_request_str.address_pu08 = (__near eel_u08*)&A[0];   //Data buffer
my_eel_request_str.identifier_u08 = 1;       //id
do
{
        do
        {
                EEL_Handler();
                EEL_Execute(&my_eel_request_str);
        }
        while (my_eel_request_str.status_enu == EEL_ERR_REJECTED);
        while (my_eel_request_str.status_enu == EEL_BUSY)
        {
                EEL_Handler();
        }
        if (my_eel_request_str.status_enu == EEL_ERR_POOL_FULL)
        {
                /* in case of a full pool, a refresh needs to be executed
                in order to free space */
                /* please note that a different request variable is used for the
                refresh so that the result of the write command can be checked in the
                outer loop */
                do
                {
                        EEL_Handler();
                        EEL_Execute(&my_refr_eel_request_str);
                }
                while (my_refr_eel_request_str.status_enu == EEL_ERR_REJECTED);
                while (my_refr_eel_request_str.status_enu == EEL_BUSY)
                {
                        EEL_Handler();
                }
                if (my_refr_eel_request_str.status_enu != EEL_OK)
                {
                        ErrorHandler();
                }
        }
        else if (my_eel_request_str.status_enu != EEL_OK)
        {
                ErrorHandler();
        }
}
while (my_eel_request_str.status_enu == EEL_ERR_POOL_FULL);

/* Terminate EEL and FDL ---------------------------------------------- */
EEL_Close();
FDL_Close();
```

## 2.6 比较和注意事项

| | | T01 | T02 (Tiny) |
|---|---|---|---|
| 数据结构的最小字节数 | | 4 字节 | 1 字节 |
| 数据结构的最大字节数 | | 255 字节 | 255 字节 |
| API 函数的个数 | | 9 | 8 |
| 执行模式 | 轮询模式 | 支持 | 支持 |
| | 超时模式 | 支持 | |
| | 强制模式 | 支持 | |
| 恢复操作<br>(参考 EEL 的用户手册) | | 在 EEL 中处理 | 在用户程序中处理 |

## 2.7 重写次数的计算 (参考)

计算基于把 16 字节的变量写到 4K 的数据闪存。1 次擦除 + 擦除后的 1 次写视为 1 次重写。在 EEL 中，库通过恢复操作自动执行擦除。

对于 RL78 G13/G14，可以保证数据被保存 5 年的最小重写次数为 100,000 次。

EEL T01 的情况:

从实际操作来看，执行整块擦除之前 16 字节的变量可以被写 160 次，所以最小的重写次数为：

最小重写次数 = 100,000 x 160 = 16,000,000 次.

EEL T02 的情况:

从实际操作来看，执行整块擦除之前 16 字节的变量可以被写 220 次，所以最小的重写次数为：

最小重写次数 = 100,000 x 220 = 22,000,000 次.

## 3. 链接指令文件设置

### 内存指令

内存指令是定义一个内存区间（内存的地址和名字）的指令。

定义的内存区间可以被段定位指令通过使用该名字（内存区间的名字）来引用。

最多可以定义 100 个内存区间（包含默认定义的内存区间）。以下是语法:

```
MEMORY memory-area-name : ( start-address , size ) [/ memory-space-name]
```

### 段定位指令

段定位指令是把一个指定的段放在指定的内存区间或指定的地址的指令。以下是语法:

```
MERGE segment-name : [AT ( start-address )] [ = memory-area-name] [ / memory-space-name]
MERGE segment-name : [merge-attribute] [ = memory-area-name] [ / memory-space-name]
```

下面例子是针对 EEL T02.p 的链接指令

- RAM 区定义在 H’FF700~H’FFE1F. 用户程序将访问该 RAM 区.
- RAM_SADDR 定义在 H’FFE20~H’FFEFF.
- 段 EEL_SDAT and FDL_SDAT 分配在 RAM_SADDR 区内.
- 段 EEL_CODE and FDL_CODE 分配在 ROM 区内.
- 段 EEL_CNST and FDL_CNST 分配在 ROM 区内.

```
;*******************************************************************************
; Redefined default RAM segment
;*******************************************************************************
MEMORY RAM:(0FF700H, 00720H)
MEMORY RAM_SADDR:(0FFE20H, 000C0H)

;*******************************************************************************
; EEL_CODE and FDL_CODE segments are located inside ROM
;*******************************************************************************
MERGE FDL_CODE:=ROM
MERGE EEL_CODE:=ROM

;*******************************************************************************
; EEL_SDAT and FDL_SDAT segment is located inside saddr RAM
;*******************************************************************************
MERGE FDL_SDAT:=RAM_SADDR
MERGE EEL_SDAT:=RAM_SADDR


;*******************************************************************************
; EEL_CNST and FDL_CNST segments are located inside ROM
;*******************************************************************************
MERGE FDL_CNST:=ROM
MERGE EEL_CNST:=ROM
```

## 3.1 在指定的 RAM 地址创建一个新的段

使用#pragma 指令，可以修改段的名字并指定新段的起始地址。

以下是语法:

```
#pragma section compiler-output-section-name    new-section-name    [AT startaddress]
```

下面的例子将数据段的名字改为 DAT1.

```
#pragma section @@DATA DAT1
  unsigned char data1[100];
  unsigned char data2[100];
  unsigned char data3[100];
```

从 map 文件中可以看到，DAT1 段被创建在 RAM 区

```
340                         @@INIT    r_systeminit
341                                   FF7F0H
342                         @@INIT    r_cg_cgc    FF7F0H
343                         @@INIT    r_cg_cgc_user
344                                   FF7F0H
345                         @@INIT    EEL_type2
346                                   FF7F0H
347                         @@INIT    FDL_DESCR
348                                   FF7F0H    00000H
349                         @@INIT    EEL_DESCRIPTOR
350                                   FF7F0H    00000H
351                         @@INIT    data        FF7F0H    00000H
352                         @@INIT    @rom        FF7F0H    00000H
353                   DAT1                      FF7F0H    0012CH  DSEG BASEP
354                         DAT1      data        FF7F0H    0012CH
355                   @@INIS                    FF91CH    00000H  DSEG UNITP
356                         @@INIS    @cstart     FF91CH    00000H
357                         @@INIS    r_main      FF91CH    00000H
358                         @@INIS    r_systeminit
359                                   FF91CH    00000H
360                         @@INIS    r_cg_cgc    FF91CH    00000H
```

DAT1 段被创建.

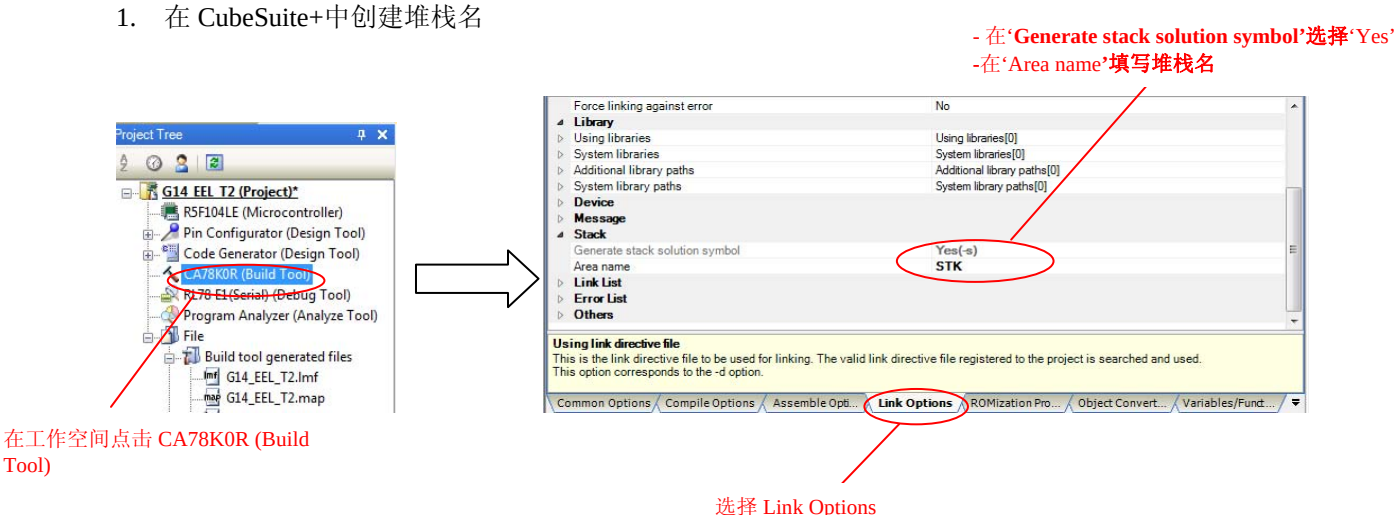
大小是 300 字节

## 3.2 修改堆栈区的地址和大小

堆栈区可以在 CubeSuite+ 环境和链接指令文件中修改。

以下流程演示如何修改堆栈区:

1. 在 CubeSuite+中创建堆栈名

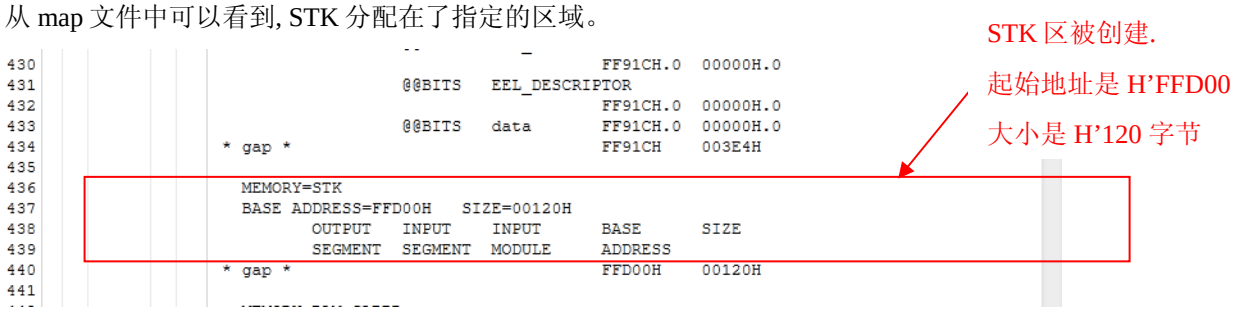



2. 在链接指令文件(.dr) 中定义堆栈区的地址和大小

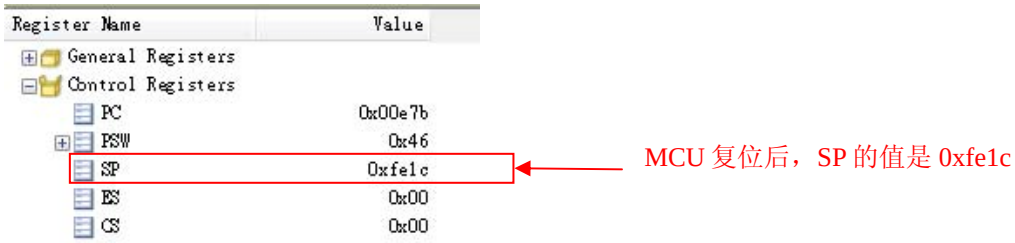
例如:

```
MEMORY STK : ( 0FFD00H, 120H )
```

从 map 文件中可以看到, STK 分配在了指定的区域。




MCU 复位后，堆栈指针指向指定的区域。



## 3.3 链接指令文件样例

```
;*****************************************************************************
; Library       : EEPROM Emulation Library (T02)
;
; File Name     : $Source: eel_sample_linker_file.dr $
; Lib. Version  : $RL78_EEL_LIB_VERSION_T02: V1.00 $
; Mod. Revision : $Revision: 1.10 $
; Mod. Date     : $Date: 2012/10/25 18:00:04MESZ $
; Device(s)     : RL78/G13 (R5F100LE)
; Description   : Linker sample file, please modify according to your device
;*****************************************************************************
; DISCLAIMER
; This software is supplied by Renesas Electronics Corporation and is only
; intended for use with Renesas products. No other uses are authorized. This
; software is owned by Renesas Electronics Corporation and is protected under
; all applicable laws, including copyright laws.
; THIS SOFTWARE IS PROVIDED "AS IS" AND RENESAS MAKES NO WARRANTIES REGARDING
; THIS SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING BUT NOT
; LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE
; AND NON-INFRINGEMENT. ALL SUCH WARRANTIES ARE EXPRESSLY DISCLAIMED.
; TO THE MAXIMUM EXTENT PERMITTED NOT PROHIBITED BY LAW, NEITHER RENESAS
; ELECTRONICS CORPORATION NOR ANY OF ITS AFFILIATED COMPANIES SHALL BE LIABLE
; FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES FOR
; ANY REASON RELATED TO THIS SOFTWARE, EVEN IF RENESAS OR ITS AFFILIATES HAVE
; BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
; Renesas reserves the right, without notice, to make changes to this software
; and to discontinue the availability of this software. By using this software,
; you agree to the additional terms and conditions found by accessing the
; following link:
; http://www.renesas.com/disclaimer
;
; Copyright (C) 2012 Renesas Electronics Corporation. All rights reserved.
;*****************************************************************************


;*****************************************************************************
; Redefined default RAM segment
;*****************************************************************************
MEMORY RAM:(0FF700H, 00600H)
MEMORY RAM_SADDR:(0FFE20H, 000C0H)
MEMORY STK : ( 0FFD00H, 120H )

;*****************************************************************************
; EEL_CODE and FDL_CODE segments are located inside ROM
;*****************************************************************************
MERGE FDL_CODE:=ROM
MERGE EEL_CODE:=ROM

;*****************************************************************************
; EEL_SDAT and FDL_SDAT segment is located inside saddr RAM
;*****************************************************************************
MERGE FDL_SDAT:=RAM_SADDR
MERGE EEL_SDAT:=RAM_SADDR


;*****************************************************************************
; EEL_CNST and FDL_CNST segments are located inside ROM
;*****************************************************************************
MERGE FDL_CNST:=ROM
MERGE EEL_CNST:=ROM
```

## 公司主页和咨询窗口

瑞萨电子主页
http://www.renesas.com/

数据闪存库
http://www.renesas.com/products/tools/flash_prom_programming/flash_libraries/data_flash_lib/index.jsp
http://www.renesas.com/products/tools/flash_prom_programming/flash_libraries/data_flash_lib/downloads.jsp

咨询
http://www.renesas.com/contact/

| 修订记录 | RL78 族, R7F0C 系列 应用说明 |
|---|---|

| Rev | 发行日 | 修订内容 | |
|---|---|---|---|
| | | 页 | 修订处 |
| 0.05 | 2013.08.09 | — | 内部版本 |
| 0.09 | 2013.08.16 | — | 初版 |
| 1.00 | 2013.11.05 | — | 初版发行 |

# 产品使用时的注意事项

本文对适用于单片机所有产品的“使用时的注意事项”进行说明。有关个别的使用时的注意事项请参照正文。此外，如果在记载上有与本手册的正文有差异之处，请以正文为准。

1. 未使用的引脚的处理

【注意】将未使用的引脚按照正文的“未使用引脚的处理”进行处理。

CMOS产品的输入引脚的阻抗一般为高阻抗。如果在开路的状态下运行未使用的引脚，由于感应现象，外加LSI周围的噪声，在LSI内部产生穿透电流，有可能被误认为是输入信号而引起误动作。

未使用的引脚，请按照正文的“未使用引脚的处理”中的指示进行处理。

2. 通电时的处理

【注意】通电时产品处于不定状态。

通电时， LSI内部电路处于不确定状态，寄存器的设定和各引脚的状态不定。通过外部复位引脚对产品进行复位时，从通电到复位有效之前的期间，不能保证引脚的状态。

同样，使用内部上电复位功能对产品进行复位时，从通电到达到复位产生的一定电压的期间，不能保证引脚的状态。

3. 禁止存取保留地址（保留区）

【注意】禁止存取保留地址（保留区）

在地址区域中，有被分配将来用作功能扩展的保留地址（保留区）。因为无法保证存取这些地址时的运行，所以不能对保留地址（保留区）进行存取。

4. 关于时钟

【注意】复位时，请在时钟稳定后解除复位。

在程序运行中切换时钟时，请在要切换成的时钟稳定之后进行。复位时，在通过使用外部振荡器（或者外部振荡电路）的时钟开始运行的系统中，必须在时钟充分稳定后解除复位。另外，在程序运行中，切换成使用外部振荡器（或者外部振荡电路）的时钟时，在要切换成的时钟充分稳定后再进行切换。

5. 关于产品间的差异

【注意】在变更不同型号的产品时，请对每一个产品型号进行系统评价测试。

即使是同一个群的单片机，如果产品型号不同，由于内部ROM、版本模式等不同，在电特性范围内有时特性值、动作容限、噪声耐量、噪声辐射量等也不同。因此，在变更不认同型号的产品时，请对每一个型号的产品进行系统评价测试。

# Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

RENESAS

**SALES OFFICES** Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.


**Renesas Electronics America Inc.**
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

**Renesas Electronics Canada Limited**
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

**Renesas Electronics Europe Limited**
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700, Fax: +44-1628-651-804

**Renesas Electronics Europe GmbH**
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

**Renesas Electronics (China) Co., Ltd.**
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

**Renesas Electronics (Shanghai) Co., Ltd.**
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

**Renesas Electronics Hong Kong Limited**
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

**Renesas Electronics Taiwan Co., Ltd.**
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

**Renesas Electronics Singapore Pte. Ltd.**
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

**Renesas Electronics Malaysia Sdn.Bhd.**
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

**Renesas Electronics Korea Co., Ltd.**
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141



Colophon 2.2