RL78 Family and R7F0C Series

Data Flash Libraries Comparison & Application

R01AN1944EC0100
Rev.1.00
8 Nov, 2013

Introduction

For Data Flash access, there are different types of Data Flash Access Libraries available on the website.

This application note intends to compare the different types of Data Flash Library and EEPROM Emulation Library, in terms of Resource and API functions. By providing sample source code example, user able to evaluate the libraries right way.

The aim of this application note is to provide more information to user, in order to select the appropriate Data Flash Access Library.

Target Device

R5F104xA, R5F104xC

R7F0C001, R7F0C002, R7F0C010

Contents

1. DATA FLASH LIBRARIES – COMPARISON AND APPLICATION 2
1.1 Resource Comparison 2
1.2 Memory Map 3
1.3 Functions and API 3
1.4 Program example 4
1.5 Comparison and Usage note 10
1.6 Maximum Function Execution Times Comparison 10

2. EEPROM EMULATION LIBRARIES – COMPARISON AND APPLICATION 11
2.1 Resource Comparison 11
2.2 Memory Map 11
2.3 Calculation of ROM occupation 12
2.4 EEL Functions and API 13
2.5 Program example 14
2.6 Comparison and Usage Note 19
2.7 Calculation of Rewrite times (Reference) 19

3. LINK DIRECTIVE FILE SETUP 20
3.1 Create a new section at a specified RAM address 21
3.2 Modify the address and size of stack area 22
3.3 Link Directive File Sample 23

WEBSITE AND SUPPORT 24

1. Data Flash Libraries – Comparison and Application

For Data Flash Access for RL78, there are 3 different types of Data Flash Libraries (FDL),

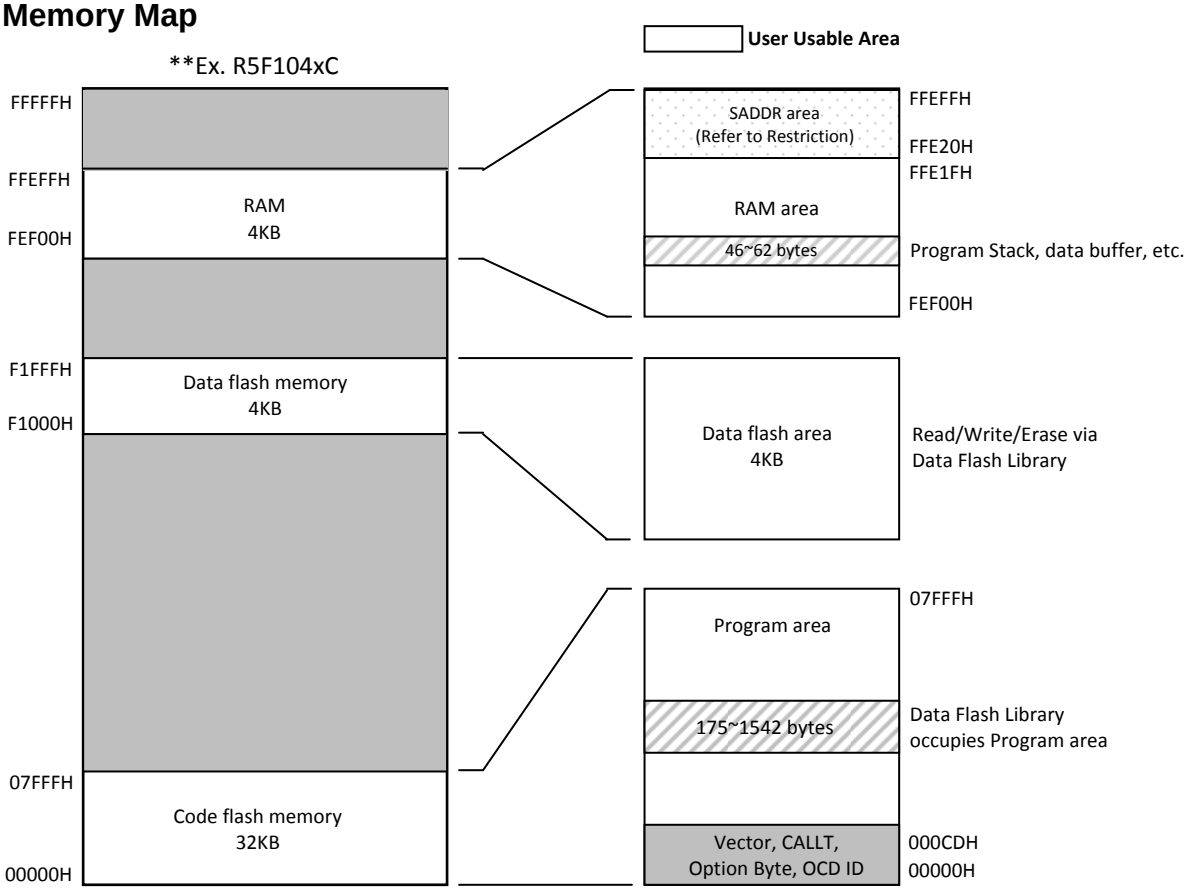
- Data Flash Access Library, Type T01
- Data Flash Access Library, Type T02 (Tiny)
- Data Flash Access Library, Type T04 (Pico)

Below are the summaries of 3 different types of Library, in term of Resource and Function.

1.1 Resource Comparison

<table>
<tr><th colspan="2">Resources</th><th>T01</th><th>T02 (Tiny)</th><th>T04 (Pico)</th><th>Description / Remarks</th></tr>
<tr><td colspan="2">Max code size
Code + Constant</td><td>1478 + 64
(1542 bytes)</td><td>572 +10
(682 bytes)</td><td>175 + 0
(175 bytes)</td><td>Allocated in Program Flash area</td></tr>
<tr><td rowspan="6">Working RAM</td><td>2.5KB RAM
R5F104xA (G14)</td><td colspan="3">No restriction</td><td rowspan="6">Working RAM is the RAM area of MCU, which is used by Data Flash Access Library. Depend on devices, each has restrictions when PFDL is used.
(For restriction, please refer to device user manual for details)</td></tr>
<tr><td>4KB RAM
R5F104xC (G14)</td><td colspan="3">No restriction</td></tr>
<tr><td>1KB RAM
R7F0C001</td><td colspan="2">FFB00H~FFC89 (395 Bytes)</td><td>No restriction</td></tr>
<tr><td>1.5KB RAM
R7F0C002</td><td colspan="2">FF900H~FFC80 (897 Bytes)</td><td>FF900H~FF9FFH
(256 Bytes)</td></tr>
<tr><td>1.5KB RAM
R7F0C010</td><td colspan="2">FF900H~FFC80 (897 Bytes)</td><td>FF900H~FF9FFH
(256 Bytes)</td></tr>
<tr><td>Other than above</td><td colspan="3">Refer to Hardware User’s Manual for detail</td></tr>
<tr><td colspan="2">Internal data
(SADDR RAM)</td><td>2 bytes</td><td>2 bytes</td><td>FFE20H~FFEFFH
(224 Bytes)</td><td>SADDR RAM (FFE20H~FFEFFH) is immediate data access area allocated in RAM.</td></tr>
<tr><td colspan="2">Max. Stack consumption</td><td>60 bytes</td><td>56 bytes</td><td>46 bytes</td><td>Other than SelfRAM, SADDR RAM</td></tr>
<tr><td colspan="2">SADDR area Restriction</td><td colspan="3">Stack, Data Buffer, Function Arguments cannot be allocated in this area</td><td>Address FFE20H~FFEFFH</td></tr>
</table>

1.2 Memory Map



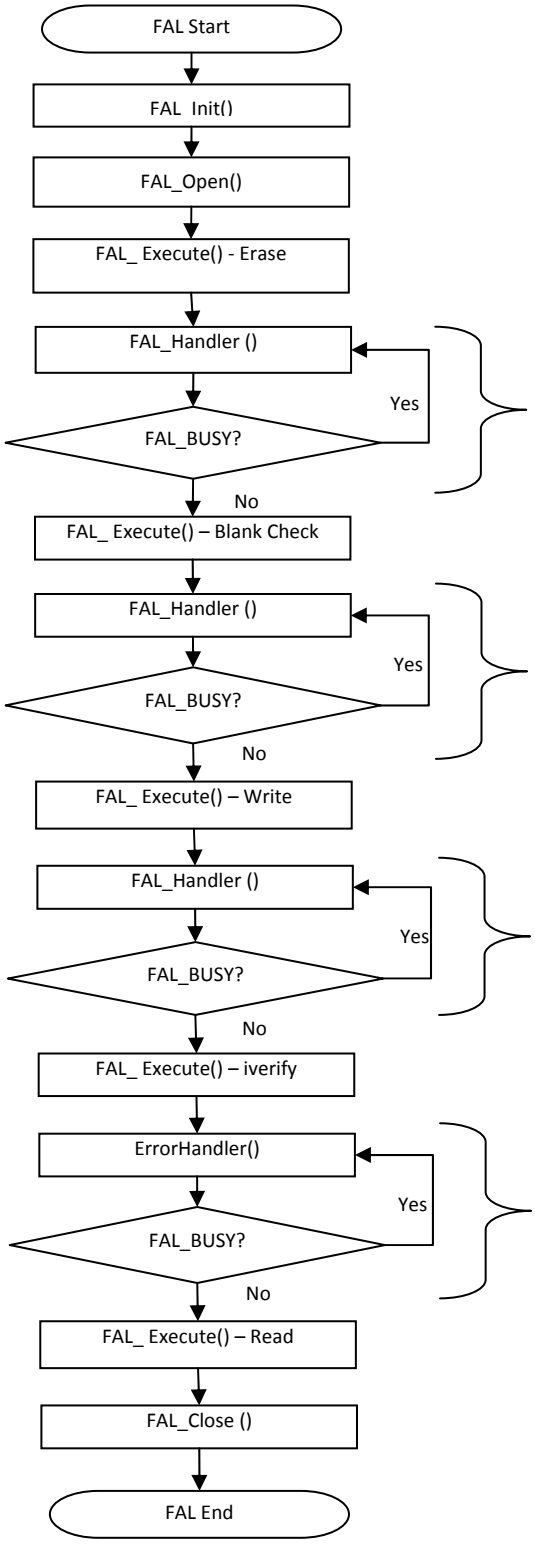
**Note: Memory map may vary for different device. Details please refer to Device User's Manual.

1.3 Functions and API

Function \ Library API		T01	T02 (Tiny)	T04 (Pico)	Description / Remarks
Init		FAL_Init	FDL_Init		Initialization of all internal data with given configuration parameter
Open		FAL_Open	FDL_Open	PFDL_Open	Initialization of the RAM used by the PFDL, enabling of the Data Flash clock
Close		FAL_Close	FDL_Close	PFDL_Close	Ending the operation of the PFD, disabling of the Data Flash clock
Execute (Command)	Erase	FAL_Execute	FDL_Execute	PFDL_Execute	Triggering and execution of commands on the Data Flash memory
	BlankCheck				
	Write				
	IVerify				
	Read				
Handler		FAL_Handler	FDL_Handler	PFDL_Handler	Checking of the current status of a running Data Flash operation and driving the command forward (status check processing)
Abort			FDL_Abort		Erase command abortion
StandBy			FDL_StandBy		Drive the library into StandBy mode (Data Flash H/W switch off)
Wakeup			FDL_WakeUp		Wake up the library from StandBy mode
GetVersionString		FAL_GetVersionString	FDL_GetVersionString	PFDL_GetVersionString	Acquisition of the version information of the PFDL

1.4 Program example

1.4.1 Program Flow Chart



****Program flow is same for T01/T02/T04. Flow chart uses T01 API as example.**

- T01 API start with FAL_xxx()
- T02 API start with FDL_xxx()
- T04 API start with PFDL_xxx()

For detail explanation of API usage, please refer to Data Flash Library User Manual.

Wait for command execution complete

Execute status is shown in

- T01/T02: request.status_enu
- T04: return value of PFDL_Handler()

Wait for command execution complete

Execute status is shown in

- T01/T02: request.status_enu
- T04: return value of PFDL_Handler()

Wait for command execution complete

Execute status is shown in

- T01/T02: request.status_enu
- T04: return value of PFDL_Handler()

Wait for command execution complete

Execute status is shown in

- T01/T02: request.status_enu
- T04: return value of PFDL_Handler()

1.4.2 Data Flash Access Library Type 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 Data Flash Access Library Type 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 Data Flash Access Library Type 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 Comparison and Usage note

	T01	T02 (Tiny)	T04 (Pico)
Minimum data size for read/write	4 Bytes	1 Bytes	1 Bytes
Maximum data size for read/write	4 Bytes	4k Bytes	4k Bytes
Address alignment	4 Bytes	1 Bytes	1 Bytes
Number of APIs	6	9	5
EEL support	Yes	Yes	No
Command execution status	status_enu in command request	status_enu in command request	Return value of PFDL_Execute()

1.6 Maximum Function Execution Times Comparison

Function (API)		T01	T02 (Tiny)	T04 (Pico)
Init		No command running: 1758/fclk Command running in background: 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 BYTE_CT)	53/fclk+(17/fclk x BYTE_CT)
Handler		974/fclk + 15us	284/fclk + 15us	251/fclk + 14us
Open		83/fclk + 12us	27/fclk + 14us	536/fclk
Close		No command running: 42/fclk Command running in background: 388/fclk + 60us	No command running: 30/fclk Command running in background: 836/fclk + 444us	823/fclk +443us
Standby			305/fclk	
WakeUp			32/fclk	
Abort			350/fclk	
GetVersionString		14/fclk	14/fclk	10/fclk

2. EEPROM Emulation Libraries – Comparison and Application

On top of Data Flash Library, users able to use similar way to access the data flash area as external EEPROM with EEPROM Emulation Libraries. By using these libraries, the management of data flash writes and erase can be omitted.

For Data Flash Access for RL78, there are 2 different types of EEPROM Emulation Libraries,

- EEPROM Emulation Library, Type T01
- EEPROM Emulation Library, Type T02 (Tiny)

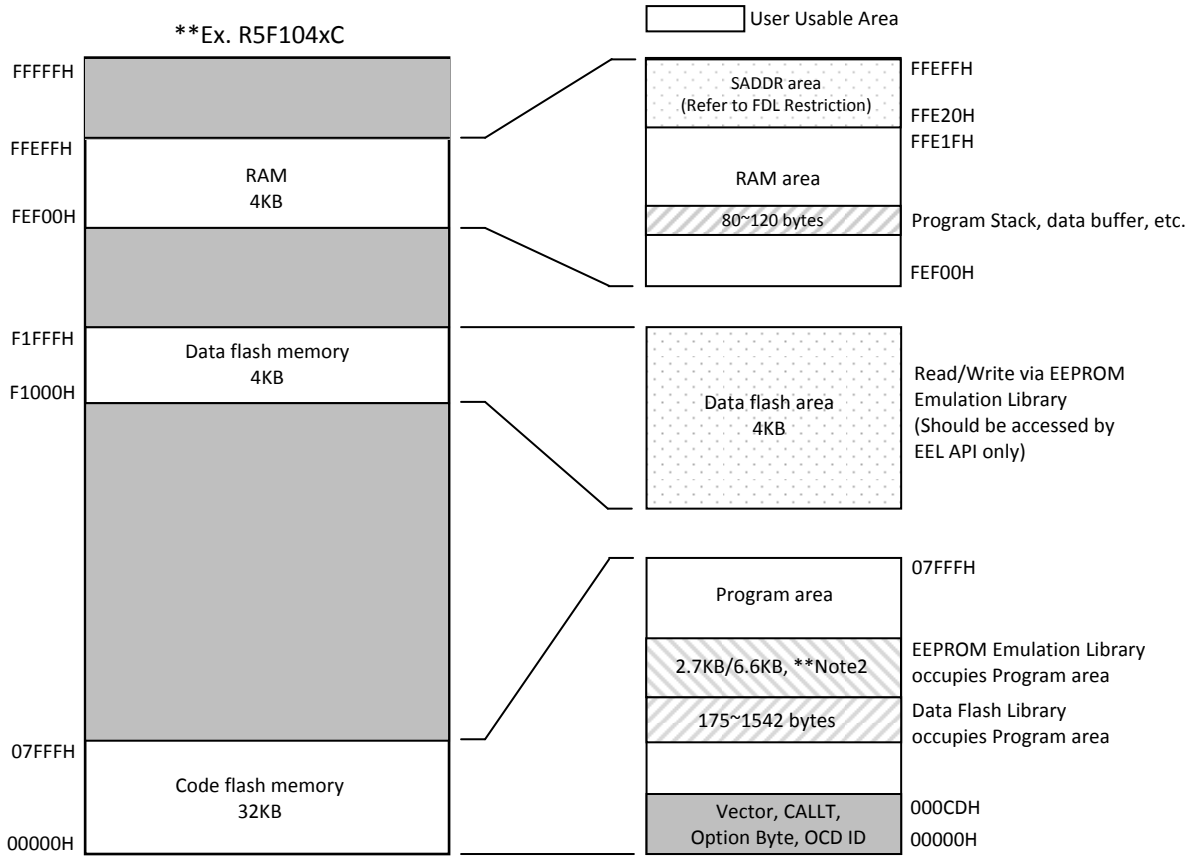
Remark: EEPROM emulation library does not have Type T04.

Below are the summaries of 2 different types of Library, in term of Resource and Function.

2.1 Resource Comparison

Resources	T01	T02 (Tiny)	Description / Remarks
Max code size Code + Constant	6.6k bytes + (4 +(number of EEL variable)*4)	2746 bytes + (4 + EEL variable count)	Allocated in Program Flash area
Minimum Data Flash	4 Block	2 Block	Minimum Data Flash size required (1KB/Block)
RAM	1 byte	-	Working RAM is the RAM area of MCU
Internal data (SADDR RAM)	9 bytes	1 bytes	SADDR RAM (FFE20H~FFF1FH) is immediate data access area allocated in RAM.
Max. Stack consumption (including FDL)	<120 bytes	80 bytes	Other than SelfRAM, SADDR RAM
FDL required	FDL T01	FDL T02 (Tiny)	Required Data Flash Library

2.2 Memory Map



** Note 1: Memory map may vary for different device. Details please refer to Device User's Manual.
Note 2: Size varies depends on total of variables. ROM size calculation is shown in next chapter.

2.3 Calculation of ROM occupation

ROM occupation for EEPROM emulation is combined from FDL, EEL and EEL descriptors. The size of FDL and EEL are fixed, but EEL descriptors are depends on application. This chapter shows how to calculate the ROM size (base on Program example).

2.3.1 EEL Descriptor size calculation

The size of the EEL descriptor depends on the number of variable defined in eel_descriptor.c, but the descriptor size for EEL T01 and T02 are different. In the examples below, the number of variable for EEL library T01 and T02 are 8.

(1) **EEL T01 Descriptors**

```
__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,  \
};
```

The size for the above EEL T01 descriptor: (number of variable + 1) x 4 = 36 Bytes.

(2) **EEL T02 Descriptors**

```
__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  */  \
}
```

The size for the above EEL T02 descriptor: (number of variable + 2) = 10 Bytes.

2.3.2 ROM Size for EEPROM Emulation

Calculation based on Section 2.3.1, 8 EEL descriptors are used in program example. Actual size may vary due to actual use of descriptors.

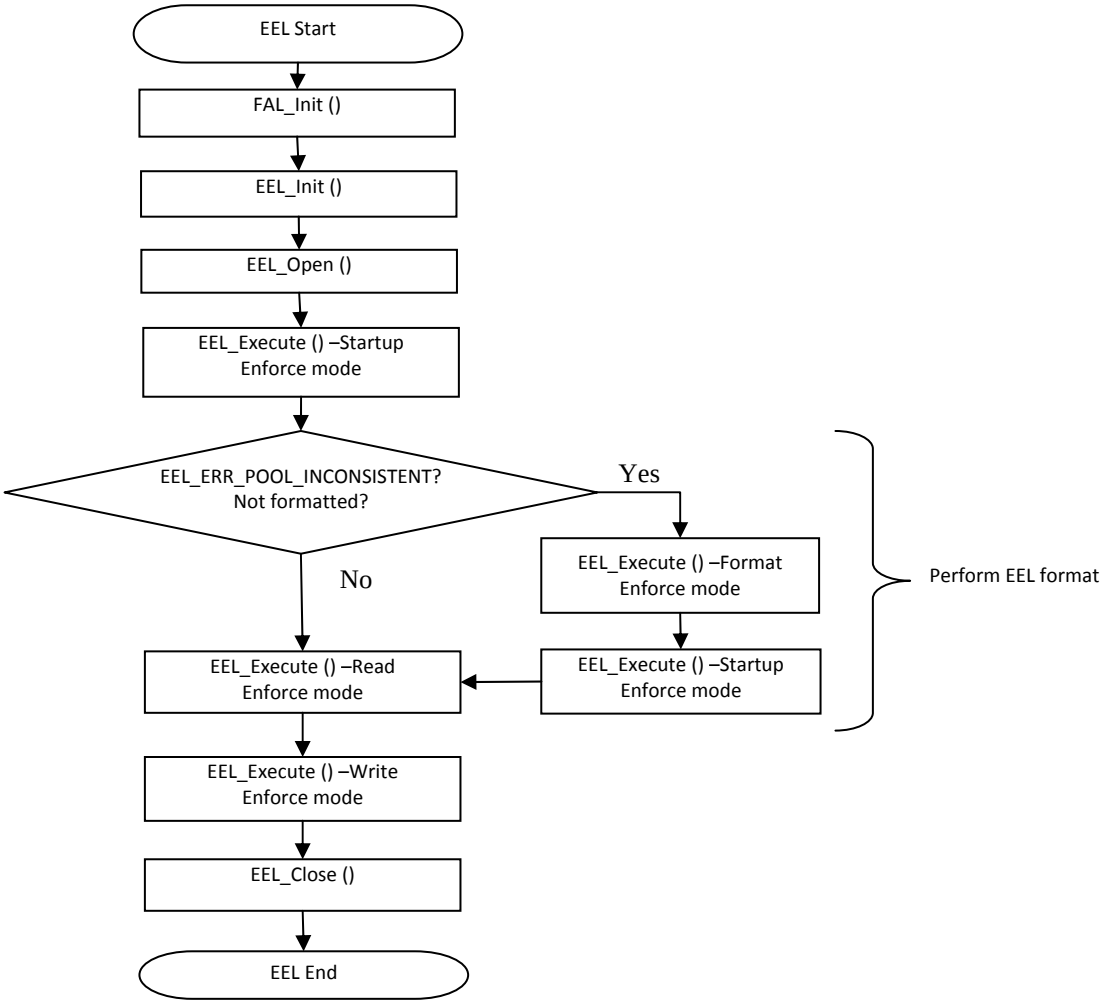
Library \ Resources	FDL	EEL	Descriptors (8 Variables)	Total ROM size
T01	1542 bytes	6.6KB	36 bytes	**8178 bytes**
T02 (Tiny)	682 bytes	2746 bytes	10 bytes	**3438 bytes**

2.4 EEL Functions and API

Function \ API	T01		T02 (Tiny)		Description / Remarks
Init	EEL_Init		EEL_Init		Initialization of all internal data with given configuration parameter
Open	EEL_Open		EEL_Open		Open EEL logically for usage
Close	EEL_Close		EEL_Close		Close EEL logically
Execute (Command)	EEL_Execute	Startup	EEL_Execute	Startup	Triggering and execution of commands
		Shutdown		Shutdown	
		Clean Up			
				Verify	
		Format		Format	
				Refresh	
		Write		Write	
		Read		Read	
Handler	EEL_Handler		EEL_Handler		Checking of the current status and driving the command forward (status check processing)
TimeOut Count Down	EEL_TimerOut_CountDown				Start Count down timer
Get Driver Status	EEL_GetDriverStatus		EEL_GetDriverStatus		Check the internal status of the EEL driver before placing a request
Get Space	EEL_GetSpace		EEL_GetSpace		Check the remaining free space in the current active block
Get Version String	EEL_GetVersionString		EEL_GetVersionString		Acquisition of the version information of EEL

2.5 Program example

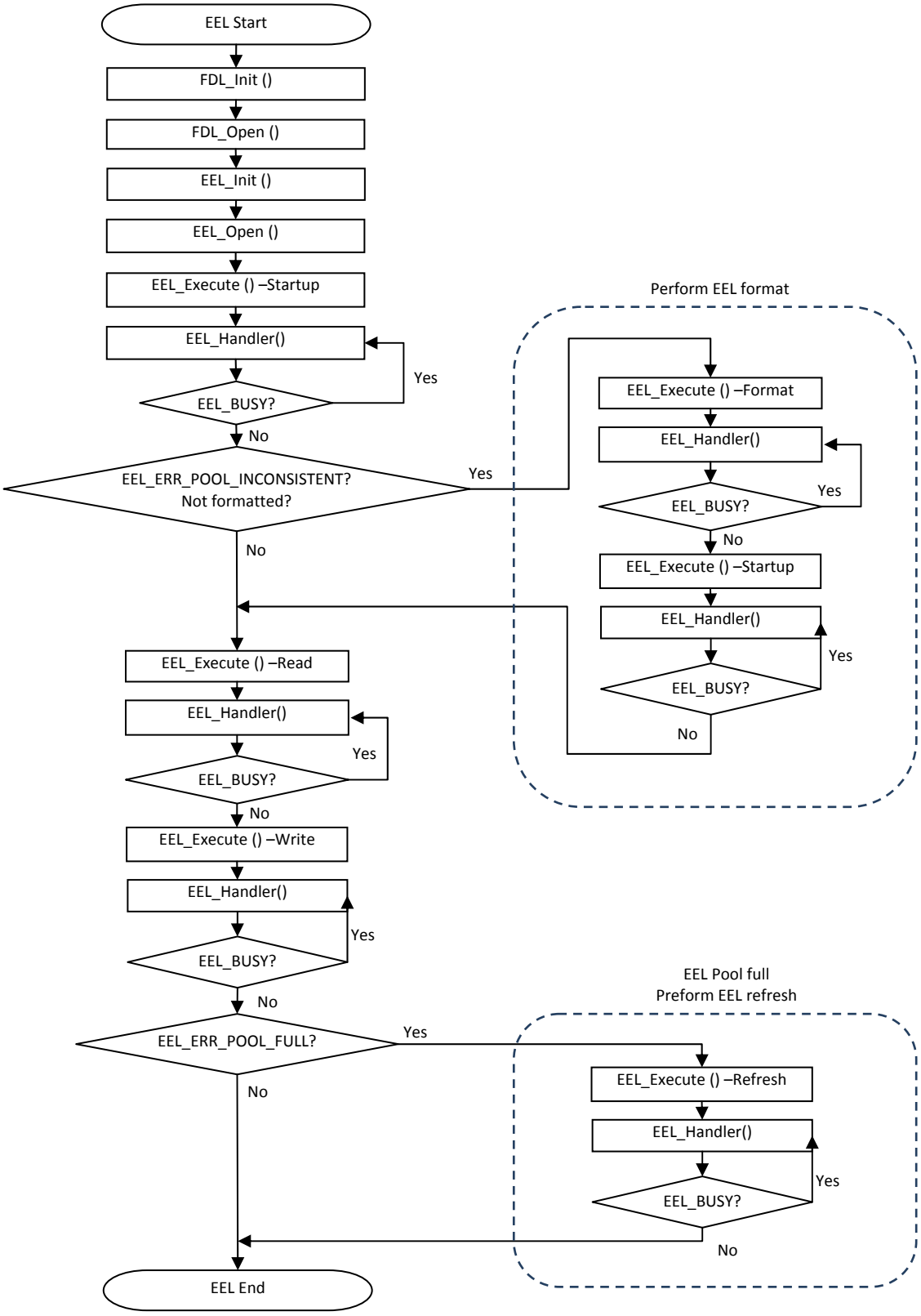
2.5.1 Flowchart of EEPROM Emulation Library T01 code example



2.5.2 EEPROM Emulation Library Type 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 Flowchart of EEPROM Emulation Library T02 code example



2.5.4 EEPROM Emulation Library Type 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 Comparison and Usage Note

		T01	T02 (Tiny)
Minimum data structure size		4 Bytes	1 Bytes
Maximum data structure size		255 Bytes	255 Bytes
Number of APIs		9	8
Execution mode	Polling mode	Support	Support
	Timeout mode	Support	
	Enforcing mode	Support	
Refresh operation (Refer to EEL user manual)		Process in EEL library	Process in user program

2.7 Calculation of Rewrite times (Reference)

Calculation based on writing 16-bytes variable into whole 4K data flash. Rewrite time count as 1 erase + 1 write after the erase is regarded as 1 rewrite. In EEL, library is automatically performing erase by refresh process.

For RL78, G13/G14, the minimum rewrite times for data retained (each rewrite) for 5 years is 100,000 times.

In EEL T01 case:

It can be obverses 16-bytes variable has been written to 160 times before refresh process. The minimum rewrite times:

Min rewrite = 100,000 x 160 = 16,000,000 times.

In EEL T01 case:

It can be obverses 16-bytes variable has been written to 220 times before refresh process. The minimum rewrite times:

Min. rewrite = 100,000 x 220 = 22,000,000 times.

3. Link Directive File Setup

Memory directive

The memory directive is a directive to define a memory area (the address of memory to implement and its name).

The defined memory area can be referenced by the segment location directive using this name (memory area name).

Up to 100 memory areas can be defined, including the memory area defined by default. The syntax is shown below.

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

Segment location directive

The segment location directive is a directive which places a specified segment in a specified memory area or places it at specific address. The syntax is shown below.

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

The following is an example of link directive for EEL T02.p

- RAM area is defined as H'FF700~H'FFE1F. User program will access this RAM area.
- RAM_SADDR is defined as H'FFE20~H'FFEFF.
- Section EEL_SDAT and FDL_SDAT are allocated inside the RAM_SADDR area.
- Section EEL_CODE and FDL_CODE are allocated inside the ROM area.
- Section EEL_CNST and FDL_CNST are allocated inside the ROM area.

```
;*******************************************************************************
; 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 Create a new section at a specified RAM address

By using the #pragma directive, the name of the section can be changed, and the start address of the new section can be specified.

The syntax is shown below:

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

The following example is to rename the name of the data section to DAT1,.

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

From the map file, the DAT1 section is created in the RAM area

```
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
```

Section DAT1 is created.

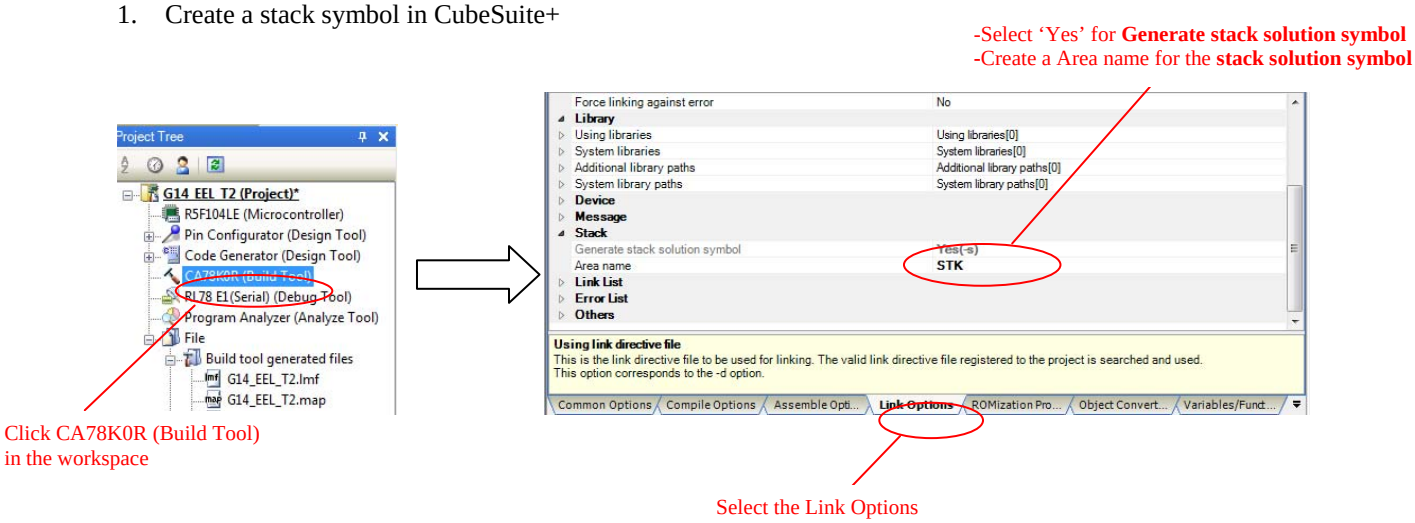
Size is 300 bytes

3.2 Modify the address and size of stack area

The stack area can be modified in CubeSuite+ environment and link directive file.

The procedure below shows how to modify the stack area:

1. Create a stack symbol in CubeSuite+

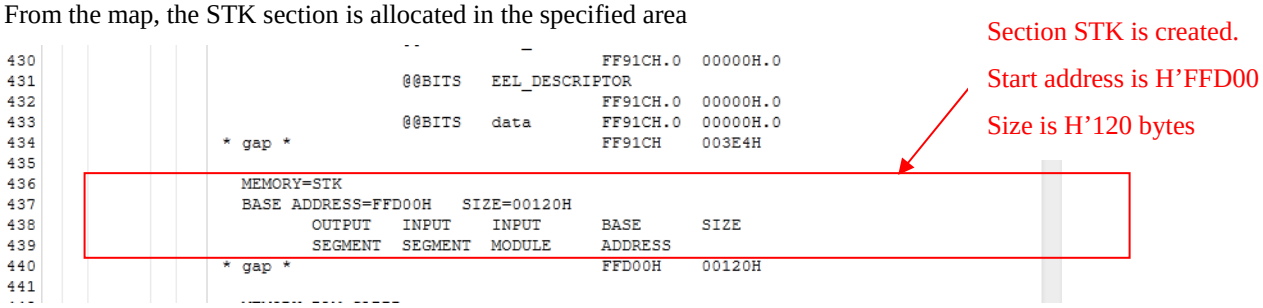


2. Define the address and size for stack solution symbol in link directive (.dr) file

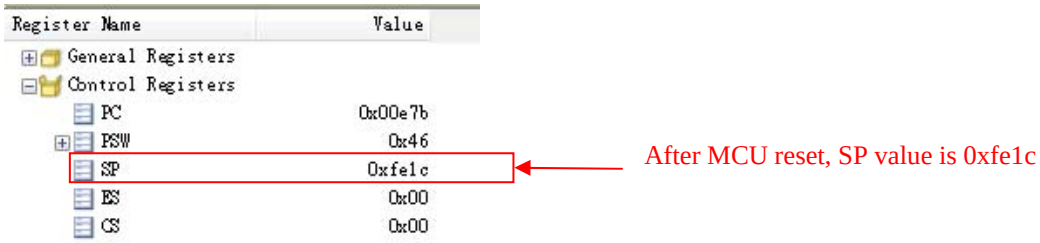
   Example:

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

From the map, the STK section is allocated in the specified area



After MCU reset, the stack pointer is started at the specified area.



3.3 Link Directive File Sample

```
;*******************************************************************************
; 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
```

Website and Support

Renesas Electronics Website
http://www.renesas.com/

Data Flash Libraries
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

Inquiries
http://www.renesas.com/contact/

Revision Record

		Description	
Rev.	**Date**	**Page**	**Summary**
0.01	1 Aug 2013		
0.05	9 Aug 2013		Internal Version
0.09	16 Aug 2013		Preliminary
1.00	8 Nov 2013		Release

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins
   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.
   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on
   The state of the product is undefined at the moment when power is supplied.
   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.
5. Differences between Products
   Before changing from one product to another, i.e. to one with a different type number, confirm that the change will not lead to problems.
   — The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

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