Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

THIS SPEC IS OBSOLETE

Spec No: 002-00249

Spec Title: AN200249 - FM MCU EEPROM EMULATION LIBRARY

Replaced by: None

AN200249

FM MCU EEPROM Emulation Library

Author: Jane Li

Associated Part Family: FM3 and FM4

To get the latest version of this application note, please visit:

http://www.cypress.com/documentation/application-notes/an200249-fm-mcu-emulation-eeprom-library

AN200249 describes an EEPROM Emulation Library for use with the Cypress FM families of MCU devices. It includes information on flash memory and library functions and serves as a guide to the library.

Contents

1 Introduction

An emulated EEPROM uses MCU internal flash to emulate an EEPROM to save data, and it cannot lose data when the MCU is powered off. This EEPROM supports the 0 ~ 2-KB data range. You can define the EEPROM size during initialization.

Different MCUs have different flash structures, so there are classes of EEPROM for different MCUs. Following are the EEPROM Emulation Library classes:

- Library Type A: FM3 MB9A310K/110K series MCU (FM3 MCU Type 5)
- Library Type B: FM3 MB9AFB40N series MCU (FM3 MCU Type 6)
- Library Type C: FM3 MB9BF520T series MCU (FM3 MCU Type 12)
- Library Type D: FM4 S6E2H and MB9BF560R series MCU (FM4 MCU Type 1 and 6)

2 System Hardware Environment

- CPU chip: Cypress Series FM3 and FM4 MCUs
- Minimum instruction time: 25 ns (write one byte time)
- RAM occupied space: 2.82 KB
- Code occupied space: 6.612 KB

3 Development Environment

Table 1 summarizes the development environment for the MCUs.

Table 1. MCU Development Environment

Name	Description	Part Number	Manufacturer
Embedded Workbench for Arm	Software integrated development environment (IDE)	Type A version: 6.21.1.284 Type B version: 6.60.1.5104 Type C version: 6.70.3.6387 Type D version: 7.10.3.6932	IAR
J-Link	Emulation tool		SEGGER

Note: Only Type D has submitted the Keil library to the web. If you want to use the related library in Keil, contact Cypress Support.

4 System Functions

This EEPROM system includes several user functions, as outlined in this section.

4.1 Macro Definitions

4.1.1 API Function Operation Result

These macros indicate the result of API functions, as summarized in Table 2.

Table 2. API Operational Status

Name	Description	Value
OK	The API function was successful.	0
NG	An API function error has occurred.	1
BUSY	A higher priority API function is operating.	2
PARA_ERROR	The value of the function parameter is not valid.	4
SYSTBUSY	The API function is a consistent call that may influence back erase.	5

You should take the following the actions when API functions return these macros:

- OK – Operation successful; no further action required.
- NG – Call the EEPROM_Check () API function. Depending on the check result, call the related API function and then call the original API function again.
- BUSY – The device is busy. Call the API function later.
- PARA_ERROR – The EEPROM address or size provided is incorrect; provide the correct value for the parameter.
- SYSTBUSY – An erase operation started earlier is currently in operation. Wait for at least 10 ms and then call the API function again.

Note: Only the write and read API function calls return the SYSTBUSY status.

4.1.2 EEPROM Check Result

The check function checks the EEPROM status and returns the status of the EEPROM and the required action, as listed in Table 3.

Table 3. Results of Check Function

Name	Description	Value
CONSISTENT	EEPROM is ready.	0
DEFINE	EEPROM needs to be defined.	1
REDEFINE	EEPROM needs to be redefined.	2
RESTORE	EEPROM status needs to be restored.	3

The following list describes operations relevant to the results:

- CONSISTENT – EEPROM is OK: User can perform the required operation.
- DEFINE – Call EEPROM_Define (): Define the EEPROM.
- REDEFINE – Call EEPROM_Define (): Redefine the EEPROM.
- RESTORE – Call EEPROM_Restore (): Restore the EEPROM status.

For details on the use of this API, see Figure 15.

4.1.3 EEPROM Size

Table 4 lists the parameters that are required for the EEPROM size API.

Table 4. EEPROM Size

Name	Description	Value
E2P_128B	EEPROM size is 128 bytes.	0x08
E2P_256B	EEPROM size is 256 bytes.	0x10
E2P_512B	EEPROM size is 512 bytes.	0x20
E2P_1024B	EEPROM size is 1024 bytes.	0x40
E2P_2048B	EEPROM size is 2048 bytes.	0x80

Following are the address ranges that can be used for different sizes:

- 128 bytes: 0 ~ 127
- 256 bytes: 0 ~ 255
- 512 bytes: 0 ~ 511
- 1024 bytes: 0 ~ 1023
- 2048 bytes: 0 ~ 2047

Note: When the EEPROM size is defined, you are limited by this defined size. Do not use a value that is beyond the defined value; otherwise, the API returns a PARA_ERROR response.

4.2 User Functions

Table 5 lists the functions that are available for use.

Table 5. User Functions

Prototype	Description	Remark
uint8_t EEPROM_Define(uint8_t ucSize)	Initialize the emulated EEPROM in one of the predefined sizes.	NA
uint8_t EEPROM_B_Write(uint16_t WtAdrr, uint8_t WDat)	Write one byte to the emulated EEPROM. When the sector is full, it copies the old data to the new sector and erases the old sector.	NA
uint8_t EEPROM_B_Read(uint16_t RAddr, uint8_t *ucData)	Read one byte from the emulated EEPROM.	NA
uint8_t EEPROM_Check(uint8_t ucSize, uint8_t *ucData)	Check the consistency of data stored in the emulated EEPROM and the information* stored in RAM. The API result informs whether it is blank, consistent, broken but able to be restored, or broken and unable to be restored. This function is intended to be performed every time after power on. Therefore, it reconstructs the EEMS stored in RAM, which is volatile in nature.	Before using libraries, you should call this API first
uint8_t EEPROM_Restore (void)	Restore the EEPROM including the EEMS in RAM. This function is intended to resolve the data inconsistency caused by an incomplete execution of the API.	NA

*To emulate the EEPROM using flash memory, some information needs to be defined in the RAM area, called the “Emulated EEPROM Management Structure (EEMS)” hereafter.

4.3 Function Prototypes

4.3.1 EEPROM_Define ()

This API defines the EEPROM size and status.

Prototype: uint8_t EEPROM_Define(uint8_t ucSize)

Parameter: ucSize: Defined EEPROM size. See Table 4.

Return: Returns the API operational status. See Table 2.

Description: Defines an emulated EEPROM and related size. See Table 4.

Table 6 lists the return values for this API.

Table 6. Return Values of Define

Return Status	Operation	Method
OK	Defined operation was successful.	NA
NG	A flash operation timeout in or the preceding API led to a flash operation timeout.	Call the API again later.
BUSY	Another API is operating.	Call the API later.
PARA_ERROR	The input EEPROM size is not in the define list.	Use an EEPROM size listed in Table 4.

For details on the use of this API, see Figure 15.

4.3.2 EEPROM_B_Write ()

This API writes one byte of data into the EEPROM.

Prototype: uint8_t EEPROM_B_Write(uint16_t WtAdrr, uint8_t WDat).

Parameter: WtAdrr: Writes the EEPROM address (smaller or equal to the defined size).
WDat: The data to be written.

Return: Returns the operational status of the API. See Table 2.

Description: Writes one byte of data into the EEPROM.

Table 7 lists the return values of this API.

Table 7. Return Values of Write

Return Status	Operation	Method
OK	Write operation was successful.	NA
NG	Read error or flash operation timeout occurred in preceding API.	Check again.
BUSY	The check, define, or restore API is operating.	Call API later.
PARA_ERROR	The input EEPROM address is beyond the EEPROM size.	Modify EEPROM address in range.
SYSTBUSY	The sector erase is operating, and there are too many continuous write operations, which may affect the erase operation.	Delay 10 ms and then write.

Figure 1 and Figure 2 depict examples of the write API.

Figure 1. Continuous Write Example

```
for(j=0;j<486;j++)     //486 is a sample, any value that in range is ok
{
    i = EEPROM_B_Write(j,j);
    if(i == SYSTBUSY)
    {
      Delay(200);      //10ms
      i = EEPROM_B_Write(j,j);
    }
}
```

Figure 2. Single Write Example

```
i = EEPROM_B_Write(0x21,0x55);    //write 0x55 to address 0x21
```

Note: The return value “i” indicates the API operational status. For details, refer to Table 2.

4.3.3 EEPROM_B_Read ()

This API reads data from the EEPROM.

Prototype	uint8_t EEPROM_B_Read(uint32_t RAddr,unsigned char *ucData).
Parameter	RAddr: EEPROM address that user wants to read (smaller or equal to defined size). *ucData: Save the read data.
Return	Return API operational status: Refer to Table 2.
Description	Read one byte of data from the EEPROM.

Table 8 lists the return statuses of this API.

Table 8. Return Statuses of Read

Return Status	Operation	Method
OK	Read operation successful.	NA
NG	Read error or flash operation timeout occurred in preceding API.	Check again.
BUSY	The check, define, or restore API is operating.	Call API later.
PARA_ERROR	The input EEPROM address is beyond the EEPROM size.	Modify EEPROM address in range.
SYSTBUSY	The sector erase is operating, and there are too many continuous write operations, which may influence the erase.	Delay 10 ms and then write.

Figure 3 shows an example of the read API.

Figure 3. Read Example

```
unsigned char ReadDat;                  //save read data

i = EEPROM_B_Read(250,&ReadDat);        //250 is sample value
```

Note: The read address must be in the range of the EEPROM size defined by the uint8_t EEPROM_Define (uint8_t ucSize) API.

4.3.4 EEPROM_Check ()

This API checks the EEPROM status to verify the EEPROM is ready or broken.

Prototype	uint8_t EEPROM_Check(uint8_t ucSize, uint8_t *ucData).
Parameter	ucSize: Defined EEPROM size: See Table 4. *ucData: Checks the EEPROM status: See Table 3.
Return	Returns the API operational status. See Table 2.
Description	Verifies the EEPROM status to determine if EEPROM needs to be redefined, restored, and so on.

Table 9 lists the return statuses of this API.

Table 9. Return Statuses of Check

Return Status	Operation	Method
OK	Check operation was successful.	NA
NG	Flash operation timeout occurred.	Check later.
BUSY	System is occupied with another API.	Check later.
PARA_ERROR	Input EEPROM size (ucSize) is not the defined value.	Modify input size; see Table 4.

Note: You need to call the relevant API according to the check result. For details on the use of this API, see Figure 15.

4.3.5 EEPROM_Restore ()

This API restores the EEPROM status in abrupt power-off and other abnormal conditions.

Prototype: uint8_t EEPROM_Restore (void).

Parameter: Void

Return: Returns the API operational status: See Table 2.

Description: Restores the EEPROM status.

Remark: Only the EEPROM EEMS will be restored.

Table 10 lists the return statuses of this API.

Table 10. Return Statuses of Restore

Return Status	Operation	Method
OK	Restore was successful.	NA
NG	Data cannot be restored because of a sector erase timeout or a timeout in the preceding API operation.	Check again.
BUSY	The system is occupied with another API.	Call the API later.

Note: For details on the use of this API, refer to Figure 15.

5 Interrupt Function

5.1 Function List

Prototype	Description	Remark
void BT0_67_IRQHandler(void)	EEPROM timer interrupt function	Only the Type A library uses this function.

5.2 Function Prototype

5.2.1 BT0_67_IRQHandler

This API is called only in BT0_7_IRQHandler ().

Prototype: void BT0_67_IRQHandler(void)

Parameter: Void

Prototype:	void BT0_67_IRQHandler(void)
Return:	Void
Description:	This is the timer interrupt that you should add to the timer interrupt function.
Remark:	Do not disable the base timer interrupt.

For details on the use of this API, see Figure 14.

Note: Only the Type A library uses this interrupt function, not Type B or Type C. Therefore, ignore this interrupt when you are using the Type B and Type C libraries.

5.2.2 BT6_IRQHandler

This API is used in the EEPROM library code. You do not need to call it again.

Prototype:	void BT6_IRQHandler(void)
Parameter:	Void
Return:	Void
Description:	This interrupt is included in library code.
Remark:	Do not disable the base timer interrupt.

Note: Only the Type D library uses this interrupt function, not Type A, Type B, or Type C. Therefore, ignore this interrupt when you are using the Type B and Type C libraries. For the Type A library, refer to BT0_67_IRQHandler.

Note: Please do not close this interrupt when the EEPROM library is using it.

6 Library Outline

6.1 MCU Application

This library can be used on FM MCUs, which include the following features:

- Work flash with sectors (range: TypeA and TypeD-0x200c4000~0x200c5fff: See Figure 4.
  TypeB-0x00204000~0x00207fff: See Figure 5.
  TypeC-0x00514000~0x00517fff: See Figure 6.
- Base timer channel 6 and channel 7 registers: See Figure 7.
- Work/dual flash arithmetic, as shown in Figure 8 and Figure 9.

Figure 4 through Figure 8 show the details of the sectors, timer registers, and flash arithmetic.

Figure 4. Type A and Type D EEPROM Used Sectors

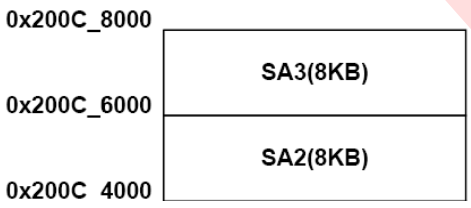


Figure 5. Type B EEPROM Used Sectors

Figure 6. Type C EEPROM Used Sectors

Figure 7. Base Timer Channel 6 and Channel 7 Registers

■ **Timer Control Register (High-order bytes of TMCR)**

bit	15	14	13	12	11	10	9	8
Field	res	CKS2	CKS1	CKS0	res		EGS1	EGS0
Attribute	R/W	R/W	R/W	R/W	R/W		R/W	R/W
Initial value	0	0	0	0	0b00		0	0

■ **Timer Control Register 2 (Low-order bytes of TMCR)**

bit	7	6	5	4	3	2	1	0
Field	T32	FMD2	FMD1	FMD0	OSEL	MDSE	CTEN	STRG
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

■ **Status Control Register (STC)**

bit	7	6	5	4	3	2	1	0
Field	res	TGIE	res	UDIE	res	TGIR	res	UDIR
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

bit	15	0
Field	PCSR [15:0]	
Attribute	R/W	
Initial value	0xXXXX	

■ **Register configuration**

bit	15	14	13	12	11	10	9	8
Field	SEL67_3	SEL67_2	SEL67_1	SEL67_0	SEL45_3	SEL45_2	SEL45_1	SEL45_0
Attribute	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

Figure 8. EEPROM Library Type A, B, and D Flash Algorithm

Command	Number of writes	1st write		2nd write		3rd write		4th write		5th write		6th write	
		Address	Data	Address	Data	Address	Data	Address	Data	Address	Data	Address	Data
Read/ Reset	1	0xXXX	0xF0	--	--	--	--	--	--	--	--	--	--
Write	4	0xAA8	0xAA	0x554	0x55	0xAA8	0xA0	PA	PD	--	--	--	--
Flash erase	6	0xAA8	0xAA	0x554	0x55	0xAA8	0x80	0xAA8	0xAA	0x554	0x55	0xAA8	0x10
Sector erase	6	0xAA8	0xAA	0x554	0x55	0xAA8	0x80	0xAA8	0xAA	0x554	0x55	SA	0x30
Sector erase suspended	1	0xXXX	0xB0	--	--	--	--	--	--	--	--	--	--
Sector erase restarting	1	0xXXX	0x30	--	--	--	--	--	--	--	--	--	--

X: Any value
PA: Write address
SA: Sector address (Specify any address within the address range of the sector to erase)
PD: Write data

Figure 9. EEPROM Library Type C Flash Algorithm

Command	Number of writes	1st write		2nd write		3rd write		4th write		5th write		6th write	
		Address	Data	Address	Data	Address	Data	Address	Data	Address	Data	Address	Data
Read/ Reset	1	0xXXX	0xF0	--	--	--	--	--	--	--	--	--	--
Write	4	0x1550	0xAA	0xAA8	0x55	0x1550	0xA0	PA	PD	--	--	--	--
Chip erase	6	0x1550	0xAA	0xAA8	0x55	0x1550	0x80	0x1550	0xAA	0xAA8	0x55	0x1550	0x10
Sector erase	6	0x1550	0xAA	0x AA8	0x55	0x1550	0x80	0x1550	0xAA	0xAA8	0x55	SA	0x30
Sector erase suspended	1	0xXXX	0xB0	--	--	--	--	--	--	--	--	--	--
Sector erase restarting	1	0xXXX	0x30	--	--	--	--	--	--	--	--	--	--

X: Any value
PA: Write address
SA: Sector address (Specify any address within the address range of the sector to erase)
PD: Write data

MB9A310K/110K (Library Type A), MB9AFB44N (Library Type B), MB9BF529T (Library Type C), and S6E2HG (Library Type D) are typical MCUs that suit these features. For detailed information, see the product flash programming manual and peripheral manual.

Note: The EEPROM libraries use the work flash sector address from 0x200c4000 ~ 0x200c7fff (SA2 and SA3) in Type A and D, the dual-flash sector address from 0x00204000 ~ 0x00207fff (SA6 and SA7) in Type B, and the sector address from 0x00514000 ~ 0x00517fff (SA6 and SA7) in Type C. Therefore, you cannot use this address if you used these libraries. When dual-flash is used, do not use the same banks of flash simultaneously.

6.2 ROM Use

ROM use describes the library flash area used, as illustrated in Figure 10 through Figure 12.

Figure 10. ROM Use for Type A and D

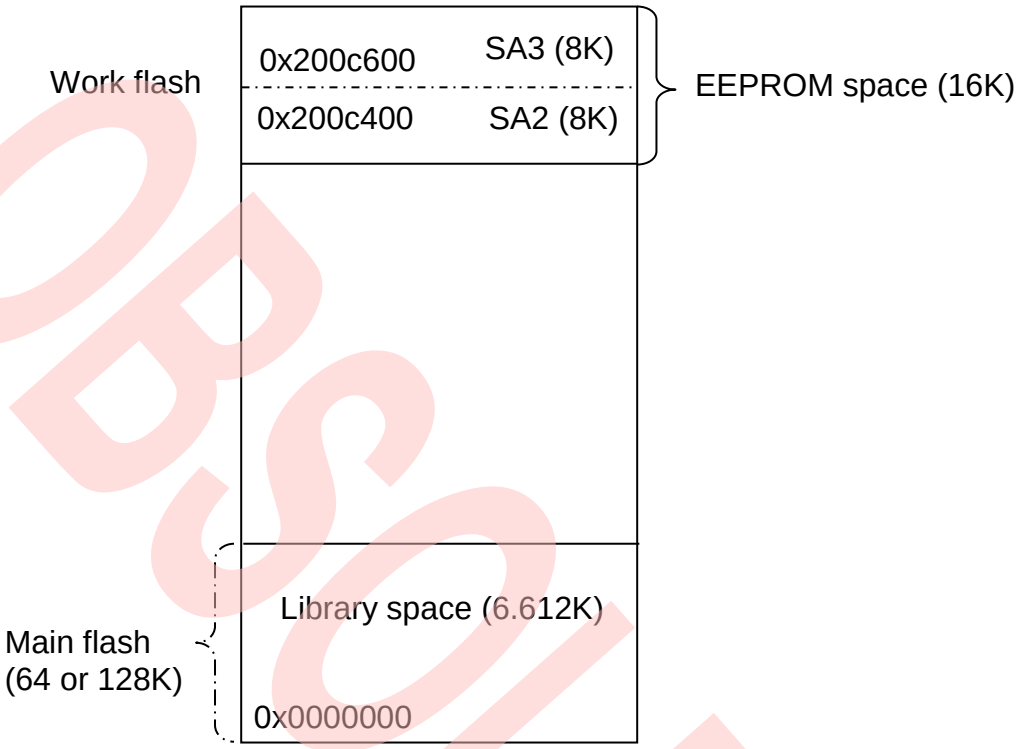


Figure 11. ROM Use for Type B

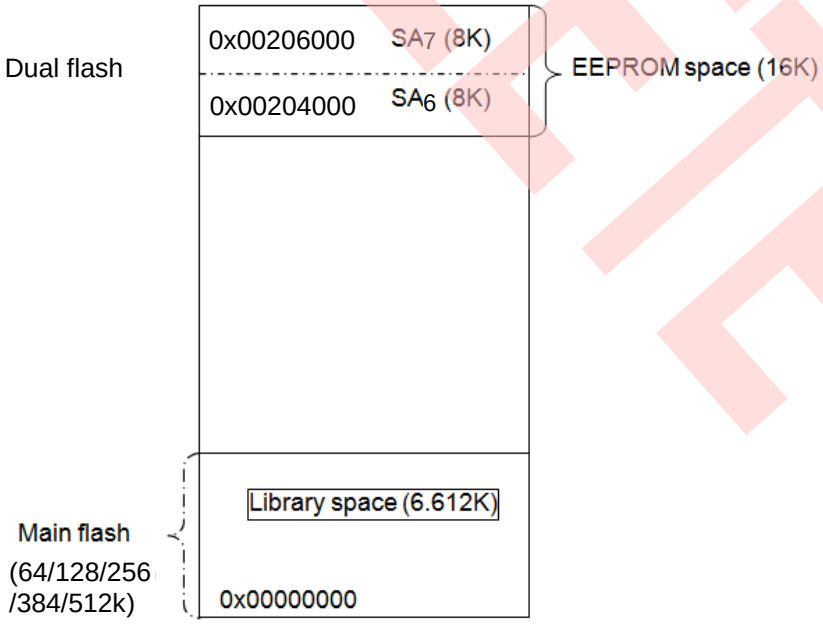
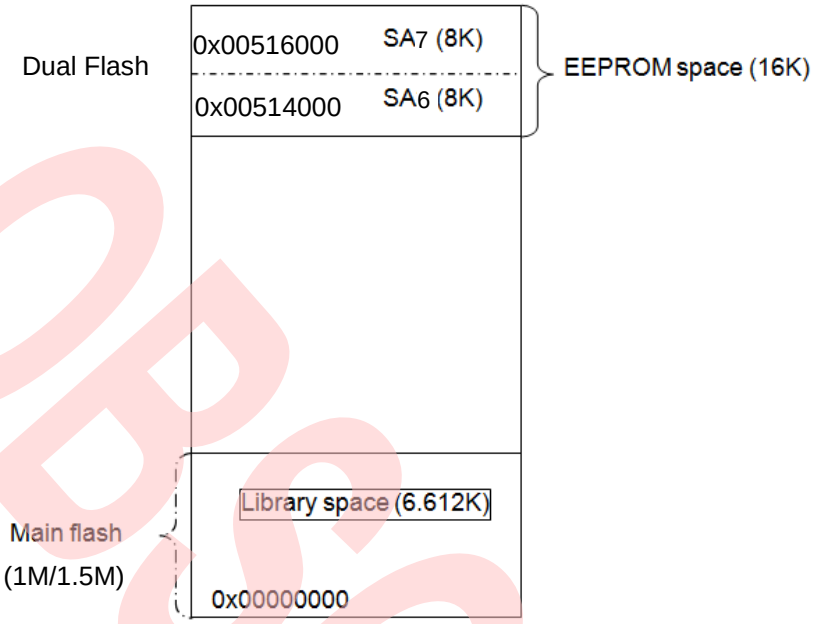


Figure 12. ROM Use for Type C



Note: The library code occupies 6.612 KB, and the occupy space is arranged by the user's project.

6.3 EEPROM Performance Comparison

Table 11 compares the performance of the traditional I²C EEPROM and the emulated EEPROM.

Table 11. EEPROM Performance Comparison

EEPROM Performance Comparison							
Type		**I²C**	**Emulated EEPROM v1.0.0**				
Max transport speed		100 kHz	MCU @40 MHz				
Size		–	2048 bytes	1024 bytes	512 bytes	256 bytes	128 bytes
Single-byte write	Normal	100 µs	76 µs	76 µs	76 µs	76 µs	76 µs
	Max.	–	10 ms	5.5 ms	2.6 ms	1 ms	1 ms
Continuous write, 189 bytes	Normal	13 ms	23.8 ms	23.8 ms	16.9 ms	15.5 ms	15.1 ms
	Max.	–	33.3 ms	28.1 ms	20.1 ms	19.1 ms	19 ms
Continuous write, 377 bytes	Normal	21 ms	47.1 ms	41.9 ms	33.1 ms	31.7 ms	31.4 ms
	Max.	–	451 ms	258 ms	36 ms	32.7 ms	32.4 ms
Single-byte read	Normal	200 µs	8.2 µs	8.2 µs	8.2 µs	8.2 µs	8.2 µs
	Max.	–	–	–	–	–	–

6.4 EEPROM Maximum Rewrite Cycle

The maximum rewrite cycle for the EEPROM is as follows:

- 150, 400, 000 times when 128- and 256-byte size is defined
- 75, 200,000 times when 512-byte size is defined
- 37, 600, 000 times when 1024-byte size is defined
- 18, 800, 000 times when 2048-byte size is defined

Note: These results depend on the MCU erase cycle being 100,000.

Erase/write cycles and data hold time (targeted value)

Erase/write cycles (cycle)	Data hold time (year)
1,000	20
10,000	10
100,000	5

6.5 Emulated EEPROM Performance

The performance of the emulated EEPROM is measured and calculated in the normal condition. It may be longer or shorter, depending on the environment (voltage, temperature, and so on).

- "Normal" means the write data in the average distribution of the EEPROM address.
- "Max." means a continuous write to the EEPROM many times, which causes the block move and erase operation to be triggered.

In conclusion, the performance time of the emulated EEPROM is shorter than that of the I^2C EEPROM.

7 Supported Devices

The following sections list all the MCUs supported by the library.

7.1 Type A MCUs

Table 12 lists the Type A supported MCUs.

Table 12. Type A Supported MCUs

Series	Product Number (Not Including Package Suffix)
MB9AF310K	MB9AF311K, MB9AF312K
MB9AF110K	MB9AF111K, MB9AF112K
MB9BF510R	MB9BF516R, MB9BF515R, MB9BF514R, MB9BF512R
MB9BF510N	MB9BF516N, MB9BF515N, MB9BF514N, MB9BF512N
MB9BF410R	MB9BF416R, MB9BF415R, MB9BF414R, MB9BF412R
MB9BF410N	MB9BF416N, MB9BF415N, MB9BF414N, MB9BF412N
MB9BF310R	MB9BF316R, MB9BF315R, MB9BF314R, MB9BF312R
MB9BF310N	MB9BF316N, MB9BF315N, MB9BF314N, MB9BF312N
MB9BF110R	MB9BF116R, MB9BF115R, MB9BF114R, MB9BF112R
MB9BF110N	MB9BF116N, MB9BF115N, MB9BF114N, MB9BF112N

7.2 Type B MCUs

Table 13 lists the Type B supported MCUs.

Table 13. Type B Supported MCUs

Series	Product Number (Not Including Package Suffix)
MB9AFA40L	MB9AFA44L, MB9AFA42L, MB9AFA41L
MB9AFA40M	MB9AFA44M, MB9AFA42M, MB9AFA41M
MB9AFA40N	MB9AFA44N, MB9AFA42N, MB9AFA41N
MB9AFA40LA	MB9AFA44LA, MB9AFA42LA, MB9AFA41LA
MB9AFA40MA	MB9AFA44MA, MB9AFA42MA, MB9AFA41MA
MB9AFA40NA	MB9AFA44NA, MB9AFA42NA, MB9AFA41NA
MB9AFA40LB	MB9AFA44LB, MB9AFA42LB, MB9AFA41LB
MB9AFA40MB	MB9AFA44MB, MB9AFA42MB, MB9AFA41MB
MB9AFA40NB	MB9AFA44NB, MB9AFA42NB, MB9AFA41NB
MB9AFB40L	MB9AFB44L, MB9AFB42L, MB9AFB41L
MB9AFB40M	MB9AFB44M, MB9AFB42M, MB9AFB41M
MB9AFB40N	MB9AFB44N, MB9AFB42N, MB9AFB41N
MB9AFB40LA	MB9AFB44LA, MB9AFB42LA, MB9AFB41LA
MB9AFB40MA	MB9AFB44MA, MB9AFB42MA, MB9AFB41MA
MB9AFB40NA	MB9AFB44NA, MB9AFB42NA, MB9AFB41NA
MB9AFB40LB	MB9AFB44LB, MB9AFB42LB, MB9AFB41LB
MB9AFB40MB	MB9AFB44MB, MB9AFB42MB, MB9AFB41MB
MB9AFB40NB	MB9AFB44NB, MB9AFB42NB, MB9AFB41NB
MB9AF140L	MB9AF144L, MB9AF142L, MB9AF141L
MB9AF140M	MB9AF144M, MB9AF142M, MB9AF141M
MB9AF140N	MB9AF144N, MB9AF142N, MB9AF141N
MB9AF140LA	MB9AF144LA, MB9AF142LA, MB9AF141LA
MB9AF140MA	MB9AF144MA, MB9AF142MA, MB9AF141MA
MB9AF140NA	MB9AF144NA, MB9AF142NA, MB9AF141NA
MB9AF140LB	MB9AF144LB, MB9AF142LB, MB9AF141LB
MB9AF140MB	MB9AF144MB, MB9AF142MB, MB9AF141MB
MB9AF140NB	MB9AF144NB, MB9AF142NB, MB9AF141NB
MB9AF340L	MB9AF344L, MB9AF342L, MB9AF341L
MB9AF340M	MB9AF344M, MB9AF342M, MB9AF341M
MB9AF340N	MB9AF344N, MB9AF342N, MB9AF341N
MB9AF340LA	MB9AF344LA, MB9AF342LA, MB9AF341LA
MB9AF340MA	MB9AF344MA, MB9AF342MA, MB9AF341MA
MB9AF340NA	MB9AF344NA, MB9AF342NA, MB9AF341NA
MB9AF340LB	MB9AF344LB, MB9AF342LB, MB9AF341LB
MB9AF340MB	MB9AF344MB, MB9AF342MB, MB9AF341MB
MB9AF340NB	MB9AF344NB, MB9AF342NB, MB9AF341NB
MB9AF150M	MB9AF156M, MB9AF155M, MB9AF154M
MB9AF150N	MB9AF156N, MB9AF155N, MB9AF154N

Series	Product Number (Not Including Package Suffix)
MB9AF150R	MB9AF156R, MB9AF155R, MB9AF154R
MB9AF150MA	MB9AF156MA, MB9AF155MA, MB9AF154MA
MB9AF150NA	MB9AF156NA, MB9AF155NA, MB9AF154NA
MB9AF150RA	MB9AF156RA, MB9AF155RA, MB9AF154RA
MB9BF520K	MB9BF524K, MB9BF522K, MB9BF521K
MB9BF520L	MB9BF524L, MB9BF522L, MB9BF521L
MB9BF520M	MB9BF524M, MB9BF522M, MB9BF521M
MB9BF320K	MB9BF324K, MB9BF322K, MB9BF321K
MB9BF320L	MB9BF324L, MB9BF322L, MB9BF321L
MB9BF320M	MB9BF324M, MB9BF322M, MB9BF321M
MB9BF120K	MB9BF124K, MB9BF122K, MB9BF121K
MB9BF120L	MB9BF124L, MB9BF122L, MB9BF121L
MB9BF120M	MB9BF124M, MB9BF122M, MB9BF121M

7.3 Type C MCUs

Table 14 lists the Type C supported MCUs.

Table 14. Type C Supported MCUs

Series	Product Number (Not Including Package Suffix)
MB9BF520S	MB9BF529S, MB9BF528S
MB9BF520T	MB9BF529T, MB9BF528T
MB9BF420S	MB9BF429S, MB9BF428S
MB9BF420T	MB9BF429T, MB9BF428T
MB9BF320S	MB9BF329S, MB9BF328S
MB9BF320T	MB9BF329T, MB9BF328T
MB9BF120S	MB9BF129S, MB9BF128S
MB9BF120T	MB9BF129T, MB9BF128T

7.4 Type D MCUs

Table 15 lists the Type D supported MCUs.

Table 15. Type D Supported MCUs

Series	Product Number (Not Including Package Suffix)
S6E2HG	S6E2HG4, S6E2HG6
S6E2HE	S6E2HE4, S6E2HE6
S6E2H4	S6E2H44, S6E2H46
S6E2H1	S6E2H14, S6E2H16
MB9BF566	MB9BF566M, MB9BF566N, MB9BF566R
MB9BF567	MB9BF567M, MB9BF567N, MB9BF567R
MB9BF568	MB9BF568M, MB9BF568N, MB9BF568R

Note: FM3 MCUs Type 5, Type 6, Type 12 and FM4 Type 1, Type 6 are verified during design. If you have problems using the library on other MCUs, contact Cypress Support.

8 Library Web Address Information

The emulated EEPROM libraries are posted on the web based on MCU type. You can find the MCU type, and then download the related library at **Tools & Software**.

Following are the web download addresses for each type of library:

- Type A Library: http://www.cypress.com/FM-EEPROM-TYPE-A-Emulation-Library
- Type B Library: http://www.cypress.com/FM-EEPROM-TYPE-B-Emulation-Library
- Type C Library: http://www.cypress.com/FM-EEPROM-TYPE-C-Emulation-Library
- Type D Library: http://www.cypress.com/FM-EEPROM-TYPE-D-Emulation-Library

9 Library Use

9.1 Recommended Initialization Procedure

Each time after power on or reset, you should call the check API function. This is the only way to start accessing the emulated EEPROM by software.

Depending on the result of the check function, if the check result is not CONSISTENT, you need to call the define or restore API function.

Figure 13 shows the initialization procedure flow chart.

Figure 13. Initialization Flow Chart

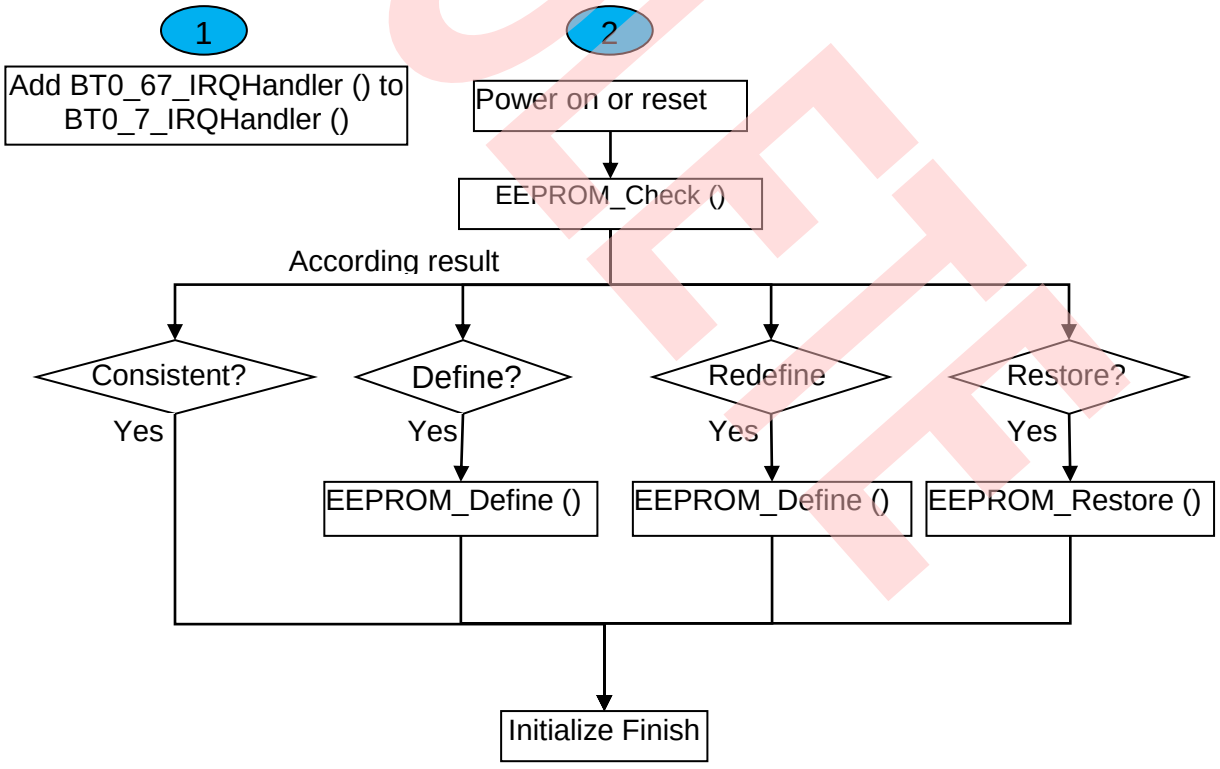


Figure 14 shows step 1, which adds the timer interrupt function.

Figure 14. Interrupt Calling Example

```
void BT0_7_IRQHandler(void)
{

  if(bFM3_BT6_RT_STC_UDIR)
    BT0_67_IRQHandler();
}
```

Type A uses this interrupt in calling code.

Figure 15 shows step 2, which initializes the EEPROM status (check API transfer).

Figure 15. Initialization Example

```
i = EEPROM_Check(E2P_2048B,&RdDat);
if((RdDat == DEFINE)||(RdDat == REDEFINE))
  i = EEPROM_Define(E2P_2048B);
else if(RdDat == RESTORE)
  i = EEPROM_Restore();
```

9.2 Library Use Example

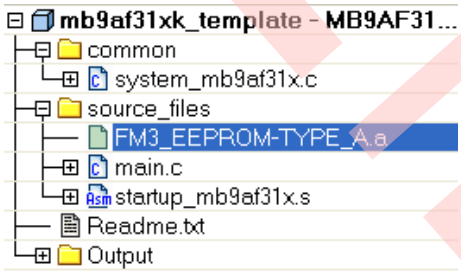
One *.a* file and one *.h* file are open to the user. When you want to use the EEPROM library, follow these steps. (The following example uses EEPROM library Type A.)

1. Add *FM3_EEPROM-TYPE_A.a* and *FM3_EEPROM-TYPE_A.h* to your project file, as shown in Figure 16.

Figure 16. Add File Example

2. Add *FM3_EEPROM-TYPE_A.a* to the project, as shown in Figure 17.

Figure 17. Add Library Example



3. Add the interrupt function to the project base timer function. See Figure 14.
4. Initialize the EEPROM. See Figure 13.
5. Add *#include FM3_EEPROM-TYPE_A.h* to *main.c*.
6. Call the API that you need.

9.3 Notes on Library Use

- When you want to use the libraries, initialize first.
- When you call the define API, do not call the check API immediately afterwards.

- The MCU has the same sectors in the work flash, and the same timer with MB9A310K can use the library: See MCU Application.
- Base timer channel 6 and channel 7 are used for the EEPROM; therefore, you cannot use them and disable the base timer interrupt too.
- Work flash sector2 and sector3 are used. Therefore, you cannot use these two sectors in Type A. See Figure 4.
- Dual flash sector6 and sector7 are used. Therefore, you cannot use these two sectors in Type B and C. See Figure 5 and Figure 6.
- When you are using other sectors (except read) of the work flash, do not use these libraries at the same time. Similarly, when you are using the libraries (except read), you cannot write, erase, and suspend the work flash.
- When the dual-flash MCU is using the library, set the code start and vector table address to the upper bank.
- For FM0 MCUs, there is no validated library. If you want to use the library on it, contact Cypress Support first.

10 Additional Information

For more Information on Cypress microcontrollers, visit:

- http://www.cypress.com/cypress-microcontrollers
- www.cypress.com

Document History

Document Title: AN200249 – FM MCU EEPROM Emulation Library

Document Number: 002-00249

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	4931297	HUAL	10/21/2015	New application note transfer from MCU-AN-510060-E-14-FM3_EEPROM-FW
*A	5043784	HUAL	01/14/2016	Added Type D EEPROM information and library web information and described the EEPROM library types.
*B	5826368	AESATMP9	07/21/2017	Updated logo and copyright.
*C	6268632	CHMA	08/01/2018	Obsoleted.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Arm® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6 MCU

Cypress Developer Community

Community | Projects | Video | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709