The following document contains information on Cypress products. The document has the series name, product name, and ordering part numbering with the prefix “MB”. However, Cypress will offer these products to new and existing customers with the series name, product name, and ordering part number with the prefix “CY”.

How to Check the Ordering Part Number

1. Go to www.cypress.com/pcn.
2. Enter the keyword (for example, ordering part number) in the **SEARCH PCNS** field and click **Apply**.
3. Click the corresponding title from the search results.
4. Download the Affected Parts List file, which has details of all changes

For More Information

Please contact your local sales office for additional information about Cypress products and solutions.

About Cypress

Cypress is the leader in advanced embedded system solutions for the world's most innovative automotive, industrial, smart home appliances, consumer electronics and medical products. Cypress' microcontrollers, analog ICs, wireless and USB-based connectivity solutions and reliable, high-performance memories help engineers design differentiated products and get them to market first. Cypress is committed to providing customers with the best support and development resources on the planet enabling them to disrupt markets by creating new product categories in record time. To learn more, go to www.cypress.com.

AN205292

FM3 MB9A310K/110K Microcontroller with EEPROM Type A

This application note describes how to use FM3 MB9A310K/110K series MCU emulation EERPOM libraries.

Contents

1 Introduction....1
1.1 Purpose....1
1.2 Definitions, Acronyms and Abbreviations....1
1.3 Document Overview....1
2 System Hardware Environment....2
3 Development Environment....2
4 System Functions....2
4.1 Macro Define....2
4.2 Function List....4
4.3 Function Prototype....4
5 Interrupt Function....7
5.1 Function List....7
5.2 Function Prototype....7
6 Library Outline....8
6.1 MCU Application....8
6.2 ROM Usage....10
6.3 EEPROM Performance Comparison....11
6.4 Explanation of Emulated EEPROM Performance....11
7 Usage Sample....12
7.1 Recommend initialize procedure....12
7.2 Library Use Sample....13
8 Usage Notes....14
9 Additional Information....14

1 Introduction

1.1 Purpose

This application note describes how to use FM3 MB9A310K/110K series MCU emulation EERPOM libraries.

1.2 Definitions, Acronyms and Abbreviations

Type A: FM3 MB9A310K/110K series MCU

1.3 Document Overview

The rest of document is organized as the following:

Chapter 2 explains the system hardware environment.

Chapter 3 explains the develop environment.

Chapter 4 explains system functions.

Chapter 5 explains interrupt function.

Chapter 6 explains library outline

Chapter 7 explains usage sample

Chapter 8 explains usage notes.

2 System Hardware Environment

Hardware Environment

- CPU Chip: Cypress MB9A310K, MB9A110K series MCU;
- CPU Frequency: 40MHz;
- Minimum Instruction Time: 25ns;
- Ram Space: 2.82KBytes;
- Code Space: 6.612KBytes;

3 Development Environment

Development Environment

Table 1. MCU Development Environment

Name	Description	Part Number	Manufacture	Remark
IAR	Software Developing IDE	embedded workbench for ARM, version:6.21.1.2846		
J-LINK	Emulate tool		SEGGER	

4 System Functions

4.1 Macro Define

4.1.1 APIs Operation Result

These macros describe the APIs operation result.

Table 2. API Operation Status

Name	Description	Value
OK	API operations normally	0
NG	API operation error	1
BUSY	A priority API is operating	2
PARA_ERROR	The input parameter is not a valid value	4
SYSTBUSY	The API is consistent call that may influence back erase	5

Following items describes the countermeasure when APIs return these macros:

OK --- No need other operation

NG --- Call EEPROM_Check (), according check result call related API and Call API again.

BUSY --- Call API later

PARA_ERROR --- the input EEPORM address or size is wrong

SYSTBUSY --- the back erase is operating, delay 10ms and then call API

Note: Only write and read API can return SYSTBUSY.

4.1.2 EEPROM Check Result

Check API will check the EEPROM status and return the status that EEPROM need to do

Table 3. Result of Check API

Name	Description	Value
CONSISTENT	EEPROM is ready	0
DEFINE	EEPROM need to be defined	1
REDEFINE	EEPROM need to be define again	2
RESTORE	EEPROM status need to be restore	3

Following items describe operations relevant to different result:

CONSISTENT --- EEPROM is OK, do that you want

DEFINE --- Call EEPROM_Define (), EEPROM need be defined

REDEFINE --- Call EEPROM_Define (), EEPROM need be redefined

RESTORE --- Call EEPROM_Restore (), EEPROM status need be restore

For detailed use, please refer to Figure 9.

4.1.3 EEPROM Size

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

When size is different, the address rang that can be used is different, following items shows the range:

128B ---- 0 ~ 127

256B ---- 0 ~ 255

512B ---- 0 ~ 511

1024B ---- 0 ~ 1023

2048B ---- 0 ~ 2047

Note: When EEPROM size is defined, user can used size is decide. So please do not use value that beyond the define value, or else the API will return PARA_ERROR.

4.2 Function List

Prototype	Description	Remark
uint8_t EEPROM_Define(uint8_t ucSize)	Initializes the emulated EEPROM in one of predefined sizes	N/A
uint8_t EEPROM_B_Write(uint16_t WtAdrr, uint8_t WDat)	Write one byte to emulated EEPROM, when sector is full it will copy old data to new sector and erasing old sector	N/A
uint8_t EEPROM_B_Read(uint16_t RAddr, uint8_t *ucData)	Read one byte from emulated EEPROM	N/A
uint8_t EEPROM_Check(uint8_t ucSize, uint8_t *ucData)	Check consistency of the data stored in the emulated EEPROM and the information* stored in RAM. It informs whether it is blank, consistent, broken but it can be restored or broken and it cannot be restored. This function is intended to be performed every time after power on. Therefore, it reconstructs the EEMS stored in RAM that is volatile by its nature.	Before use libraries call this API First
uint8_t EEPROM_Restore (void)	Restore the EEPROM including the EEMS in RAM. This function is intended to solve data inconsistency that is caused by incomplete execution of the API software	N/A

*to emulate EEPROM using flash memory, some information need to be defined in RAM area, it is called EEMS (Emulated EEPROM Management Structure) hereafter.

4.3 Function Prototype

4.3.1 EEPROM_Define ()

Prototype	uint8_t EEPROM_Define(uint8_t ucSize)
Parameter:	ucSize: defined EEPROM size, refer Table 4
Return	Return API operation status, refer Table 2
Description	Define a emulate EEPROM and relate size.
Remark	This API decide EEPROM size

This API defines EEPROM size and status. The return value of this API is listed as following:

Table 5. Return Status of Define

Return Status	Operation	Method
OK	Define operation successfully	N/A
NG	flash operation timeout in or preceding API lead flash operation time out	Call again later
BUSY	Other API is doing	Call API later
PARA_ERROR	The input EEPROM size is not in define list	Use EEPROM size in list, refer Table 4

For detailed usage for this API, please refer Figure 9.

4.3.2 EEPROM_B_Write ()

Prototype	uint8_t EEPROM_B_Write(uint16_t WtAdrr, uint8_t WDat)
Parameter:	WtAdrr: wrote EEPROM address (smaller or equal to defined size) WDat: wrote data
Return	Return API operation status, refer Table 2
Description	Write one byte data to EEPROM.
Remark	N/A

This API writes one byte data to EEPROM. The return value of this API is listed as following:

Table 6. Return Status of Write

Return Status	Operation	Method
OK	Write operation successfully	N/A
NG	Read error or flash operation timeout in preceding API	Check again
BUSY	The check, define or restore API is doing	Call API later
PARA_ERROR	The input EEPROM address is beyond EEPROM size	Modify EEPROM address in range
SYSTBUSY	The sector erase is doing and the continue write times is too many that may influence the erase	Delay 10ms and then write

Following pictures are sample for write API:

Figure 1. Continuous Write Sample

```
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 Sample

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

Note: Return value “i” indicates the API operation status. For detailed, please refer Table 2.

4.3.3 EEPROM_B_Read ()

Prototype	uint8_t EEPROM_B_Read(uint32_t RAddr,unsigned char *ucData)
Parameter:	RAddr: EEPROM address that user want to read(smaller or equal to defined size) *ucData: save the read data
Return	Return API operation status, refer Table 2
Description	Red one byte data from EEPROM
Remark	N/A

This API read data from EEPROM. The return value of this API is listed as following:

Table 7. Return Status of Read

Return Status	Operation	Method
OK	Read operation successfully	N/A
NG	Read error or flash operation timeout in preceding API	Check again
BUSY	The check, define or restore API is doing	Call API later
PARA_ERROR	The input EEPROM address is beyond EEPROM size	Modify EEPROM address in range
SYSTBUSY	The sector erase is doing and the continue read times is too many that may influence the erase	Delay 10ms and then read

Following picture is a sample for call read API:

Figure 3. Read Sample

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

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

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

4.3.4 EEPROM_Check ()

Prototype	uint8_t EEPROM_Check(uint8_t ucSize, uint8_t *ucData)
Parameter:	ucSize: defined EEPROM size, refer Table 4 *ucData: checked EEPROM status, refer Table 3
Return	Return API operation status, refer Table 2
Description	Check EEPROM status that EEPROM need redefine or restore and so on
Remark	N/A

This API checks EEPROM status, whether the EEPROM is read or broken. The return value of this API is listed as following:

Table 8. Return Status of Check

Return Status	Operation	Method
OK	Check operation successfully	N/A
NG	Flash operation timeout.	Check later
BUSY	System is doing other API	Check later
PARA_ERROR	Input EEPROM size(ucSize) is not the defined value, refer Table 4	Modify input size, refer Table 4

Note: User need according check result to call relevant API. For detailed usage for this API, please refer Figure 9.

4.3.5 EEPROM_Restore ()

Prototype	uint8_t EEPROM_Restore (void)
Parameter:	void
Return	Return API operation status, refer Table 2
Description	Restore EEPROM status
Remark	Just EEPROM EEMS be Restored,

This API restores EEPROM status in abnormal power off and other abnormity. The return value of this API is listed as following:

Table 9. Return Status of Restore

Return Status	Operation	Method
OK	Restore successfully	N/A
NG	Data cannot be restored, sector erase timeout or preceding API flash timeout.	Check again
BUSY	System is doing other API	Call API later

For detailed usage for this API, please refer Figure 9.

5 Interrupt Function

5.1 Function List

Prototype	Description	Remark
void BT0_67_IRQHandler(void)	EEPROM timer interrupt function	N/A

5.2 Function Prototype

5.2.1 BT0_67_IRQHandler ()

Prototype	void BT0_67_IRQHandler(void)
Parameter:	void
Return	void
Description	Timer interrupt that user need to add it to timer interrupt function
Remark	Please do not disable Base Timer interrupt

Note: This API is just called in BT0_7_IRQHandler () will be OK. For detailed usage, please refer Figure 8.

6 Library Outline

6.1 MCU Application

The library can be used in FM3 two independent on-chip flashes MCU that has following features:

- Work flash has sector (range is 0x200c4000~0x200c5fff)
- Work flash has sector (range is 0x200c6000~0x200c7fff), refer Figure 4
- Base timer channel6 and channel7 (registers refer Figure 5)
- Work flash arithmetic is same as Figure 6

Following pictures show the detailed features for sector, timer registers and flash arithmetic:

Figure 4. EEPROM Used Sectors

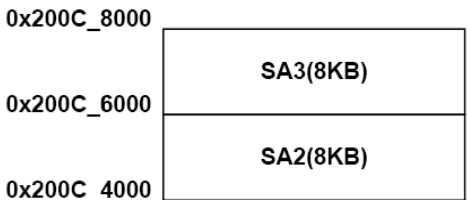


Figure 5. Base Timer Channel 6&7 Registers

■ **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 6. Work Flash Arithmetic

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	--	--	--	--	--	--	--	--	--	--

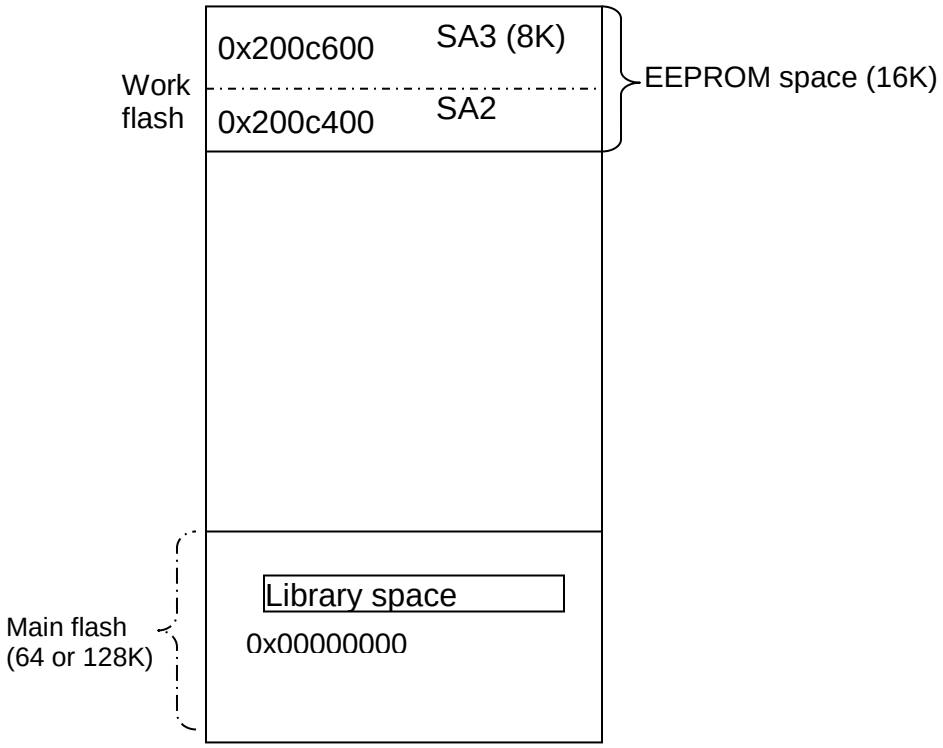
X: Any value
PA: Write address
SA: Sector address (Specify any address within the address range of the sector to erase)
PD: Write data

MB9A310K/110K series FM3 MCU (Type 5) is the typical MCU that suit these features. For detailed information please refer to flash programming manual and peripheral manual for MB9A310K.

Note: The libraries used work flash sector address from 0x200c4000 ~ 0x200c7fff (SA2 and SA3), so user can not use this address, if user used these libraries.

6.2 ROM Usage

The ROM usage describes the library used flash area:



Note: The library code occupy 6.612K and the occupy space is arranged by user project.

6.3 EEPROM Performance Comparison

Below table shows the performance comparison between traditional I2C EEPROM and emulation EEPROM.

Table 10. FM3 MB9A310K/110K EEPROM Performance

EEPROM Performance Comparison							
Type		I2C	Emulate EEPROM V1.0.0				
Max transport speed		100KHz	MCU @40MHz				
Size		-	2048B	1024B	512B	256B	128B
Single byte write	Normal	100us	76us	76us	76us	76us	76us
	Max	-	10ms	5.5ms	2.6ms	1ms	1ms
Continue write 189B	Normal	13ms	23.8ms	23.8ms	16.9ms	15.5ms	15.1ms
	Max	-	33.3ms	28.1ms	20.1ms	19.1ms	19ms
Continue write 377B	Normal	21ms	47.1 ms	41.9ms	33.1ms	31.7ms	31.4ms
	Max	-	451ms	258ms	36ms	32.7ms	32.4ms
Single byte read	Normal	200us	8.2us	8.2us	8.2us	8.2us	8.2us
	Max	-	-	-	-	-	-

6.4 Explanation of Emulated EEPROM Performance

This performance is measured and calculated in the normal condition. And it may be longer or shorter depend on environment (voltage, temperature …).

Normal case means write data in the average distribute of EEPROM address.

Max case means continually write EEPROM any times which causes block move and erase operation trigger.

Conclusion is that the emulated EEPROM is shorter than I2C EEPROM.

7 Usage Sample

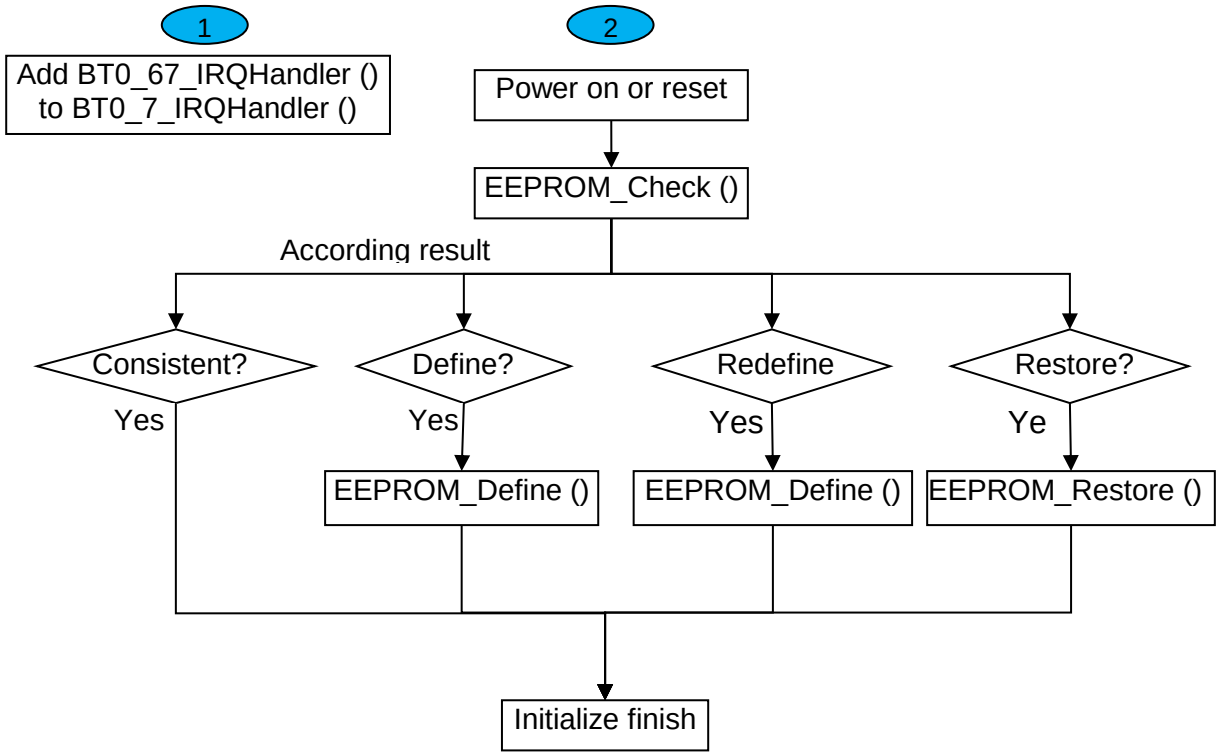
7.1 Recommend initialize procedure

Each time after power on or reset, user need to call check API, this is the only way to start accessing the emulated EEPROM by software.

Depend on the result of check API, if the check result is not the “consistent”, user need to call “define” or “restore” API.

Following figure roughly describes the initialize procedure flow chart:

Figure 7. Initialization Flow Chart



Following picture shows the step1 that add timer interrupt function.

Figure 8. Interrupt Calling Sample

1

```
void BT0_7_IRQHandler(void)
{

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

Following picture shows the step2 that initialize EEPROM status (check API transfer):

Figure 9. Initialize Sample

2

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

7.2 Library Use Sample

There are one .a file and one .h file opened to user. When user wants to use the EEPROM library, please refer following steps:

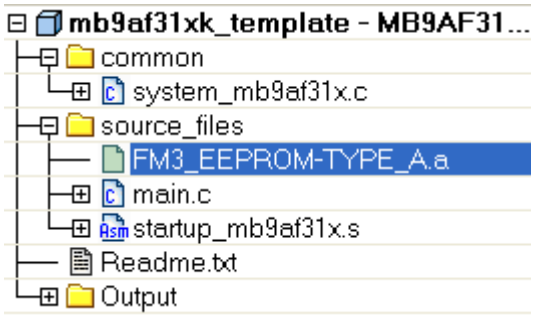
- Add “FM3_EEPROM-TYPE_A.a” and “FM3_EEPROM-TYPE_A.h” to user’s project file.

Figure 10. Add File Sample



- Add “FM3_EEPROM-TYPE_A.a” to project.

Figure 11. Add Lib Sample



- Add interrupt function to project base timer function. Please refer “Figure 8”
- Initialize EEPROM, please refer “Figure 7”
- Add “#include FM3_EEPROM-TYPE_A.h” in main.c
- Call API that user need

8 Usage Notes

- When user want to use these libraries, please initialize first
- When define API is called, do not call Check API immediately.
- If the FM3 MCU has the same sectors in work flash and the timer is same with MB9A310K, the LIB can be used to the MCU too. Refer chapter 6.1
- Base timer ch6 and ch7 are used for EEPROM, so user can not use these timers and can not disable the base timer interrupt too.
- The work flash sector2 and sector3 are used, so user can not use these two sectors. For detailed address refer Figure 4
- When user is using (except read) work flash other sectors, please do not use these libraries at the same time.
- Similarly when user is using libraries (except read), user can not write, erase and suspend work flash.

9 Additional Information

For more information on Cypress Semiconductor products, visit the following websites:

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

http://www.cypress.com/cypress-mcu-product-softwareexamples

Document History

Document Title: AN205292 - FM3 MB9A310K/110K Microcontroller with EEPROM Type A

Document Number: 002-05292

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	HUAL	07/30/2012	Initial Release
*A	5045107	HUAL	12/10/2015	Migrated Spansion Application Note from MCU-AN-510060-E-10 to Cypress format
*B	5593141	HUAL	01/19/2017	Request from HUAL to post the document on web along with Chinese version
*C	5831988	MALI	07/25/2017	Updated logo and copyright.

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

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

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

© Cypress Semiconductor Corporation, 2012-2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.