

Freescale Semiconductor

Application Note

Document Number: AN3743 Rev. 0, 10/2008

Emulated EEPROM Quick Start Guide

by: Matthew Grant 16-Bit Applications Freescale Semiconductor

1 Introduction

1.1 Emulated EEPROM

Some families of the <u>S12X series of microcontrollers</u> (MCUs), such as the MC9S12XEP100, have an enhanced region labeled as EEPROM in the memory map. This region does not contain conventional EEPROM, but has flash and RAM components that can be managed by the enhanced flash module (FTM) in a unique manner, allowing the emulation of EEPROM. Because the FTM allows the partitioning of EEPROM resources to suit an application, a significant degree of memory flexibility becomes available to users.

1.2 Objective

The objective of this application note is to accelerate the introduction of emulated EEPROM (EEE). In particular, it is suited to individuals familiar with programming and the use of a debugger but who have little or no experience

Contents

1	Intro	oduction
	1.1	Emulated EEPROM1
	1.2	Objective
2	S12	XE EEE in the S12X Memory Map 2
3	Cor	figurations of the Enhanced EEPROM Region 3
	3.1	Launching FTM Commands 4
	3.2	Checking for Errors
4	EEE	EPROM Quick Start Project Setup
5	Acc	essing EEPROM 10
6	EEE	E Quick Start Code Steps 12
7	Sun	nmary



© Freescale Semiconductor, Inc., 2008. All rights reserved.



S12XE EEE in the S12X Memory Map

with EEE. This document seeks to teach users through a simple EEE example. Most low-level details are put aside so as to allow focus on higher-level concepts associated with the EEE. Users will need a programming board with an S12XE MCU and a debugger to single-step through the example code and run through portions of that code. This document shows:

- Where the EEE components exist in the memory map
- Examples of three configurations
- The FTM command flow
- How data behaves in the buffer RAM in different configurations

Other important topics are also discussed. For more details on EEE, refer to Freescale application note AN3490, "Overview of the MC9S12XE Emulated EEPROM," available at www.freescale.com.

2 S12XE EEE in the S12X Memory Map

Figure 1 shows how this enhanced EEPROM region fits into the S12XE local and global memory map. The figure also reveals the enhanced EEPROM components of data flash and buffer RAM.



Figure 1. EEPROM Region in the S12XE Memory Map

Emulated EEPROM Quick Start Guide, Rev. 0



3 Configurations of the Enhanced EEPROM Region

Because of the enhancement of the FTM, these EEE components can be used in three general configurations described below:

As Data Flash and Buffer RAM: All sections of buffer RAM and sectors of data flash are used as conventional RAM and flash. No partitions of EEE are created, and EEPROM emulation is disabled.



Figure 2. EEPROM Region Configured as Data Flash and Buffer RAM

As EEE: All of the EEE resources are fully partitioned as EEE. This results in all buffer RAM being used as EEE RAM, and all data flash used as EEE flash. Once EEE is enabled, this allows the FTM to take volatile data written into the EEE RAM and store records of this data in EEE flash. The FTM can automatically manage these records without user intervention. After the next MCU reset, the FTM will stall the central processing unit (CPU) from executing instructions until data from the records in EEE flash has been copied back to EEE RAM. This management of data allows the EEE RAM to be perceived as EEPROM, except that users can modify data in this section at any time.



Figure 3. Configured as EEE

As EEE, Data Flash, and Buffer RAM: This is a mixed configuration. Portions of the buffer RAM and the data flash are partitioned as EEE RAM and EEE flash, respectively. The remaining sections and sectors still exist as buffer RAM and data flash and may be used as desired. Data placed in EEE RAM will be managed by the FTM to support EEE.



Configurations of the Enhanced EEPROM Region

EEE Region	
Nonvolatile	User Data Flash
Nonvolatile	EEE Flash
Volatile	User Buffer RAM
Nonvolatile RAM	EEE RAM
	EEE Region Nonvolatile Nonvolatile Volatile Nonvolatile

Figure 4. Configured as EEE, Data Flash, and Buffer RAM

3.1 Launching FTM Commands

In order to write, read, or configure the memory regions associated with the FTM, commands must be loaded into the FTM and executed. The process involves a specific sequence that is used to set the FTM command parameters and launch execution. The following portion of code illustrates an example of instructions within a function, "LaunchFlashCommand," used to set up and execute FTM commands. Users should follow a defined flow to reduce the risk of module errors. Figure 5 shows the recommended FTM command execution flow.



```
FSTAT = 0 \times 30;
                               //Use store instruction to clear ACCERR, FPVIOL.
/********SET UP THE FTM COMMAND AND PARAMETERS********/
FCCOBIX = 0;
                               //Set CCOB index to 0 to begin command setup.
FCCOBHI = ccob0high;
                               //Write ccob0 high-byte command value.
FCCOBLO = ccob0low;
                               //Write ccob0 low-byte parameter, if used.
if (params > 0)
                               //Continue if more parameters to specify.
{
 FCCOBIX++;
 FCCOB = ccob1;
                               //Write next word parameter to CCOB1 buffer.
  if (params > 1)
                               //Continue if more parameters to specify.
  {
   FCCOBIX++;
   FCCOB = ccob2;
                               //Write next word parameter to CCOB2 buffer.
    if (params > 2)
                               //Continue if more parameters to specify.
    ł
     FCCOBIX++;
     FCCOB = ccob3;
                               //Write next word parameter to CCOB3 buffer.
     if (params > 3)
                               //Continue if more parameters to specify.
      ł
       FCCOBIX++;
                               //Write next word parameter to CCOB4 buffer.
       FCCOB = ccob4;
       if (params > 4)
                               //Continue if more parameters to specify.
         FCCOBIX++;
         FCCOB = ccob5;
                               //Write next word parameter to CCOB5 buffer.
         if (params > 5)
                               //Continue if more parameters to specify.
          ł
           FCCOBIX++;
           FCCOB = ccob6;
                               //Write next word parameter to CCOB6 buffer.
           if (params > 6)
                               //Continue if more parameters to specify.
            {
             FCCOBIX++;
                               //Write next word parameter to CCOB7 buffer.
             FCCOB = ccob7;
            }
         }
       }
     }
    }
  }
}
FSTAT = 0x80;
                               //Clear buffer-empty-flag to start FTM command.
                               /\,/ \text{Now} wait for the FTM command to complete.
while (!FSTAT_CCIF);
status.fstat_var = FSTAT;
                               //Copy FSTAT register state for later comparison.
status.ferfstat_var = FERSTAT; //Copy FERSTAT register state for later comparison.
return(status);
                               //After FTM command completed, return status.
```



Configurations of the Enhanced EEPROM Region



Generic Flash Command Write Sequence Flowchart

Figure 5. FTM Command Execution Flowchart

3.2 Checking for Errors

After the completion of each FTM command, indicated by a set CCIF flag, it is important to check both flash status registers. The FSTAT and FERSTAT registers contain error flags that are set whenever a particular error condition is met. If errors are detected, the user should use software to mitigate the occurrence and clear the error flag. When registers such as FSTAT and FERSTAT have multiple bits that can be cleared via a write 1, users should avoid the use of the BSET instruction to clear these bits. This could result in an unintended clear of other flags within the register. If applicable, a STAA instruction or STAB instruction may be more appropriate.



4 EEEPROM Quick Start Project Setup

The EEEPROM Quick Start project containing the example code for this application note can be downloaded from the Freescale public website. Users without a compiler can also download a free version of CodeWarriorTM for the S12X. To begin setup of the EEEPROM Quick Start project, open the "EEEPROM_Quick_Start.mcp" project file. Once opened, users should select and open the "main.c" file, which contains the majority of the code referenced in this application note. Comments have been placed throughout the file, along with step numbers that correspond with the discussion in this application note.

Freescale CodeWarrior File Edit View Search Project Processor Expert Window Help														
jile Edit View Search Project Processor Expert Window Help														
🍓 🏝 🛋 🔲 🗛 🐼 🕨	a . b .	A	M (M)	🕒 🔈 🗔 🖡										
			<u> </u>											
EEEPROM_Quick_Start.mcp														
SofTec HCS12														
Files Link Order Targets														
🖌 File	Code	Data 🔞	* A											
readme.txt	n/a	n/a •	x											
🚺 tips.txt	n/a	n/a •	_											
⊡ 🥽 Sources	1K	2 •	• 🔳		Open the									
📗 main.c 🔸	632	0 •	+ 🔳		main c file									
📲 datapage.c	546	0 •	• 🔳											
xgate.cxgate	480	2 •	• 🔳											
🛛 👘 xgate.h	0	0 •												
⊞	71	6•	• 🔳											
	0	0 •												
	n/a	n/a •												
P&E_Multilink_CyclonePro	n/a	n/a												
BUML_linker.prm	n/a	n/a	<u> </u>											
Abatron_BUI_linker.prm	n/a	n/a	프											
Sorrec_HUS12_Inker.pm	n/a	nza •	프											
Full_Crip_Simulation_inke	ri/a 0	riza 0 •												
	15K	4K -												
	131	чт. • П •												
MC9S12XEP100 c	n D	895 •	+ 🗐											
ansixbi.lib	7672	2019 •												
	8K	1K •												
	0	0 •												
🕀 🧰 Debugger Cmd Files	Ō	0 •												

Before attempting to compile and load the project, a few important settings must be checked. First, a target should be selected that matches the communication device that the user will use when programming the MCU.

NP

EEEPROM Quick Start Project Setup



Next, a suitable flash clock divide value (FCLK_DIV) must be defined to ensure that the FTM works properly. In the following clock divide section of "main.c," the user should ensure that a viable FCLK_DIV value is uncommented, or should define a new divide value that corresponds to the oscillator on the MCU board. A table of suitable values can be found in the S12XE flash block guide. In this example, the code does not engage the clock generator (CRG) module, but runs entirely from an external oscillator.

```
//CLOCK DIVIDE SECTION
```

```
//Uncomment the FCLK_DIV value according to the oscillator crystal.
//These values are chosen from the MC9S12XE100 datasheet.
//#define FCLK_DIV 0x1 // Flash clock divider for 2MHz crystal
#define FCLK_DIV 0x3 // Flash clock divider for 4MHz crystal
//#define FCLK_DIV 0x7 // Flash clock divider for 8MHz crystal
//#define FCLK_DIV 0x9 // Flash clock divider for 10MHz crystal
//#define FCLK_DIV 0x18 // Flash clock divider for 26MHz crystal
```

In the following code section, the parameters for partitioning the EEPROM resources are defined. A macro and #define are used to define the size of the EEE before compiling the project. The amount of buffer RAM to be set aside as EEE RAM is specified, as well as the amount of flash to be reserved as data flash. The remaining resources of data flash will be used as EEE flash. Once defined, these parameters are then used later in the FTM command process to fully partition and implement the resources.

Other configurations are possible. The user may adjust these parameters within proper limitations and proper ratio, as explained in application note AN3490.¹

1. The formula used to calculate the number of sectors of data flash for partitioning uses the minimum ratio of partitioned buffer RAM sections to partitioned data flash sectors. Other ratios are possible. Refer to AN3490 for more details.



#endif

When the FTM creates a full partition using the parameter values above, the arrangement yields 1 section of EEE RAM and 12 sectors of EEE flash. The remaining sections and sectors exist as buffer RAM and data flash, respectively. Although sectors describe physical divisions that are present within flash, the term "sections" is used here to describe only imaginary divisions of buffer RAM. Figure 6 shows a block diagram of the resulting configuration of the EEPROM region. The figure identifies the buffer RAM address increases. When EEE is enabled, the FTM copies new data written in EEE RAM and stores records of that data in the sectors partitioned as EEE flash.



Example Configuration: 1 EEE RAM Section, 12 EEE Flash Sectors

Figure 6. Example Configuration: 1 EEE RAM Section, 12 EEE

To use the example, compile and load the code into an S12XE MCU. If using CodeWarrior, this should cause the debugger to appear. Figure 7 shows a debugger image capture where a memory window and variables can be used to observe registers and data values in the EEPROM region. More captures and figures follow to highlight important steps within the project.



Accessing EEPROM

File View Run inDART-HCS12 Component Memory Window Help							
□☞■ ४๒€ १№ →२국4२ㅋ ⋺							
S Source:1	📖 Memo	ory					
HC12 D:\Data\Freescale\TSPG\8_and_16_Bit\XTestProjects\EEEPROM_ Line: 345							
<pre>return(sdest[index]); //Return last character ad void main(void) { volatile char before[]="BEFORE"; //Chars to be written to the I volatile char during[]=" DURING";//Chars to be written to the I volatile char after[]=" AFTER"; //Chars to be written to the I volatile char after[]=" AFTER"; //Chars to be written to the I volatile char after[]=" AFTER"; //Chars to be written to the I volatile char after[]=" AFTER"; //Chars to be written to the I</pre>	13FD00 13FD10 13FD20 13FD30 13FD40 13FD50 13FD50 13FD60 13FD70 13FD70	B7 7F FD 6D A4 C4 83 21 E5 F6 B7 EF 44 05 21 C9	D1 B 47 7 A9 5 DD 2 E7 D A9 6 24 6 50 D	F D3 D FE 1 36 8 85 F 58 1 CA 2 7E A 14	65 86 88 71 F2 EB 00 05	21 9 69 I 45 6 01 I EA 7 7F 9 62 3 0D 2	95 DC 68 DC 7B 5E 36 28
signed char diPart, erPart; //Holds EEE partition size of	13FD80	7E DF 7D B9	EF 9	F 5F 5 12	63 नन	OD A	AA C6
	13FDA0	57 84	00 8	D C8	22	4E 3	36
Assembly:1	13FDB0	14 35	5E D	7 2B	DD	29 5	52
	13FDC0	B7 C1	AC D	C FF	DE	F6 7	7F
HC12 main	13FDD0	07 7A	56 B	B E3	9C	DE 7	7D
FE80F8 LEAS -30,SP	13FDE0	96 71	45 9	D 58	00	ED 4	4B
FE80FB LDD #17730	13FDF0	53 59	91 2	4 00	81	04 0	01
	13FE00	F8 C3	DD 3	E E7	88	EF	74
Register:1 P Pro 🗆 🗙	132220	7C C4	35 L 20 D	5 19	95	23 8	AE 47
HC12 HC12	135520	16 99	50 D		01	60 4	ч/ лл
	13FE40	78 78	D7 8	E 25	47	BS F	82
	135550	F9 7F	व नद	F 4D	~5	C9 0	95
	13FE60	10 48	40 1	4 77	48	88 1	14
IP FE80F8 PC 80F8 PPAGE FE	13FE70	81 A0	20 4	7 42	68	15 1	16
SP 20FD IPL 0 U U CCR SXHIN	13FE80	5A 7F	9F F	5 7F	F3	68 I	DD
	13FE90	FB 63	FE F	F ED	FB	F8 (07
🔂 Data:1 🔤 🔿 Data:2	13FEA0	1A 11	34 0	4 B4	00	41 4	49
HC12 r HC12 main	13FEB0	90 41	A4 0	3 41	F5	01 E	B3
	13FEC0	97 59	C1 F	F D7	B5	FF (C3
E FSTAT (1) volatile E Status	13FED0	DE BF	6F B	7 F6	75	CF I	F1
H RCCOBIY (1) volatile H during	13FEE0	32 87	30 8	C 48	58	55 8	81
	13FEF0	8D 98	01 2	5 28	C0	82 4	49
Figure 7. Debugger Example							

5 Accessing EEPROM

To read and write within the EEPROM region, pointers can be assigned appropriate address locations. This is done at the beginning of the main function, as shown in the code below and in Figure 8. At different stages of the programming example, text strings will be written into adjacent sections 1 and 2 of EEE RAM and buffer RAM, respectively. In this case, adjacent sections are chosen to make the difference in behavior more apparent in the memory window. As users proceed, the behavior of the text strings should be observed after each example step. This will help users understand the emulated properties of EEE RAM versus general buffer RAM.



Source:1 Memory
HC12 D:\Data\Freescale\TSPG\8_and_16_Bit\XTestProjects\EEEPROM_ Line: 357
EnableInterrupte:
section1 ptr = (signed char t) 0x0F00. (Assign beginning add 13F130 16 88 60 20 F8 01 60 4
//buffer BAM to pointe 138540 75 75 77 85 75 47 85 F
section2 ptr = (signed char *) 0x0E00;//Assign beginning add 13FP50 F9 7F AF FE 4D C5 C9 9
//buffer RAM to pointe: 13FE50 10 A8 40 14 77 48 88 1
13FE 0 81 A0 20 47 42 68 15 1
/******************* (1) SET UP THE FLASH CLOCK DIVIDE REGISTER********/ 📮 13FE30 5A 7F 9F F5 7F F3 68 E
This (IT COTF) (Whit for FTM to be ready 13FESO FB 63 FE FF ED FB F8 C
13FEAD 1A 11 34 04 B4 00 41 4
A Assembly:1
HC12 main 13FEC1 97 59 C1 FF D7 B5 FF D7
ISFED DE BF 6F B7 F6 75 CF B
FE815D LDV #3584
FE8160 STY 3, SP
Register:1
HC12 Auto HC12 13FF30 45 C5 D9 84 DD 45 8C 8
D 4552 A 45 B 52 main () 13FF 0 F 6F E7 F5 BF FE 0A 4
TX F00 IV 0 13FFs0 4F EC F0 2F F5 51 3D 5
TP FF915D PC 915D PPAGE FF
13FF70 00 11 D1 31 30 0A 43 9
SP ZODE IFL O O O CCK SANIT
HCI2 J III HCI2 Address FULL Size I Imain II SFED 22 4F 94 10 00 02 14 5
E _FCLKDIV <1> volatile E status
FSTAT <1> volatile before
E FCCOBIX <1> volatile E during 13FFF0 26 8 C4 51 31 A9 A7 1
FCCOB <2> volatile # after 140000 CA E CA FE CA
FERSIAT <1> VOLATILE diPart 140010 CA HE CA FE C
EFAGE <1> Volatile Fraft 140020 CA t CA FE CA F
LIAG (22) VOIALITE I SECTION DE MUCCHZEDON 140030 CA PC CA FE CA F
Reginning address
pointer assigned beginning of
section address in C code or sections 1 and 2.

Figure 8. Assigning Address Locations to Pointers

Emulated EEPROM Quick Start Guide, Rev. 0



EEE Quick Start Code Steps

6 EEE Quick Start Code Steps

The remainder of this application note proceeds through ten steps labeled in the main project function. Each step is followed by a corresponding image capture of the code. Users should use a debugger to execute the code and observe the changes that occur in the debugger memory window. This should help to understand some of the EEE behavior.

1. Assign the previously defined FCLK_DIV value to the flash clock divide register (FCLKDIV). This is necessary before any commands can be issued to the FTM.

2. Use the debugger memory window to see the contents of sections 1 and 2 of the EEE RAM and buffer RAM respectively, before any data is modified.

3. The "LaunchFlashCommand" function was written in this application note to accept parameters and execute FTM commands. For this step, a full partition is performed, which erases and initializes EEE RAM and EEE flash. Afterwards, the debugger memory window (Figure 9) shows all data in EEE RAM (section 1) in an erased state (0xFFFF). Adjacent sections of buffer RAM not managed by the FTM have seemingly random data values. Users should always check the FSTAT and FERSTAT registers for errors after completion of an FTM command.

	/	

																	Auto Global
13FE00	F9	E3	DD	3E	F3	8B	EF	F3	31	EC	B1	35	E3	DB	D3	76	>15v
13FE10	7C	C4	35	D5	F9	9D	Α5	AC	75	DC	7F	D5	A6	F2	6D	BE	.5um.
13FE20	07	FF	31	DE	70	12	D8	47	11	DC	A1	68	61	35	98	01	1.pGha5
13FE30	1E	80	61	3D	E8	31	6C	44	Α5	C4	48	1C	4E	84	1B	DC	a=.11DH.N
13FE40	7E	7F	F6	8B	7E	47	B7	BA	3E	66	FF	0D	26	44	20	FF	~~G>f≨D .
13FE50	F9	FF	AF	FE	0D	C5	D3	94	7B	F6	Α2	27	3B	F9	01	F7	
13FE60	11	Α9	40	97	77	5C	88	54	80	1B	C4	16	72	31	1C	30	@.w\.Tr1.0
13FE70	81	28	20	47	46	FB	15	1E	Α3	Α4	17	8E	87	B8	64	94	.(GFd.
13FE80	4B	7F	9F	D1	5E	F3	6A	CD	BF	7E	21	63	35	Α4	7B	BF	K^.j~!c5.{.
13FE90	F9	E3	ΕE	F7	65	FA	F1	OF	7F	1A	6F	7A	EF	E3	76	F3	eozv.
13FEA0	1A	3B	34	0E	B8	A0	41	49	21	84	64	81	55	91	C5	3E	.;4AI!.d.U>
13FEB0	90	41	A 0	83	42	F5	01	B3	1C	10	28	32	C3	99	00	11	.AB(2
13FEC0	97	59	C1	DF	DE	B5	FF	43	DF	64	6F	BF	84	F7	ΕE	5C	.Y\
13FED0	7F	3F	77	B5	F7	75	CF	B1	14	DF	C5	BD	2F	AF	E8	ED	.?wu/
13FEE0	32	8F	30	8C	4E	DB	51	83	E4	83	35	04	90	4C	29	81	2.0.N.Q5L).
13FEF0	8D	58	Α1	25	38	C4	0A	CD	45	DC	9F	9E	74	09	45	D1	.X.%8Et.E.
13FF00	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	
13FF10	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	
13FF20	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	
13FF30	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	
13FF40	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	
13FF50	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	•••••
13FF60	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	
13FF70	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	•••••
13FF80	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	•••••
13FF90	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	•••••
13FFA0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	•••••
13FFB0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	•••••
13FFC0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	•••••
13FFD0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	•••••
13FFE0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	•••••
13FFF0	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	

Figure 9. Data in EEERAM after Erase

4. Check the partitioning by using a query command. The parameters can be read back from the FCCOBLO register. A match in the parameters ensures the EEE is partitioned as expected.

```
//Use flash command function to query the EEEPROM partitioning.
//Return the status of the FSTAT and FERSTAT Flash registers.
status = LaunchFlashCommand(0 , EEPROM_QUERY, 0, 0, 0, 0, 0, 0, 0, 0, 0); //Check the EEE status
//Check if any error bits are set.
ErrorCheck(status, (accerr|fpviol|mgstat1|mgstat0), (erserif|pgmerif|epviolif|ersvif1|ersvif0|dfdif|sfdif));
FCCOBIX = 1;
                 //Set CCOB index to get Data flash partition result.
dfPart = FCCOBLO; //Copy Data flash partition result.
FCCOBIX++;
                 //Set CCOB index to get EEE RAM partition result.
erPart = FCCOBLO; //Copy EEE RAM partition result.
#ifdef FORCE_PARTITON_FOR_DEBUG //If defined, check that the FTM properly allocated
                               //and erased the EEE sectors in buffer RAM.
//Use query results to determine if D-Flash has been partitioned
//for EEEPROM with erPart == EEE_RAM and dfPart == DFPART.
if (EEE_RAM > 0)
```

NP

EEE Quick Start Code Steps

5. This task illustrates that users can read and write freely to EEE RAM prior to enabling the EEE. The text string "BEFORE" is written at the beginning of the section. For comparison, the neighboring section of buffer RAM is also written with the same string. In this case, the data behaves as volatile data, and a power cycle at this point would result in a loss of all data in the buffer RAM as well as any data in the EEE RAM for which the FTM has not created a record. These strings will be checked again in later steps.

```
/****(5)WRITE PATTERN TO EEE SECTIONS BEFORE EEE ENABLED***/
//Now write the character pattern, "BEFORE", to the beginning of the
//lst and 2nd buffer RAM sections. The 1st 256byte section is partitioned
//as EEE RAM. The 2nd and remaining sections are general user RAM.
section1_ptr = copy_string(&before, section1_ptr);//Copy "BEFORE" to EEE RAM
section2_ptr = copy_string(&before, section2_ptr);//Copy "BEFORE" to user RAM
```

	📟 Mem	ory																_ 🗆 ×
"BEFORE"																		Auto Global
written to	13 1 000	42	45	46	4 F	52	45	EF	F3	33	EC	В1	35	E3	DB	D3	72	BEFORE35r
section 2	13FE10	7C	C4	35	D5	F9	D5	Α5	AC	75	DC	7F	D7	AE	B2	6D	BE	.5um.
00000112	13FE20	07	FF	B1	DE	74	12	D8	47	11	DC	21	68	23	35	D8	09	tG!h#5
	13FE30	16	80	61	2C	E8	21	6C	44	A5	C4	48	1C	CE	84	7B	DC	a,.!1DH{. 🔤
	13FE40	7E	7F	D7	8B	7F	47	B7	BA	2E	6E	FF	8D	06	44	30	FF	~GnD0. 💻
	13FE50	F9	FF	AF	FE	1D	C5	D3	91	7B	F6	Α6	26	3B	F9	01	F7	
	13FE60	15	Α9	40	96	76	54	88	54	80	5B	C4	14	72	31	9C	30	@.vT.T.[r1.0
	13FE70	81	A 8	20	47	46	EΒ	11	1E	A3	Α4	17	8F	83	B8	64	94	GFd.
	13FE80	4B	7F	9F	D1	5E	F3	6B	CD	9F	7E	21	63	3D	Α4	7B	BF	K^.k~!C=.{.
	13FE90	FB	Ε3	ΕE	F7	E5	73	FO	07	7 F	1A	6B	7A	EF	E3	76	E3	skzv.
	13FEA0	1A	3B	34	0E	B8	00	C1	49	21	84	65	91	55	91	C5	3A	.;4I!.e.U:
	13FEB0	90	41	20	8B	42	F5	01	Β7	1C	00	28	12	C3	99	00	51	.A .B(Q
	13FEC0	97	59	C1	DF	FE	Α7	FF	43	DF	64	6B	BF	Α4	F7	ΕA	5C	.YC.dk\
	13FED0	7E	1C	F7	B5	F6	75	CF	В1	14	DF	C5	BC	2F	AF	E8	ED	~u/
"BEFORE"	13FEE0	33	8F	30	9D	4A	FB	41	A3	Α4	83	35	04	90	4C	28	81	3.0.J.A5L(.
written to	13FEF0	8D	59	21	2D	28	C4	0A	CD	55	9C	9F	9E	74	09	45	D1	.Y!-(Ut.E.
	131100	42	45	46	4F	52	45	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	BEFORE
Section 1																		

6. This step enables the EEE. Once enabled, the EEE recognizes that new data, from step 5, has been written to EEE RAM (section 1). The FTM stores records of this new EEE RAM data to the sectors of data flash partitioned as EEE flash. However, the same data written in the adjacent section of buffer RAM (section 2) is not backed up because this section was not partitioned as EEE RAM.

```
/***********(6)ENABLE WRITES OF EEE RECORDS************/
//This enables the FTM to take any revised data written to the EEE
//partitioned section(s) of the buffer RAM and update the record(s)
//in the EEE partitioned section(s) of data Flash.
#ifdef ENABLE_EEE_RECORD_WRITING
if(erPart > 0)
```



7. The following instructions demonstrate that users can continue to read and write freely to the EEE RAM while EEE is enabled. However, the FTM may actively copy data written to the EEE RAM to store and maintain records in EEE Flash. "DURING" is the next string of text written, which is placed behind the previous string. The text strings should be observed in the debugger memory window, as shown in Figure 10.

	📰 Mem	ory																		_ 🗆	×
"DURING"																		Auto		Global	
written to-	13FE00	42	45	46	4F	52	45	20	44	55	52	49	4E	47	₽B	D3	72	BEFORE	DUR	INGr	
section 2	13FE10	7C	C4	35	D5	F9	D5	Α5	AC	75	DC	7F	D7	AE	B2	6D	BE	1.5	.u.	m.	
	13FE20	07	FF	B1	DE	74	12	D8	47	11	DC	21	68	23	35	D8	09	t.	G	h#5	
	13FE30	16	80	61	2C	E8	21	6C	44	A5	C4	48	1C	CE	84	7B	DC	a,.!]	D	i{.	
	13FE40	7E	7F	D7	8B	7F	47	Β7	BA	2E	6E	FF	8D	06	44	30	FF	~G	n.	D0.	
	13FE50	F9	FF	AF	FE	1D	C5	D3	91	7B	F6	Α6	26	3B	F9	01	F7		. {	٤;	
	13FE60	15	Α9	40	96	76	54	88	54	80	5B	C4	14	72	31	9C	30	0.vT.	Т.[.	r1.0	
	13FE70	81	A 8	20	47	46	EB	11	1E	A3	Α4	17	8F	83	B8	64	94	GF		d.	
	13FE80	4B	7F	9F	D1	5E	F3	6B	CD	9F	7E	21	63	3D	Α4	7B	BF	K^.3	c~!	c=.{.	
	13FE90	FB	E3	EE	F7	E5	73	F0	07	7F	1A	6B	7A	EF	E3	76	E3			czv.	
	13FEA0	1A	3B	34	0E	B8	00	C1	49	21	84	65	91	55	91	C5	3A	.;4	I!.e	e.U:	
	13FEB0	90	41	20	8B	42	F5	01	B7	1C	00	28	12	C3	99	00	51	.A .B		(Q	
	13FEC0	97	59	C1	DF	FE	Α7	FF	43	DF	64	6B	BF	Α4	F7	EA	5C	.Y	C.dl	ε\	
	13FED0	7E	1C	F7	B5	F6	75	CF	B1	14	DF	C5	BC	2F	AF	E8	ED	~u		/	
"DURING"	13FEE0	33	8F	30	9D	4A	FB	41	A3	Α4	83	35	04	90	4C	28	81	3.0.J.Z	۱s	5L(.	
written to	13FEF0	8D	59	21	2D	28	C4	0A	CD	55	90	9F	9E	74	09	45	D1	.Y!-(.U.	t.E.	
	131100	42	45	46	41	52	45	20	44	55	52	49	4E	47	FF	FF	FF	BEFORE	DUR	ENG	
Section 1																					

Figure 10. Results of Executing Step 7

8. Next, the EEE will be disabled. However, before disabling the EEE, a status check is performed to ensure that the FTM is not currently busy and also that records for all modified EEE RAM data have been updated. The number of pending records to be stored or updated is tracked in the ETAG register. When ETAG reads 0, all records have been updated.

```
/**********(8)DISABLE WRITES OF EEE RECORDS**********/
if(erPart > 0)
{
    //Wait for FTM to store new EEE RAM data into EEE Flash.
```

Emulated EEPROM Quick Start Guide, Rev. 0



9. Write the last string ("AFTER") to the EEE RAM and buffer RAM. Observe the string in the debugger. The memory window should resemble Figure 11.

	📖 Mem	ory																<u>_ ×</u>
"AFTER"																		Auto Global
written to	10FE00	42	45	46	417	52	45	20	44	-55	52	49	4E	4	20	41	46	BEFORE DURING AF
section 2	13FE10	54	45	52	D5	FB	D5	Α7	EE	75	DC	7 F	F7	FE	F6	6D	FE	TERum.
	13FE20	07	BF	30	DC	60	12	D8	47	01	58	A1	68	21	34	80	01	0.`G.X.h!4
	13FE30	16	80	61	2C	E8	21	48	44	84	C4	40	1C	4E	84	09	DC	a,.!HD@.N
	13FE40	FE	FΒ	F7	8B	7F	47	BF	BA	3E	EF	FF	8F	2E	45	20	FF	G>E . 💻
	13FE50	FD	FF	EF	FE	5F	C5	D7	BD	7 F	FF	A6	67	FF	F9	37	F7	g7.
	13FE60	01	A8	40	94	46	44	88	14	80	ΟA	04	14	12	31	10	30	@.FD1.0
	13FE70	81	80	20	47	46	2B	11	14	A3	A4	17	8E	83	28	04	10	GF+ (
	13FE80	4B	7F	5F	F1	7E	F3	6B	DD	FF	FE	61	63	3D	Α4	FB	FF	K~.kac=
	13FE90	FB	E3	ΕE	F7	Ε5	F3	F1	1F	7 F	3A	7F	FE	EF	E3	F6	F3	
	13FEA0	18	31	34	04	B8	00	41	49	20	80	24	81	51	80	C5	3A	.14AI .\$.Q:
	13FEB0	00	41	AO	83	42	D4	01	93	1C	00	28	12	83	81	00	11	.AB(
	13FEC0	97	D9	C9	FF	FF	B7	FF	4 F	DF	7C	6F	BF	Β4	F7	FE	5D]
	13FED0	FF	FF	FF	Β7	F7	F7	EF	F1	14	FF	D5	BD	2F	AF	E8	ED	/
"AFTER"	13FEE0	32	81	30	8C	48	D8	41	81	AO	83	34	04	10	0C	28	80	2.0.H.A4(.
written to	13FEF0	81	58	01	25	28	C4	0A	C9	05	98	8C	8E	24	01	41	D1	.X.%(\$.A.
	1 OFFOO	42	45	46	47	52	45	20	44	- 55	52	49	4E	4	20	41	46	BEFORE DURING AF
section 1	13FF10	54	45	52	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	TER

Figure 11. Results of Executing Step 9

- 10. This step does not require any execution of code, but observation of the memory window. Thus far, it has been demonstrated that as long as the CPU is active, data can be written in either buffer RAM or EEE RAM sections. The memory window should display these three text strings "BEFORE", "DURING", and "AFTER" in EEE RAM and the adjacent buffer RAM section. The following sub-steps illustrate buffer RAM and EEE RAM differences.
 - a) Use the debugger to perform a reset. After reset, the FTM copies data from records stored in EEE flash back into EEE RAM. At this time, any attempt by the CPU to access EEE RAM before the FTM has completed the copy process will cause the CPU to be stalled until the FTM



EEE Quick Start Code Steps

has finished. The memory window should resemble Figure 12. Notice that the text string "AFTER" has disappeared from the EEE RAM. This is because no record of this string at this address was stored in EEE flash. Instead, the FTM returned all bytes of this address back to an erased state. Conversely, all three strings in the adjacent section of buffer RAM still remain despite the fact that the buffer RAM is volatile memory. This is because the MCU has not been power cycled.

	📖 Mem	ory																
																		Auto Global
	13FE00	42	45	46	4 F	52	45	20	44	55	52	49	4E	47	20	41	46	BEFORE DURING AF
	13FE10	54	45	52	D5	FB	D5	Α7	ΕE	75	DC	7F	F7	FE	F6	6D	FE	TERum.
	13FE20	07	BF	30	DC	60	12	D8	47	01	58	A1	68	21	34	80	01	0.`G.X.h!4
	13FE30	16	80	61	2C	E8	21	48	44	84	C4	40	1C	4E	84	09	DC	a,.!HD@.N
	13FE40	FE	FΒ	F7	8B	7F	47	BF	BA	3E	EF	FF	8F	2E	45	20	FF	E . 💻
	13FE50	FD	FF	EF	FE	5F	C5	D7	BD	7F	FF	Α6	67	FF	F9	37	F7	g7.
	13FE60	01	A8	40	94	46	44	88	14	80	0A	04	14	12	31	10	30	@.FD1.0
	13FE70	81	80	20	47	46	2B	11	14	A3	Α4	17	8E	83	28	04	10	GF+(
	13FE80	4B	7F	5F	F1	7E	F3	6B	DD	FF	FE	61	63	3D	Α4	FB	FF	K~.kac=
"AFTER"	13FE90	FB	E3	ΕE	F7	E5	F3	F1	1F	7F	ЗA	7F	FE	EF	E3	F6	F3	
has been	13FEA0	18	31	34	04	B8	00	41	49	20	80	24	81	51	80	C5	ЗA	.14AI .\$.Q:
nas been	13FEB0	00	41	AO	83	42	D4	01	93	1C	00	28	12	83	81	00	11	.AB(
overwritten	13FEC0	97	D9	C9	FF	FF	Β7	FF	4F	DF	7C	6F	BF	Β4	F7	FE	5D]
by the FTM	13FED0	FF	FF	FF	B7	F7	F7	EF	F1	14	FF	D5	BD	2F	AF	E8	ED	
often need	13FEE0	32	81	30	8C	48	D8	41	81	AO	83	34	04	10	0C	28	80	2.0.H.A4(.
anter reset.	13FEF0	81	58	01	25	28	C4	0A	C9	05	98	8C	8E	24	01	41	<u>D1</u>	.X.≋(\$.A.
-	101700	22	95		9F	52	45	20	99	- 55	52	99	92		FF	FF.	FF	BEFORE DURING
	126610	11	2.2		2.2	2.2	11	11	2.2	11	11	2.2	2.2	11	11	11	11	•••••

Figure 12. EEE RAM and Buffer RAM after Reset

- b) Now exit the debugger application. Power cycle the MCU to force all buffer RAM and EEE RAM to lose data. Reconnect to the MCU with the debugger; this can be done by reprogramming the same code back into the MCU.
- c) Without executing any code, observe the memory window and notice that all three strings have disappeared from the buffer RAM. The absence of these strings illustrates the volatility of the buffer RAM. The EEE RAM, however, still has the "BEFORE" and "DURING" strings since they were copied from EEE flash by the FTM. This feature allows the FTM to emulate EEPROM using flash and RAM resources.

	🛄 Mem	ory																	×
Strings no																		Auto Globa	
longer —	18FE	F8	C3	DD	3F	F5	8B	EF	F1	33	EC	В9	35	41	DB	D3	76	?	
present in	13FE10	70	C4	35	D5 DC	F9	95	23	AE	75	FC 1C	71	D7	FE	B6	6F	FE	.5#.uo.	
buffer RAM	13FE30	16	88	60	3D	E8	11	6C	44	A4	C4	40	18	CC	84	29	DC	`=1D@).	
	13FE40	7E	7 F	F7	AF	7 F	47	BD	BA	3E	67	FF	4D	2E	44	71	BF	~G>g.M.Dq.	
	13FE50	F9	E7	AF	FE	4D	C5	CB	BD	7B	FF	A2	27	3B	F9	8D	F7	M{';	
	13FE70	81	A0	21	47	42	69	15	16	E2	A4	77	8E	83	28	34	90	!GBiw(4.	
	13FE80	5A	7 F	1F	F5	75	F3	68	DD	DE	7E	6B	73	75	Α4	FB	FF	Zu.h~ksu	
	13FE90	FB	E3	EE	F7	ED	F3	F9	0F	3F	1E	6F	FF	EF	E3	F6	C3	?.0	
	13FEB0	90	45	34 A0	04	В4 4В	D5	41	49 B3	10	84 00	28	32	41 C3	90 91	00	36 51	4AI!.d.A6 .EK(20	
	13FEC0	9F	59	Ε1	DF	F7	В7	FF	CF	DF	6C	67	FF	94	E7	FE	5D	.Ylg]	
	13FED0	DE	BF	EF	97	FE	77	CF	F1	94	DF	D1	FD	2E	AF	E8	ED	W	
Strings	13FEE0	12 8D	87 D8	30	25	28	-58 -C0-	-00	-C9-	£4 51	03 90	34 95	92	90 74	4C 09	28 45	80 D1		
remain in 🗕	13F100	42	45	46	4 F	52	45	20	44	55	52	49	4E	47	FF	FF	FF	BEFORE DURING	
EEE RAM	13FF10	FF	FF	FF	F.F.	11	11	FF	FF	F 1	FF	FF	FF	FF	FF	FF	FF		



7 Summary

The EEEPROM feature of the S12X flash module is the newest enhancement allowing users the flexibility to configure the behavior of memory in the EEPROM region. After completing this example, users should understand the following points:

- The location of the EEPROM region and its physical subcomponents: buffer RAM and data flash
- The possible configurations of the EEPROM region
- The FTM command execution flow
- The necessity of always performing error checks after each FTM command execution
- How data behaves when written in buffer RAM and EEE RAM
 - before EEE is enabled
 - while EEE is enabled
 - after EEE is disabled
- That when EEE is partitioned, the CPU is stalled after a reset until the FTM has finished copying data from records in EEE flash back into EEE RAM

Although this application note highlights some of the characteristics of EEE, more details are available. See Freescale application note AN3490, "Overview of the MC9S12XE Emulated EEPROM," for a broader overview of EEE.



THIS PAGE IS INTENTIONALLY BLANK



How to Reach Us:

Home Page: www.freescale.com

Web Support:

http://www.freescale.com/support

USA/Europe or Locations Not Listed:

Freescale Semiconductor, Inc. Technical Information Center, EL516 2100 East Elliot Road Tempe, Arizona 85284 +1-800-521-6274 or +1-480-768-2130 www.freescale.com/support

Europe, Middle East, and Africa:

Freescale Halbleiter Deutschland GmbH Technical Information Center Schatzbogen 7 81829 Muenchen, Germany +44 1296 380 456 (English) +46 8 52200080 (English) +49 89 92103 559 (German) +33 1 69 35 48 48 (French) www.freescale.com/support

Japan:

Freescale Semiconductor Japan Ltd. Headquarters ARCO Tower 15F 1-8-1, Shimo-Meguro, Meguro-ku, Tokyo 153-0064 Japan 0120 191014 or +81 3 5437 9125 support.japan@freescale.com

Asia/Pacific:

Freescale Semiconductor Hong Kong Ltd. Technical Information Center 2 Dai King Street Tai Po Industrial Estate Tai Po, N.T., Hong Kong +800 2666 8080 support.asia@freescale.com

For Literature Requests Only: Freescale Semiconductor Literature Distribution Center P.O. Box 5405 Denver, Colorado 80217 1-800-441-2447 or 303-675-2140 Fax: 303-675-2150 LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN3743 Rev. 0 10/2008 Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-compliant and/or non-Pb-free counterparts. For further information, see http://www.freescale.com or contact your Freescale sales representative.

For information on Freescale's Environmental Products program, go to http://www.freescale.com/epp.

Freescale[™] and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © Freescale Semiconductor, Inc. 2008. All rights reserved.

