Freescale Semiconductor
Application Note

Document Number: AN3822
Rev. 0, 2/2009

Emulated EEPROM Implementation in Dual Flash Architecture on MC9S08LG32

With Demo Description

by: Saurabh Jhamb
Reference Design and Applications Engineering
Microcontroller Solutions Group

Contents

1 Introduction

The flash memory on the MC9S08LG32 is intended primarily for program storage. In-circuit programming allows the operating program to be loaded into the flash memory after final assembly of the application product. It is possible to program the entire array through the single-wire background debug interface. Because no special voltages are needed for flash erase and programming operations, in-application programming is also possible through other software-controlled communication paths. For a more detailed discussion of in-circuit and in-application programming, refer to HCS08RMv1, *HCS08 Family Reference Manual, Volume I*, available at Freescale.com.

This application note explains how to do EEPROM emulation on the MC9S08LG32 using two arrays. This is demonstrated through an example that retrieves data from SCI, stores the data into emulated EEPROM, and shows the EEPROM status and data on the display. Flash memory is intended primarily for program storage.

© Freescale Semiconductor, Inc., 2009. All rights reserved.

In-circuit programming allows the operating program to be loaded into flash memory after the final assembly of the application product. Data storage is becoming common in flash arrays because of new technologies that support longer data retention and higher rates of write cycles.

The MC9S08LG32/16 is a Freescale 8-bit microcontroller that contains two flash arrays. Program and erase operations can be conducted on one array while executing code from the other. This feature allows easy EEPROM emulation while the microcontroller runs. This can improve the layout and eliminate external components, which reduces costs.

The security and protection features in MC9S08LG32 series MCUs treat the two flash arrays as a single memory entity. Programming and erasing of each flash array is conducted through the same command interface, which is detailed in the following sections. It is not possible to page erase or program both arrays at the same time. The mass erase command erases both arrays, and the blank check command checks both arrays.

2 EEPROM Uses and Applications

The requirements for nonvolatile data storage are very common in automotive applications. While program flash can be used to contain data that will not change during the life of a module, EEPROM has traditionally been used to contain data that may change over the life of a module or data that is specific to a particular module. This data might be as simple as an electronic serial number or as extensive as motor positioning data. Data that might potentially be stored in EEPROM would include:

- Odometer value
- Serial number
- Test history and date of manufacture
- Calibration information
- Default application tables
- User configurable data
- Position data
- Encryption keys
- Dynamic network address
- Error code information
- Diagnostic test codes
- Black box recording
- Software feature activation

The flash EEPROM emulation in the MC9S08LG32 family has a number of advantages related to both software and hardware design. Some of these advantages include:

- Fast access to data
- Reduction of printed circuit board components
- Lower microcontroller pin count requirement
- Continued application execution during programming and erase procedures

- Automated program and erase timing

3 Design Considerations for Emulated EEPROM

During the conceptual phase of a design, several approaches can be considered when implementing EEPROM memory storage for data. The use of external EEPROM, emulation of EEPROM with program flash, and on-chip EEPROM are all approaches that have their own advantages and disadvantages in cost and reliability. In addition, each approach will have an impact on the software used to manage the data.

In an MC9S08LG32, the strategy is to use a portion of flash program memory to emulate EEPROM. There are two common approaches when doing this.

The first approach is to keep a copy of the EEPROM data in a RAM buffer and periodically write the entire contents of the buffer to the program flash. This approach is relatively simple to implement, permits the data to be read from the RAM buffer at any time, and allows control of the number of program/erase cycles. The obvious drawback to this approach is the amount of RAM that must be dedicated to the emulated EEPROM buffer. In addition, there is a risk of losing data if a reset occurs after updating a RAM buffer, but before the data has been programmed into flash.

A second approach uses multiple sectors of program flash to store nonvolatile data using a flash file system. Depending on the implementation, this method will usually require less RAM than the first approach. The major disadvantage to this approach is the size and complexity of the firmware required to implement a robust flash file system, such that it minimizes the risk of losing data when an unexpected reset or loss of power occurs.

Both of these approaches can require multiple program and erase operations when changing a single piece of data. This can lead to a significant increase in the amount of bulk capacitance required on the microcontroller's power supply lines, to guarantee completion of all program and erase operations in the event of loss of power.

One disadvantage of EEPROM emulation is the fact that, because an application's interrupt vectors are located in flash, all interrupts must be masked during emulated EEPROM operations. Depending on an application's interrupt latency requirements, this may impose some limitations on the system.

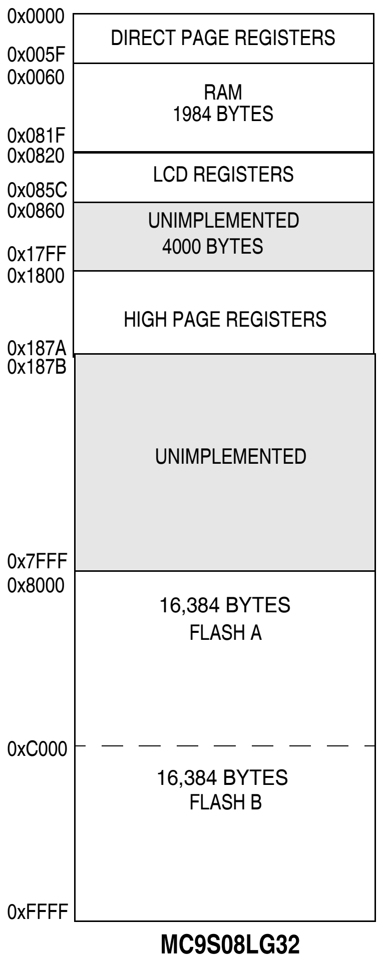
4 MC9S08LG32/16 Flash Memory Characteristics

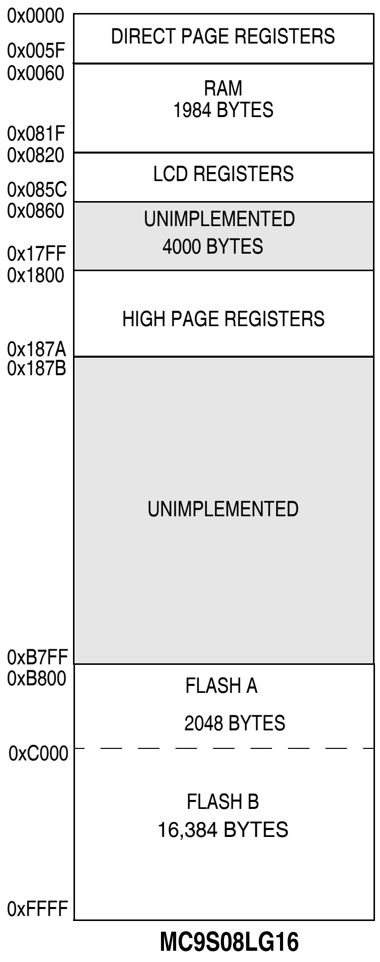
Features of the flash memory include:

- Flash size
  - — MC9S08LG32: 32,768 bytes (16,384 bytes in Flash A, 16,384 bytes in Flash B)
  - — MC9S08LG16: 18,432 bytes (2,048 bytes in Flash A, 16,384 in Flash B)
- Single power supply program and erase
- Command interface for fast program and erase operation
- Up to 100,000 program/erase cycles at typical voltage and temperature
- Flexible block protection
- Security feature for flash and RAM

- Auto power-down for low-frequency read accesses minimizes run I_{DD}
- Flash read/program/erase functions work properly over full operating voltage or temperature

5 Memory Map





6 Flash Programming

This flash memory module includes integrated program/erase-voltage generators and separate command-processor state machines that can perform automated byte programming page (512 bytes flash) or mass erase, and blank check commands. Commands are written to the command interface. Status flags report errors, and indicate when commands are completed. The block protection feature prevents the protected region of flash from accidental program or erase operations.

A security mechanism can be engaged to prevent unauthorized access to the flash and RAM memory contents. An optional user-controlled, back-door key mechanism can be used to allow controlled access to secure memory contents for development purposes.

6.1 Program and Erase Times

One advantage of emulated EEPROM is that program and erase times are very fast compared with other technologies.

Table 1 shows program and erase times. The times include overhead for the command-state machine, and enabling and disabling of program and erase voltages.

Table 1. Program and Erase Times

Parameter Cycles	FCLK Time	If FCLK = 200 KHz
Byte program	9	45 µs
Byte program (burst)	4	20 µs (excluding start/end overhead)
Page erase	4000	20 ms
Mass erase	20,000	100 ms

6.1.1 Flash Clock

Before any program or erase command can be accepted, the flash clock divider register (FCDIV) must be written to set the internal clock for the flash module to a frequency (fFCLK) between 150 kHz and 200 kHz.

6.1.2 Flash Clock Divider Register (FCDIV)

This register can be written only once, so normally this write is done during reset initialization. FCDIV cannot be written if the access error flag, FACCERR in FSTAT, is set. You must ensure that FACCERR is not set before writing to the FCDIV register. One period of the resulting clock (1/fFCLK) is used by the command processor to time the program and erase pulses. To complete a program or erase command, the command processor uses an integer value of these timing pulses. Bit 7 of this register is a read-only status flag. Bits 6 through 0 may be read at any time but can be written one time only. Before any erase or programming operations are possible, write to this register to set the clock frequency for the nonvolatile memory system within acceptable limits.

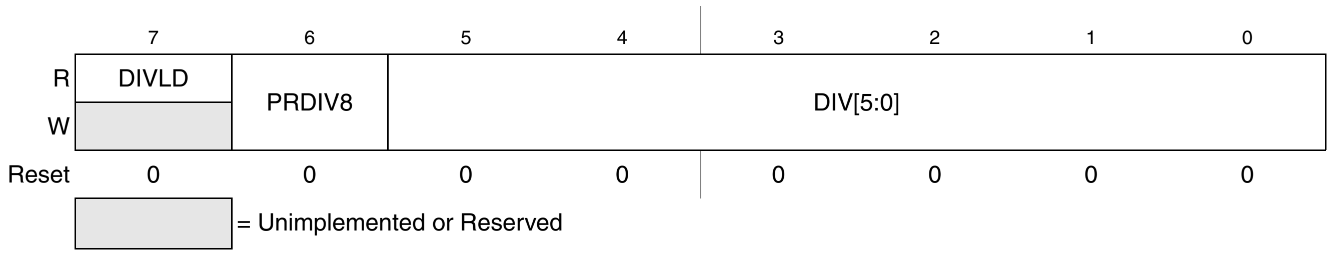


Figure 1. Flash Clock Divider Register (FCDIV)

Table 2. FCDIV Register Field Descriptions

Field	Description
7 DIVLD	**Divisor Loaded Status Flag** — When set, this read-only status flag indicates that the FCDIV register has been written since reset. Reset clears this bit, and the first write to this register causes this bit to become set regardless of the data written. 0 FCDIV has not been written since reset; erase and program operations disabled for flash. 1 FCDIV has been written since reset; erase and program operations enabled for flash.
6 PRDIV8	**Prescale (Divide) Flash Clock by 8** 0 Clock input to the flash clock divider is the bus rate clock. 1 Clock input to the flash clock divider is the bus rate clock divided by 8.
5:0 DIV[5:0]	**Divisor for Flash Clock Divider** — The flash-clock divider divides the bus-rate clock (or the bus-rate clock divided by 8 if PRDIV8 = 1) by the value in the 6-bit DIV5:DIV0 field, plus one. The resulting frequency of the internal flash clock must fall within 200 kHz to 150 kHz for proper flash operations. See Equation 1 and Equation 2. For more details, see Freescale document AN3824, "The EEPROM Emulation Driver for MC9S08LG32."

If PRDIV8 = 0:

$$f_{FCLK} = f_{Bus} \div (DIV + 1) \qquad \textit{Eqn. 1}$$

If PRDIV8 = 1:

$$f_{FCLK} = f_{Bus} \div (8 \times (DIV + 1)) \qquad \textit{Eqn. 2}$$

Table 6-3. Flash Clock Divider Settings

f_{Bus}	PRDIV8 (Binary)	DIV (Decimal)	f_{FCLK}	Program/Erase Timing Pulse (5 μs Min, 6.7 μs Max)
20 MHz	1	12	192.3 kHz	5.2 μs
10 MHz	0	49	200 kHz	5 μs
8 MHz	0	39	200 kHz	5 μs
4 MHz	0	19	200 kHz	5 μs
2 MHz	0	9	200 kHz	5 μs
1 MHz	0	4	200 kHz	5 μs
200 kHz	0	0	200 kHz	5 μs
150 kHz	0	0	150 kHz	6.7 μs

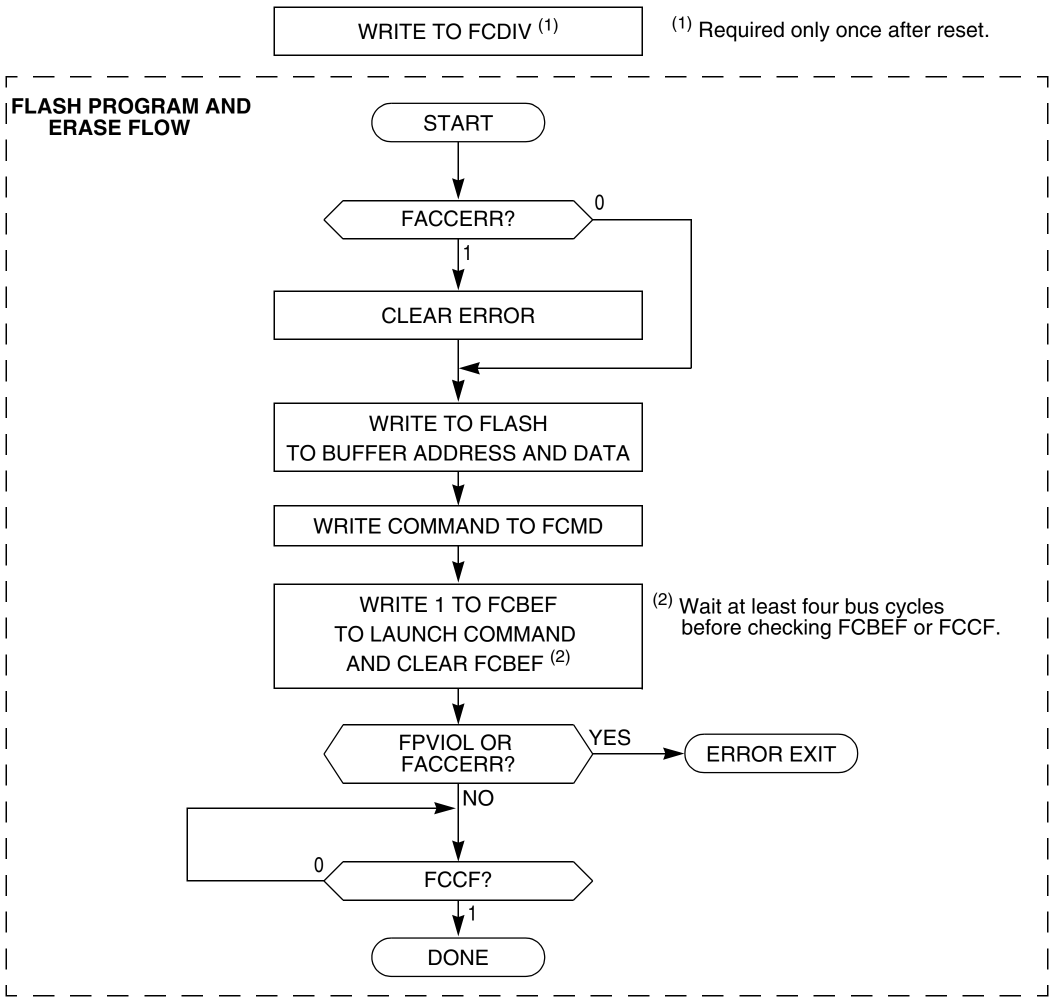
6.2 Program and Erase Flash Algorithms

All the program and erase algorithms, such as turn on/off charge pumps, control times, etc., are done by a hardware state machine that takes care of all processes without interfering in the microcontroller functionality. You must determine where the data is stored and then launch the program sequence. The details of using these algorithms are explained in Section 6.2.1, "Program and Erase Command Execution."

6.2.1 Program and Erase Command Execution

The steps for executing any of the commands are listed below. The FCDIV register must be initialized and any error flags must be cleared before beginning command execution. The command execution steps are:

1. Write a data value to an address in the flash array.
   a) The address and data information from this write is latched into the flash interface. This write is a required first step in any command sequence. For erase and blank check commands, the value of the data is not important. For page erase commands, the address may be any address in the 512-byte page of flash to be erased. For mass erase and blank check commands, the address can be any address in the flash memory. Whole pages of 512 bytes are the smallest block of flash that may be erased.
   b) Do not program any byte in the flash more than once after a successful erase operation. Reprogramming bits to a byte that is already programmed is not allowed without first erasing the page in which the byte resides or else mass erasing the entire flash memory. Programming without first erasing may disturb data stored in the flash.
2. Write the command code for the desired command to FCMD. The five valid commands are blank check (0x05), byte program (0x20), burst program (0x25), page erase (0x40), and mass erase (0x41). The command code is latched into the command buffer.
3. Write 1 to the FCBEF bit in FSTAT to clear FCBEF and launch the command (including its address and data information).

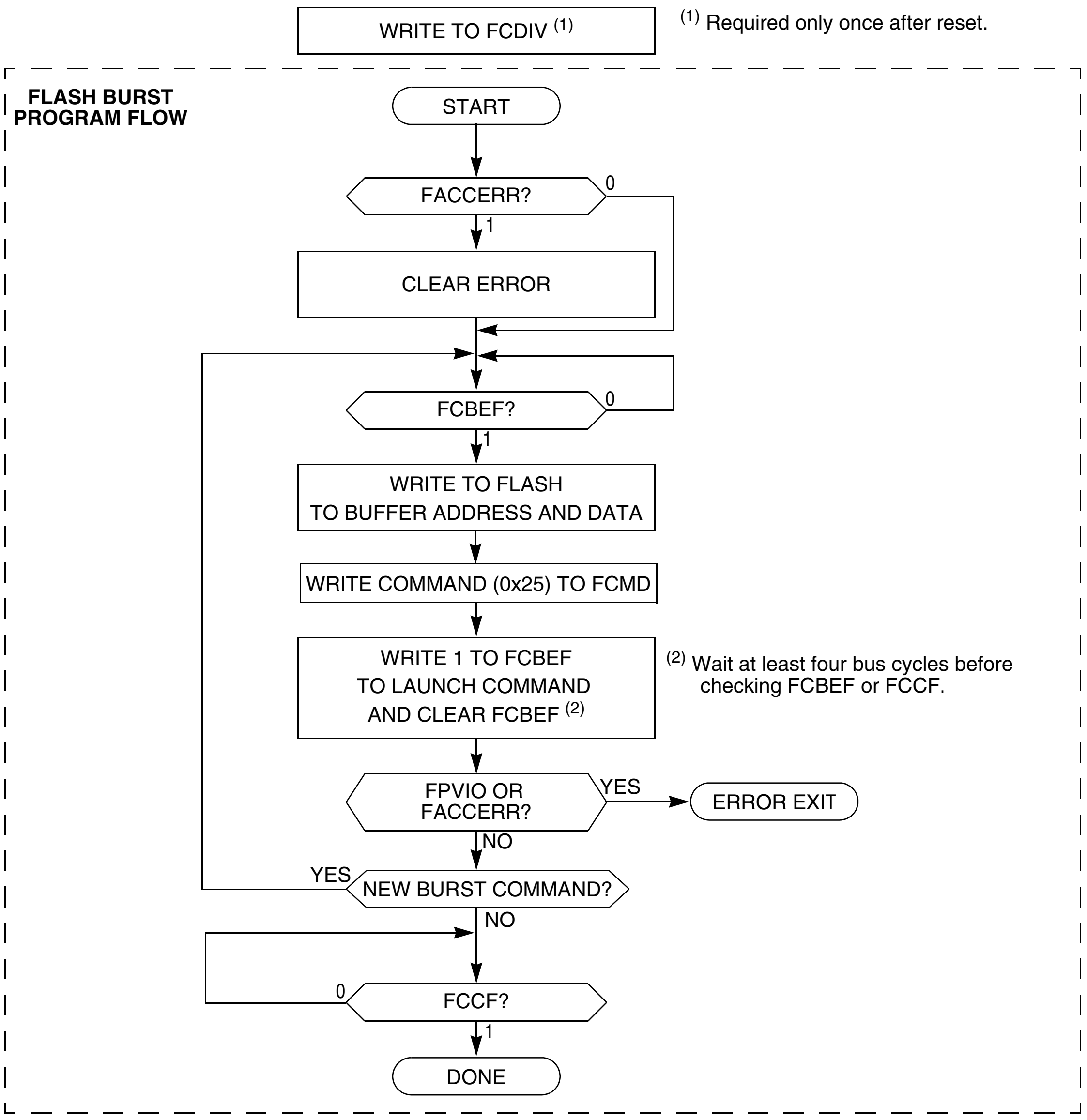

WRITE TO FCDIV (1)
(1) Required only once after reset.
FLASH PROGRAM AND ERASE FLOW
START
FACCERR?
0
1
CLEAR ERROR
WRITE TO FLASH TO BUFFER ADDRESS AND DATA
WRITE COMMAND TO FCMD
WRITE 1 TO FCBEF TO LAUNCH COMMAND AND CLEAR FCBEF (2)
(2) Wait at least four bus cycles before checking FCBEF or FCCF.
FPVIOL OR FACCERR?
YES
ERROR EXIT
NO
0
FCCF?
1
DONE

6.3 Burst Program Execution

The burst program command is able to program sequential bytes of data in less time than would be required using the standard program command. This is possible because the high voltage to the flash array does not need to be disabled between program operations. Ordinarily, when a program or erase command is issued, an internal charge pump associated with the flash memory must be enabled to supply high voltage to the array. Upon completion of the command, the charge pump is turned off. When a burst program command is issued, the charge pump is enabled and then remains enabled after completion of the burst program operation, if these two conditions are met:

- The next burst program command has been queued before the current program operation has completed.
- The next sequential address selects a byte on the same physical row as the current byte being programmed. A row of flash memory consists of 64 bytes. A byte within a row is selected by addresses A5 through A0. A new row begins when addresses A5 through A0 are all zero.

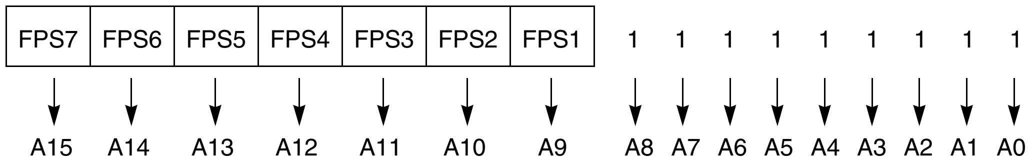
The first byte of a series of sequential bytes being programmed in burst mode takes the same amount of time to program as a byte programmed in standard mode. The subsequent bytes are programmed in the burst program time if the conditions above are met. In a case where the next sequential address is the beginning of a new row, the program time for that byte is the standard time instead of the burst time. This is because the high voltage to the array must be disabled and then enabled again. If a new burst command has not been queued before the current command completes, then the charge pump is disabled and the high voltage is removed from the array.


WRITE TO FCDIV (1)
(1) Required only once after reset.
FLASH BURST PROGRAM FLOW
START
FACCERR?
0
1
CLEAR ERROR
FCBEF?
0
1
WRITE TO FLASH TO BUFFER ADDRESS AND DATA
WRITE COMMAND (0x25) TO FCMD
WRITE 1 TO FCBEF TO LAUNCH COMMAND AND CLEAR FCBEF (2)
(2) Wait at least four bus cycles before checking FCBEF or FCCF.
FPVIO OR FACCERR?
YES
ERROR EXIT
NO
YES
NEW BURST COMMAND?
NO
0
FCCF?
1
DONE

6.4 Flash Block Protection

The block protection feature prevents program and erase commands from being written to the protected region of flash. Block protection is controlled through the flash protection register (FPROT). When enabled, block protection begins at any 512-byte boundary below the last address of flash, 0xFFFF.

After exit from reset, FPROT is loaded with the contents of the NVPROT location, which is in the nonvolatile register block of the flash memory. FPROT cannot be changed directly from application software to prevent runaway programs from altering the block protection settings. Because NVPROT is within the last 512 bytes of flash, if any amount of memory is protected, NVPROT is itself protected and cannot be altered (intentionally or unintentionally) by the application software. FPROT can be written through background debug commands, which allows a protected flash memory to be erased and reprogrammed.



The FPS bits are used as the upper bits of the last address of unprotected memory. This address is formed by concatenating FPS7:FPS1 with logic 1 bits as shown. For example, to protect the last 1536 bytes of memory (addresses 0xFA00 through 0xFFFF), the FPS bits must be set to 1111 100, which results in the value 0xF9FF as the last address of unprotected memory. In addition to programming the FPS bits to the appropriate value, FPDIS (bit 0 of NVPROT) must be programmed to logic 0 to enable block protection. Therefore, the value 0xF8 must be programmed into NVPROT to protect addresses 0xFA00 through 0xFFFF.

7 EEPROM Emulated Demo

After understanding how the program-flash and erase-flash commands work, we apply these concepts to EEPROM emulation.

This demonstration includes checking basic EEPROM functionality, such as writing and reading the data based on record IDs. It reads the existing records of data and displays them though SCI.

7.1 Configuration

- Sector size: 512 bytes
- Data ID size: 1 byte
- Data size: 1 byte
- EEPROM size: 254 bytes
- FLASH BLOCK A (0x8000–0xBFFF) used for EEPROM data
- EEPROM start address: 0x8000
- Callback function to reset the COP
- Code runs from FLASH BLOCK B(0xC000–0xFFFF)

7.2 Function

Following are the steps in which the EEPROM demo executes:

1. Initializes EEPROM by calling “FSL_InitEeprom”.
2. Checks whether any data records for ID 1 and 2 exist or not.
3. If data record exists, then it displays the data through SCI port — else goes to step 4.

4. Erases the existing records of data.
5. Writes data (previous value +1) for records for ID 1 and 2 by calling “FSL_WriteEeprom”.
6. End.

8 Conclusion

The MC9S08LG32/16 has many features (LCD controller, IIC, SCI, A/D, SPI, etc.). The feature explored here was the flash memory with two arrays. When implementing EEPROM emulations, you can program and erase the flash memory without stopping the application. This is because of a state machine and two arrays of flash already implemented in the MCUs.

9 References

See S08LG Product Summary Page for more information and the documents released for MC9S08LG32.

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 China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
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: AN3822
Rev. 0
2/2009

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. 2009. All rights reserved.