Document Number: AN3824
Rev. 0, 2/2009

The EEPROM Emulation Driver for MC9S08LG32

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

1 Introduction

The MC9S08LG32 is a member of the Freescale HCS08 family MCUs. It uses the S08 core and integrates many peripherals, such as LCD, SPI, IIC, SCI, and ADC. This document is the note of the EEPROM emulation driver for the MC9S08LG32 microcontroller family.

The electrically erasable programmable read only memory (EEPROM) can be byte, word programmed, or erased and is often used in automotive electronic control units (ECUs). This flexibility for program and erase operations make it suitable for data storage of application variables that must be maintained when power is removed, and need to be updated individually during run-time. For devices without EEPROM memory, the page-erasable flash memory can be used to emulate EEPROM through EEPROM emulation software.

Contents

1 Introduction 1
2 System Architecture 2
2.1 EEPROM Emulation Memory Layout 2
2.1.1 EEPROM Sectors 2
2.1.2 EEPROM Sector Scheduling 3
2.1.3 EEPROM Data Organization 3
2.1.4 Data Update and Status Accounting 4
2.1.5 Sector Status Accounting 4
2.2 Cache Table Configuration 5
2.3 Callback Notification 5
2.4 Return Codes 5
2.5 Macros Used 6
3 Functions and Calling Conventions 6
3.1 High Level APIs (User Level APIs) 6
3.2 Middle Level APIs 7
3.3 Low Level APIs 7
4 API Description 7
4.1 FSL_InitEeprom 7
4.2 FSL_ReadEeprom 8
4.3 FSL_WriteEeprom 8
4.4 FSL_ReportEepromStatus 8
4.5 FSL_DeinitEeprom 8
4.6 Assumptions 8
5 References 8

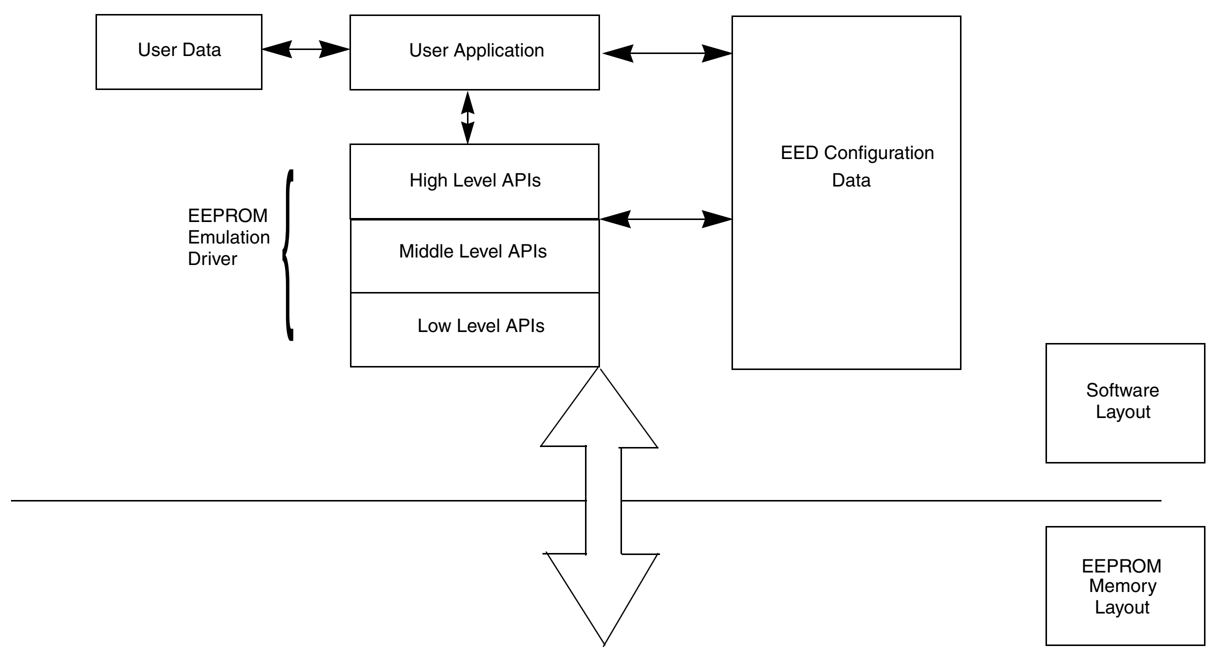
The EEPROM emulation driver for the MC9S08LG32 implements the fixed-length data record emulation on the available MCU flash. The emulated EEPROM functions include:

- Organizing data records
- Initializing and de-initializing EEPROM
- Reporting EEPROM status reading
- Writing
- Deleting data records

2 System Architecture

The EEPROM emulation driver has three level APIs; high level, middle level, and low level.

- High level (user level) APIs provide the user's interface and program flow controlling.
- Middle level APIs provide the relative independent task unit.
- Low level APIs use the standard software driver to provide the fundamental Flash operations.



2.1 EEPROM Emulation Memory Layout

2.1.1 EEPROM Sectors

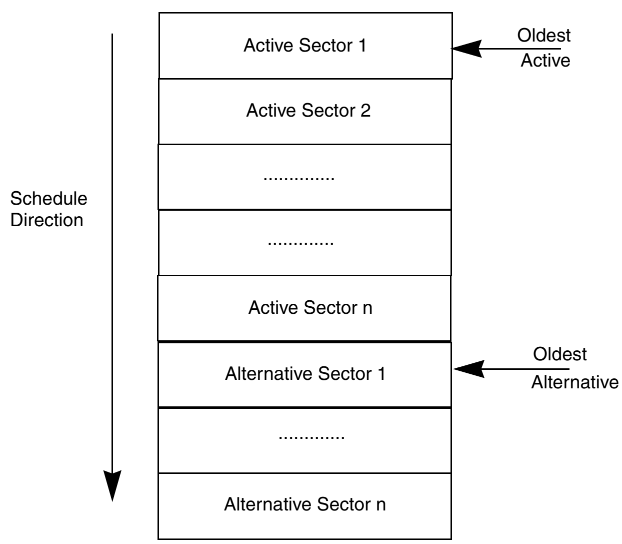
The EEPROM emulation driver adopts the HCS08 family flash to emulate as EEPROM. A minimum of two sectors; active, and alternative sectors are needed to emulate EEPROM. There can be more than one active and alternative sectors used for emulation. The number of sectors used for active and alternative sector set can be different. The number of sectors used as active sector is decided by the number of bytes wanting to store in the EEPROM. The number of alternative sectors can be user configurable. At least one alternative sector is necessary for the emulation scheme to work. Many alternative sectors can be allocated, depending on the availability of the memory and the requirement of the application.

Table 1. EEPROM Sector Macros

Macro Name	Description
EED_SECTOR_SIZE	Size of sector
EED_SECTOR_CAPACITY	Number of data records that can be stored in a sector
EED_SECTOR_NUMBER	Number of active sectors that are used for emulation
EED_READY_SECTORS	Number of alternative sectors that are used for emulation
EED_SECTORS_ALLOTED	Total number of flash sectors used for emulation EED_SECTORS_ALLOTED = EED_READY_SECTORS + EED_SECTOR_NUMBER)

2.1.2 EEPROM Sector Scheduling

All sectors in the Active sector set must be marked active. These sectors must be scheduled in a round-robin scheme. If one sector gets filled up, the consecutive active sector must be used to store the EEPROM data. If there are no active sectors available, the data from the oldest active sector must be compressed to the oldest alternative sector that should now be used as an active sector. The oldest active sector must be erased and declared as an alternative.



2.1.3 EEPROM Data Organization

Each emulation sector contains:

- Sector Status Field — Stores the sector status. The actual status depends on a combination of the status byte, erased cycles, and the first data record ID byte.
- Erasing Cycles — Store the sector's erasing cycles, because the EEPROM emulation is set up. It increments after each erase.

- Data Records Field — Each data record has three fields:
  - Data Record Status Field — The data record status
  - Data Record ID — The data record identifier
  - Data — User's raw data
- Blank Field — Used for storing new data records.

Table 2. EEPROM Data Macros

Macro Name	Description
DATA_STATUS_SIZE	Data status size has 1 byte fixed length.
DATA_ID_SIZE	Data ID size is user configurable.
DATA_SIZE	Data size is user configurable.
EEPROM_SIZE	EEPROM size is user configurable.
DATA_RECORD_NUMBER	Number of data records is calculated if the total data size is known.
DATA_RECORD_SIZE	Data record size can be calculated as sum of status, ID, and size.

NOTE

DATA_RECORD_SIZE cannot exceed 127 (0x7F) because the AIX instruction does not support more than 127(0x7F).

2.1.4 Data Update and Status Accounting

The data record cannot be updated directly in the same location. Instead, a new data record with the new value is written to the EEPROM. The read routine reads the latest ID occurrence in the active sectors.

When updating data, the status field, the ID, and bytes all get updated. The order of an update is as follows:

1. Program data ID field
2. Program data field
3. Program data status field

Table 3. Data Update and Status Accounting

Data Status Field	Data ID Field	Status of Data Record
$FF	$FF	Record in the erased state.
$FF	XX	Data record is under update. Further writes are possible only in the next record location.
XX	XX	Data record contains valid data.

2.1.5 Sector Status Accounting

The status byte of the sector is a single byte field. It can hold only two values, 0xFF or any value other than 0xFF. The status of the sector is determined by a combination of the sector status field, the sector erase cycle field, and the sector's first data ID field as shown in Table 4.

Table 4. Sector Status Accounting

Sector Status Field	Sector Erase Cycles	First Data ID Field	Status of Sector
$FF	$FFFF	$FF	This sector is blank.
$FF	XXXX	$FF	This sector is an alternative sector.
$FF	XXXX	XX	This sector is under update. Specifically, it is in the process of sector compression.
XX	XXXX	XX	This is the active sector. It takes the new data to be stored in EEPROM.

2.2 Cache Table Configuration

A cache table that holds the address of the latest occurrence of the most frequently used data IDs is used to speed up data reading and sector compression.

The cache table defines the start address of the cache. You can configure the table size using macro, EED_CACHETABLE_ENTRY. It is recommended to not configure a large size for the cache table.

This table must hold the address of IDs starting from 0 to (EED_CACHETABLE_ENTRY – 1). It is the user's responsibility to use these IDs to store the most commonly or frequently used data records.

2.3 Callback Notification

The EEPROM emulation driver enables supplying a pointer to the CallBack() function therefore time-critical events can be serviced during EEPROM operations. Servicing watchdog timers is one of the time critical events. If it is not necessary to provide the CallBack() service, it can be disabled by a NULL function macro.

```
NULL_CALLBACK equ $FFFF
```

The job processing callback notifications must have no parameters and no return value. If a job processing callback notification is configured as NULL_CALLBACK, the corresponding callback routine must not be called.

2.4 Return Codes

Table 5 shows the return codes that must be used.

Table 5. Return Codes

Name	Value	Description
EE_OK	0x00	The requested operation was successful.
EE_ERROR_ACCERR	0x10	Access error flag is set while operating the Flash.
EE_ERROR_PVIOL	0x20	Protection violation flag is set while operating the Flash.
EE_ERROR_NOT_BLANK	0x30	The flash memories are not blank.
EE_ERROR_SECURITY_ENABLED	0x40	The part is secured.

Table 5. Return Codes (continued)

Name	Value	Description
EE_ERROR_VERIFY	0x50	Corresponding source data and content of destination location mismatch.
EE_ERROR_NOMEM	0x60	Not enough EEPROM memory.
EE_ERROR_NOFND	0x70	Record not found in sector.
EE_ERROR_CLOCK_SETTING	0x80	The FLASH clock has already been initialized and the new clock divider does not match the value in FCDIV register.
EE_ERROR_SSTAT	0x90	Sector status error.
EE_ERROR_IDRNG	0xA0	Record identifier exceeds the valid range.

2.5 Macros Used

Table 6. Macros

Name	Value	Description
EE_SECTOR_ACTIVE	0x00	Sector status is active.
EE_SECTOR_ALTERNATIVE	0x55	Sector status is alternative.
EE_SECTOR_BLANK	0xFF	Sector status is blank.
EE_SECTOR_UPDATE	0xAA	Sector status is partially updated.
EFLASH_START_ADDRESS	0x40	Starting address of the flash allocated for emulation of EEPROM.
EFLASH_END_ADDRESS	0x50	End address of flash allocated for emulation of EEPROM.
EED_CACHETABLE_ENTRY	0x08	Number of entries the cache table holds. This also represents the maximum record ID the cache table holds.

3 Functions and Calling Conventions

The EEPROM emulation driver (EED) provides three hierarchies of application programming interfaces (APIs):

- High level
- Middle level
- Low level APIs

3.1 High Level APIs (User Level APIs)

These APIs provide direct operations on the emulated EEPROM such as, initialize EEPROM, read record, write record, delete data record, report EEPROM status, and de-initialize EEPROM.

- FSL_InitEeprom — Initializes the flash memory used for EEPROM emulation.
- FSL_ReadEeprom — Read the specific data record from emulated EEPROM.

- FSL_WriteEeprom — Write a data record to emulated EEPROM.
- FSL_ReportEepromStatus — Report the status of the emulated EEPROM.
- FSL_DeinitEeprom — De-initialize the flash memory used for EEPROM emulation.

3.2 Middle Level APIs

These APIs provide individual functionality to support the high level APIs for operating the emulated EEPROM

- FSL_Erase — Erase the Flash pages.
- FSL_Program — Program the data into the flash memory.
- FSL_CopyRecord — Copy one data record to the flash memory.
- FSL_InitSector — Initialize one sector including erase, blank check, and update the erased cycles field of this sector.
- FSL_SwapSector — Copy the latest data records from the oldest active sector to the oldest alternative sector.
- FSL_SearchRecord — Search the required data record ID in a sector.
- DoHVCopyDown — Copy code necessary to initiate high voltage operation into RAM from flash.
- FSL_SectorStatus — Return the status of the sector.
- FSL_GetAddr — Stores the start address of the sector and calculates the end address of the sector.
- FSL_AddSectorSize — Adds the sector size to the contents of the HX register.
- FSL_SubSectorSize — Subtracts the sector size from the contents of the HX register.
- FSL_SearchLoop — Loops across all the active sectors to search for the record ID.

3.3 Low Level APIs

These APIs are basic flash operations:

- FlashErase — Erase continuous flash logical pages.
- FlashProgram — Program data into data flash.
- DataVerify — Depending on an input parameter, verify the content of the destination with the source or verify if the destination is blank.
- HighVoltage — This function launches the flash command written into the flash command register by FlashErase or FlashProgram function and waits until the command finishes. This is an internal function that should only be called by low level functions such as, FlashErase and FlashProgram.
- FlashInit — This function is used to set the clock divider during the initialization of flash for the EEPROM emulation.

4 API Description

4.1 FSL_InitEeprom

unsigned char FSL_InitEeprom (void)

Description — Performs the flash module clock initialization. This function determines active, alternative and brown out affected sectors, and erases or updates sectors. Initializing variables that hold active sector related information, for example the start and end addresses of the active sector. The blank space available is also in this function. The cache table is initialized in this function. If no sectors are initialized, this function should initialize all the sectors in a round-robin queue.

4.2 FSL_ReadEeprom

unsigned char FSL_ReadEeprom (void)

Description — This function is to read the specific data record. The starting address of the record data is returned.

4.3 FSL_WriteEeprom

unsigned char FSL_WriteEeprom (void)

Description — This function encapsulates data in a record and writes it to the emulated EEPROM. If there is not enough free space in active sector, this routine must check if the next sector available is an active sector. Otherwise, this routine initiates sector swapping to clean up the EEPROM.

4.4 FSL_ReportEepromStatus

unsigned char FSL_ReportEepromStatus (void)

Description — This function reports statistics, for example active emulation sector erasable cycles and checks the emulation sector status.

4.5 FSL_DeinitEeprom

unsigned char FSL_DeinitEeprom (void)

Description — This function releases all flash used for EEPROM emulation. After de-initialized, the flash pages for emulation are fully erased.

4.6 Assumptions

The descriptions in this document assumes the person reading it has full knowledge of the configuration registers of all blocks in MC9S08LG32, especially Flash Security.

5 References

See S08LG Product Summary Page for more information and the documents released for the 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: AN3824
Rev. 0
2/2009