Freescale Semiconductor
Application Note

Document Number: AN3490
Rev. 0, 09/2007

Overview of the MC9S12XE Emulated EEPROM

by: Martyn Gallop
East Kilbride, Scotland

1 Introduction

The MC9S12XEP100 is the first member of the S12X microcontroller family in a low-power 0.18µ technology. This document introduces the S12XE's support for a new, highly flexible, and scalable emulated EEPROM (EEE) architecture.

Due to the high level of integration provided by the FTM memory controller, accessing and updating data in the EEE is similar to using an NVRAM. After it is configured, an application may access dynamic NVM data as if it is in a simple RAM, but one where the contents are retained during power down.

1.1 Objective

The objective of this document is to detail how memory controller commands can be used to configure and use the EEE. An overview of the EEE implementation and some application considerations are also included.

Contents

1.2 Overview

The S12XE EEE can reduce much of the overhead of typical software EEPROM management.

Features include

- A dedicated RAM buffer that is automatically initialized at reset and that the application sees as a virtual NVRAM—no need for a separate RAM mirror of EEPROM data
- Automated programming of NVM when the buffer RAM contents are modified
- Improved write/erase performance (over past EEPROM technologies)—achieved by higher NVM to data ratios and inherent wear leveling
- Hardware ECC support of NVM data

Like a typical software driver, the EEE allows the application

- Read/write the EEE data at any time
- Halt programming operations
  - Pending EEE writes can be paused
  - The memory controller can be brought into an idle state
- Determine if any programming operations are still pending

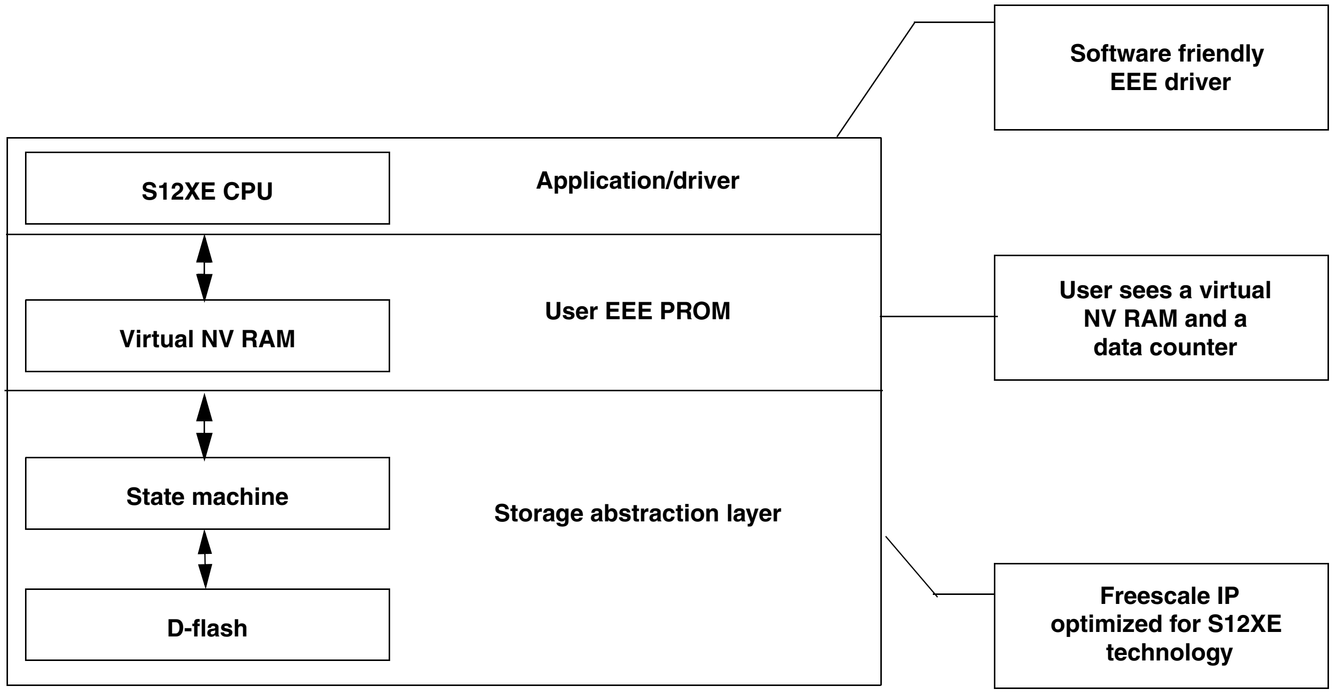

Figure 1. EEE Top Level

As a member of the 16-bit S12X family, the EEE is designed for optimal use with aligned word (16-bit) data—on previous S12X parts, the EEPROM is also aligned-word based. Both the RAM and the flash EEE resources are word organized.

Byte wide data is also managed seamlessly by the EEE. Byte data has the same NVM requirement as word-aligned data so is not quite as efficient a use of the EEE NVM storage.

Misaligned word data is also managed cleanly by the EEE; however, it is not recommended as it results in increased EEE NVM cycling and there is a risk of data corruption when a device is reset or powered down before all pending EEE data is programmed to the EEE NVM. This is a consideration also common to existing byte- and word-oriented EEPROM solutions (see Section 6.1, “Data Coherency Across RAM Word Boundaries”).

1.2.1 EEE Data Frequency

The S12XE EEE scheme is well suited to moderate to high update cycling of data that changes often. While easily managed by the EEE, constant data (such as data that is never or rarely updated) presents some additional considerations and may be more suited to being programmed directly into flash memory during module production or by the application—this is explained in further detail later on.

To optimize support for different data requirements, the EEE resources can be configured to most closely fit the data requirements of an individual application.

NOTE

Data that does not change or rarely changes in value is described as constant data in this document. This must not be confused with constant data types as defined in high-level programming languages (although constant type data elements placed into the EEE will generate constant EEE data).

1.2.2 Resources

The EEE sub-system resides within the flash memory (FTM) module. The key resources are the buffer RAM and the D-flash (data flash) block.

The EEE resources are able to be configured once, typically during ECU module production (in special modes the EEE can be repartitioned for application development). Buffer RAM and D-flash resources not allocated for EEE continue to be individually accessible by application software. Any D-flash allocated for EEE (the EEE NVM partition) is automatically protected from unintentional corruption by an application.

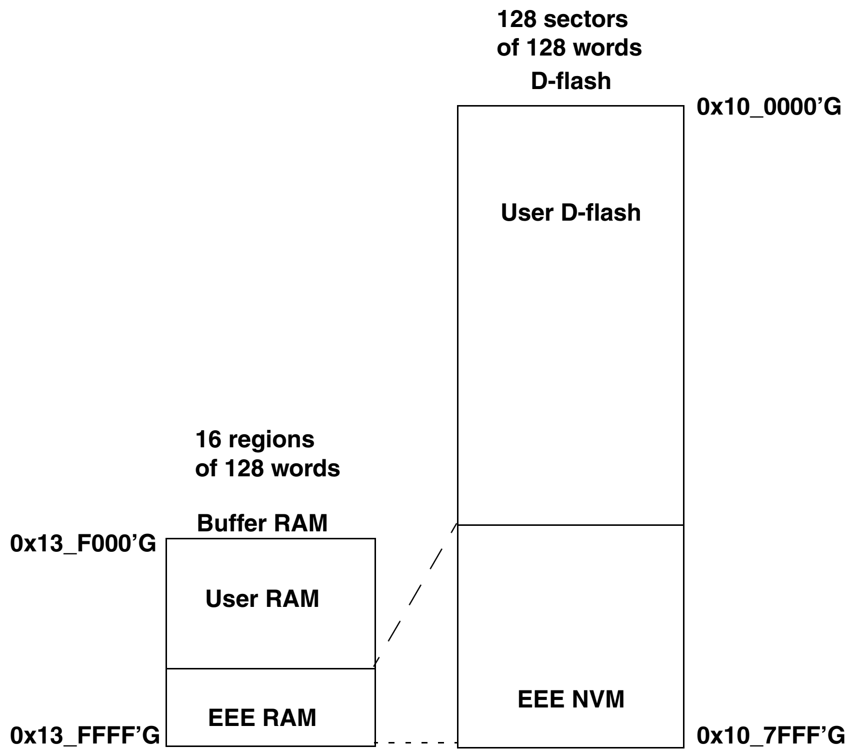

Figure 2. S12XEP 100 EEE Memory Resources

EEE resources can be

- Used individually by an application as SRAM and small sector flash memory (User RAM and User D-flash)
- Used as EEE supported by the memory controller (EEE RAM and EEE NVM)
- A combination of the above

The memory controller coordinates all of the NVM and EEE activities.

1.3 Address Notation

All address references in this document use the S12X global address map (indicated by the G suffix). This is a 23-bit linear address format.

1.4 The FTM Memory Controller

The FTM is the module on the S12XE containing the P-flash memory block(s), a D-flash memory block, and a memory controller state machine to manage them. The FTM can be considered as having three independent application interfaces each with its own feedback mechanism (i.e. flags)

Common command object (CCOB) commands

- For programming / erasing the P-flash and D-flash and configuring the EEE
- Parameters are located in the CCOB registers
- The FTM reset sequence (including the EEE copy down activity) can be considered as an implicit CCOB command
- Status and error flags are located in the FSTAT register

EEE sub-system

- User data is accessed via the EEE RAM partition (part or all of the buffer RAM)—writes to the EEE RAM can automatically generate an NVM record
- EEE data can be read immediately after it has been written.
- EEE records are stored in the EEE NVM partition (part or all of the D-flash)
- Status can be monitored via the ETAG count register, the MGBUSY flag in the FSTAT register, and by using the EEE Query CCOB command
- Error flags are located in the FSTAT and FERSTAT registers

P-flash and D-flash reads

- For reading / verifying code and constant data in the P-flash and D-flash
- ECC error flags are located in the FERSTAT register
- ECC error details are reported in the FECCR registers

1.5 The FTM Common Command Object (CCOB)

FTM operations are initiated by writing command information (typically including data and global address information) to a set of six banked word registers called the common command object (CCOB) and then clearing the CCIF flag in the FSTAT command register to launch the command (the CCIF flag is write 1 to clear).

The CCOB registers communicate command and parameter information to and from the memory controller.

When the CCOB command activity completes, the CCIF flag is set by the memory controller and the error and status flags in the FSTAT register will indicate if the FTM command activity access was successful.

There is no command queue as on the 0.25µ S12X) After a command is launched, the CCOB registers cannot be written to until the command has completed (i.e. the CCIF flag is set).

You must always follow the generic flash command write sequence as specified in the data book, both in application code and any development tools scripts for configuring or controlling the EEE. Most importantly, the CCIF flag must be checked to ensure that it is set before attempting to write the CCOB and launching any command. It is also highly recommended to evaluate the appropriate error flags following any command. There is a dedicated interrupt vector for the CCIF flag and the FTM command flow can be implemented as either a polled or an interrupt-driven flow.

There are five CCOB commands for controling EEE operation

- Full partition D-flash—Erases and configures the EEE record system (special mode only)
- Partition D-flash[1]—configures the EEE record system
- Enable EE emulation—turns programming of EEE NVM records on
- Disable EE emulation—turns programming of EEE NVM records off
- EEE Query[1]—reports EEE parameters and general status

CAUTION

Before launching any FTM commands involving program or erase operations, the FTM clock divider (FCLKDIV) must be configured or an access error will be generated (ACCERR bit set in the FSTAT register).

1.6 D-Flash

The D-flash is a separate flash block with error correction coding (ECC) intended for storing NVM data. On the S12XEP100 the D-flash is 16 K words.

The D-flash

- Is implemented as 16-bit data + 6-bit ECC syndrome
- Erases 128 words at a time
- Must be always be programmed as aligned words to allow the 6-bit syndrome to be generated
- Is aligned word oriented
- Can be programmed directly through the CCOB 1 to 4 words at a time
- Can be programmed implicitly by the EEE
- Can be flexibly partitioned for part use directly by the application and part use by the EEE

1.7 Buffer RAM

The buffer RAM can be flexibly partitioned as User RAM and EEE Ram. User RAM can be read and written at any time. EEE RAM can be read and written any time after the end of the FTM reset sequence. Part or all of the EEE RAM can be protected against inadvertent writes by configuring the EPROT register. The buffer RAM can be accessed on a byte-wide basis.

The EEE record system is based on aligned word data in the EEE RAM. Writing single byte data to the EEE RAM generates a record for the associated aligned word containing the byte. A multiple byte data entity crossing an aligned word boundary in the EEE RAM will be stored transparently across multiple EEE records. See Section 6.1, “Data Coherency Across RAM Word Boundaries,” for data coherency considerations.

1.The partition D-flash and EEE Query commands are not available on the xM22E mask sets.

To optimize the EEE write cycling, the S12XE implementation automatically filters out creating new NVM records for aligned word data if the data value written to the EEE RAM is the same as that already in the RAM. Writes of byte or misaligned word data will always create new records.

Following any reset, the data in the EEE RAM is initialized from the EEE NVM. The end of this copy-down process is indicated by CCIF being set by the memory controller. If the EEE RAM is accessed by the application during the copy down, the CPU will be stalled until the copy down is complete and the EEE RAM data is valid (see Section 3.1, “Reset Release”).

The EEE has an implicit data priority. Where there are multiple pending EEE data, data is programmed to the EEE NVM from the highest address in the EEE RAM first (see Section 2.2.2, “Data Writes”).

1.8 Configuring the EEE Size

Before the EEE can be used, its size must be configured. By default a new device is fully erased, resulting in all of the D-flash and buffer RAM configured as User D-flash and User RAM, with no EEE partition.

NOTE

The standard shipping condition for the D-flash and P-flash memory is erased with security disabled; however, it is recommended that each block or sector is erased before factory programming to ensure that the full data retention capability is achieved.

One of the two partition D-flash commands (see Section 5, “EEE CCOB Commands”) must be run on the memory controller once during the lifetime of an application to assign some of the above resources for use as EEE.

The D-flash is partitioned using a CCOB command taking two parameters:

ERPART = Number of 128 word regions of buffer RAM reserved for EEE

DFPART = Number of 128 word D-flash sectors reserved as User D-flash.

We use sectors and regions here because where flash sectors are physical entities defined by the sector erase boundaries, the RAM regions are only notional entities.

This is a logical way to define the EEE partition. It defines the EEE and User D-flash requirements necessary for the application while allocating all of the remaining D-flash for EEE to maximize the effective EEE write cycles. The copy-down time following reset is predominantly related the size of the EEE NVM partition and it may be advantageous for some applications, where maximum write cycling is not the key consideration but where the copy-down time is, to select a smaller EEE NVM partition to achieve a shorter copy-down time.

The configuration parameters are stored in the in a non-volatile information row in the D-flash and must be configured only once in the ECU production flow. In development they can be reconfigured as required. The rules for the minimum amount of D-flash required for EEE per region of the buffer RAM to be used as EEE are:

- (128–DFPART)/ERPART must be at least 8
- If ERPART ==1, then (128–DFPART) must be at least 12

Higher ratios will improve the effective write erase cycling. See Section 7, “Write Cycling.”

The partitioning commands format the EEE NVM ready for data record storage. Following partitioning, all EEE RAM locations read as erased (0xFFFF).

After partitioning, any D-flash sectors allocated for EEE (the EEE NVM) are automatically protected from being unintentionally modified by CCOB commands.

Any problems encountered during partitioning will be reported by the FSTAT error flags (see Section 8, “MCU System Considerations”). The EEE Query command can also be used at any time to read the partition configuration. This can be used as an additional verify following partitioning and also in the application flow to check to see that the EEE has been configured correctly to the application’s requirements.

Resets to the part must be avoided during partitioning and all efforts at the application level should be taken to avoid this, such as ensuring than an external watchdog timeout does not occur, etc.

In the case that partitioning activity is reset after it has started to program the EEE information, this condition will be detected as an EEE format fault by the FTM reset sequence (see Section 3, “Reset Activity”).

If a reset occurs during partitioning (either in normal or special mode) the EEE configuration can be recovered in special mode using the Full Partition D-Flash CCOB command or erased in special mode using the Erase All Blocks CCOB command.

2 Reading and Writing the EEE

2.1 The Basic Application API

Application data variables can be placed directly into the EEE RAM area.

The contents of the EEE RAM are backed up as aligned data word records in the EEE NVM. Programming of the EEE NVM data can be enabled and disabled using the Enable and Disable EEE CCOB commands (see, Section 5.3, “Enable EE Emulation,” and Section 5.4, “Disable EE Emulation”). The reset default is EEE NVM programming disabled.

Data coherency of variable elements that cross word boundaries in the EEE RAM must also be managed by the application with regards to any unplanned power down scenarios (see Section 6.1, “Data Coherency Across RAM Word Boundaries”).

2.1.1 Reading Data

From the application perspective, reading data from the EEE RAM is identical to reading from the main system RAM on the device (cycle timing may differ slightly).

2.1.2 Writing Data

Writing to the EEE RAM is identical to writing to the main system RAM (timing may differ slightly).

Writes to the EEE RAM (other than for aligned word data where the RAM contents are not changed) also cause a counter in the memory controller (visible in the ETAG register) to increment. The value of this counter indicates how many EEE NVM writes are currently pending at any time.

If more data is written to the same EEE RAM location before previous data has been programmed to the EEE NVM by the memory controller (because it was busy or the EEE is disabled), only the RAM contents will be updated; the increment of the counter will remain valid from first data write to the EEE RAM.

2.2 Memory Controller EEE Activity

2.2.1 Data Reads

Reading the EEE RAM data generates no EEE activity.

2.2.2 Data Writes

When the EEE is enabled (i.e. an Enable EEE command has been executed) the memory controller is triggered by a non-zero ETAG counter value. It automatically decrements the pending data count and generates a new EEE NVM record for the data at the highest addressed EEE RAM location containing pending data. During this activity there is no conflict with further EEE RAM writes by the CPU.

Pending data is programmed into the EEE NVM highest address in the EEE RAM first (i.e. records are created for data located closest to address 0x13_FFFF‘G first). Where there is data that should be preferentially written to the EEE NVM first, place it at a higher EEE RAM address. This prioritization only affects pending EEE data, any ongoing programming of lower addressed data must complete before following data written to a higher EEE RAM address will be programmed to the EEE NVM.

Memory controller CCOB commands, such as those for programming the User D-flash and for disabling programming EEE NVM, have priority over pending EEE data and can be safely interleaved with EEE activity.

CAUTION

While the EEE sub-system is accessing the D-flash, any reads of data in the User D-flash partition by the CPU are at risk of data corruption (see Section 6.5, “Considerations Accessing the User D-Flash”).

2.3 Monitoring EEE Activity

The ETAG register holds the count of pending EEE data (new data in the EEE RAM not yet read by the memory controller). Writes to the EEE RAM automatically increment ETAG so a non-zero count always shows there is EEE data pending. The ETAG count is decremented when the memory controller reads the data from the EEE RAM, not after a record for the data has been programmed to the EEE NVM, so an ETAG count of zero alone does not indicate that all new EEE data has been programmed to the EEE NVM.

The MGBUSY flag is set during any memory controller activity. Monitoring the ETAG for a zero count jointly with monitoring the MGBUSY flag has cleared indicates when all pending EEE data is programmed and all CCOB commands have completed.

Code example:

```
while((ETAG > 0) || (MGBUSY == 1)){  /* wait for EEE to */
}                                    /* to finish       */
```

These two elements, MGBUSY flag and ETAG count, are managed automatically by the memory controller and are independent with no specific timing dependency.

3 Reset Activity

3.1 Reset Release

Following the release of the system reset signal, the FTM reset sequence is activated as an integral part of any MCU reset. This consists of a core-hold phase (CPU held inactive) followed by a core-active phase (CPU executing instructions).

The FTM reset sequence can be treated as an implicit FTM command with the CCOB registers being locked and CCIF going high on completion. During this sequence

- The FTM is busy and memory controller commands cannot be executed
- The EEE RAM is not accessible

Each of the reset phases has EEE related activities. During the core hold phase

- the EEE protection (the EPROT register) is configured
- the ERPART/DFPART configuration is checked to determine if there is an EEE partition configured

The CPU core hold is released on completion. If an error is detected reading the EEE configuration, this will be flagged by the MGSTAT bits being set.

During the core active phase:

- The EEE pending count is cleared
- The EEE data is copied from the NVM EEE to the RAM EEE

On completion, the CCOB registers are unlocked, the CCIF flag in the FSTAT register, and any errors detected during the reset process are flagged by the MGSTAT bits (and possibly the ACCERR bit) being set.

CAUTION

If the CPU accesses any EEE RAM location before the CCIF flag is set, the CPU will be stalled until the end of the FTM reset sequence and the EEE RAM data is valid.

CAUTION

If an attempt is made to launch an FTM command before the end of this sequence, the command will be ignored.

3.1.1 Reset Timing

The reset timing is dominated by the EEE copy down and the size of the EEE NVM partition. For example, configuring all 128 D-flash sectors as EEE NVM and using a 4 MHz oscillator without the IPLL (2 MHz bus clock frequency) results in a core active phase on the order of 100 ms.

The reset timing is directly proportional to the bus clock frequency and using the IPLL can significantly reduce the core active phase timing. The IPLL module has a typical lock time of approximately 200 µs; configuring the IPLL immediately following reset (at the beginning of the C startup routine following the load of the stack pointer, for example) and selecting it as the bus clock as soon as possible can minimize the core active phase timing—the above example (with all of the D-flash configured as EEE NVM) can reduce to < 5 ms at 50 MHz bus clock.

3.2 Reset Assert

On the assertion of any reset, all memory controller activity is immediately halted. Any pending EEE RAM data will be lost. Any ongoing EEE NVM programming is also likely to be incomplete and the new EEE NVM record will be lost as well.

In this scenario, if pending data is not programmed successfully to the EEE NVM, the EEE RAM values following reset revert to the most recent previous values that were successfully programmed to the EEE NVM or, in the case that this was the first write to the EEE RAM location, to the erased value 0xFFFF.

Where a system reset or power down is generated intentionally by the application, in order not to lose the latest EEE data values, the EEE should be disabled or allowed to complete programming all pending data before the reset is asserted.

4 Record System Format

The EEE partition commands logically format the physical D-flash sectors allocated for EEE into a filing system managed as a circular buffer

4.1 Sector Headers

Each sector starts with a header field containing two words that define its status and erase count. The sector header format is designed to allow:

- All state transitions to be achieved using single programming operations
- Effective use of the D-flash ECC
- Detection of EEE reset condition
- Basic tracking of the sector erase count (up to 65534)

4.2 Data Records

EEE data is stored consecutively in the EEE NVM sectors. Each EEE NVM data record consists of two fields:

- A 16-bit address label field—identifying the EEE RAM word address associated with the data
- A 16-bit data field—the data associated with the EEE RAM location

Data records are created as required at the next available EEE NVM record location. There is no fixed positional relationship between any word location in the EEE RAM data and its associated valid data record in the EEE NVM partition.

4.3 EEE NVM Data Flow

The data record management flow is designed to allow:

- Record generation using single programming operations
- Effective use of the D-flash ECC
- Detection of the EEE state following reset

Each sector in the EEE NVM partition is in one of the following states:

- Containing data records
- Ready (erased)
- Flagged as invalid following a program or erase fault (and skipped by the EEE flow)

The following description considers two particular sectors containing data records

- the current sector where data records are being programmed, the Active sector
- the current sector containing the oldest data records, the Oldest_0 sector

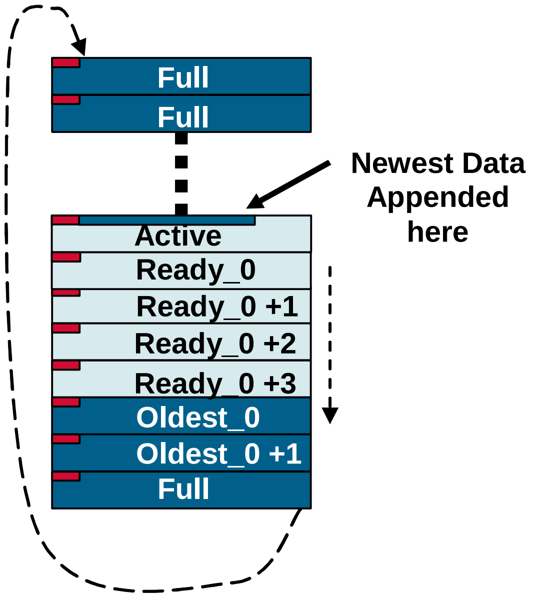

Figure 3. EEE Flash Sector Organization

New data written to the EEE RAM generates new EEE NVM records that are appended sequentially to the active sector.

A group of four Ready sectors are maintained in the erased state. These are used as a write buffer for an automatic, cyclical copy/erase process (data-clean up) that recovers unused space from old, out of date, data records as each sector is filled with new data records. Having four ready sectors allows for error conditions to be managed cleanly.

4.3.1 The Data Record Cycle

After the Active sector is full, then on the next record entry:

Ready_0 becomes the active sector

Ready_0+1 becomes the new Ready_0

Ready_0+2 becomes the new Ready_0+1

Ready_0+3 becomes the new Ready_0+2

A data clean-up cycle occurs to recover any valid data records from the Oldest_0 data sector to free it up to become the new Ready_0+3 sector.

The data clean-up cycle automatically creates new copies of valid records in Oldest_0 in the new Active sector. The now redundant Oldest_0 is erased and configured as the new Ready_0+3 (to maintain the four Ready sector reserve) and Oldest_0+1 becomes the new Oldest_0 sector. This is effectively a make before break flow with older valid data not erased until a newer copy is available.

In this fashion, the group of Ready sectors progresses through the EEE NVM as new data records are created. After the last sector in EEE NVM is reached, the process wraps around to the first sector in the partition, and so on.

The circular nature of this flow offers a number advantages including:

- Maximum flash usage
- All sectors get the same number of erase cycles
- No write erase overhead invalidating old data
- Robust recovery from inadvertent resets during EEE activity
- Where there is any dynamic data, all data records get refreshed over time, reducing the data retention requirement

In the case that there is a program or erase error that cannot be resolved, a sector is declared as invalid and is dropped from the EEE flow; subsequent loops around the EEE NVM skip this sector automatically. This is known as a dropped or dead sector.

4.4 Record System Status

Following a system reset, the FSTAT error flags can be checked to see if there were any EEE reset issues.

The EEE Query command can be executed at any time following the reset activity to evaluate the record system status.

Dropping a significant number of sectors from the EEE flow is considered an extreme condition associated with the D-flash being used well in excess of its normal operating life. If this limit is reached an error flag is generated, which indicates that the EEE is effectively worn out.

The number of dropped EEE NVM sectors that can be managed by the EEE is limited to a maximum of 24 (or the maximum available if (128–DFPART) is <= 28). For smaller EEE NVM configurations, the allowed number of dropped sectors is less.

For example, with ERPART = 3 and (128–DFPART) = 24, the maximum allowed dropped sector limit is 13 before the EEE will generate a format error.

A more controlled approach to managing the status of the EEE is recommended using the EEPROM emulation query command to monitor the number of dropped sectors and flagging a system error if this exceeds 25% of the total sectors allocated for EEE.

5 EEE CCOB Commands

There are five EEE related CCOB commands:

- Full partition D-flash—Erases and formats the EEE record system (special mode only)
- Partition D-flash[1]—Formats the EEE record system (execute once in normal mode)
- Enable EE emulation—Turns on programming of NVM records
- Disable EE emulation—Turns off programming of NVM records
- EEE Query[1]—Reports EEE parameters / status

The partition D-flash commands configure the resources allocated to the EEE and low-level format the EEE (to a default erased value of 0xFFFF). Partitioning stores the EEE configuration parameters and initializes the header field in each EEE NVM sector.

There are two versions of the partition command, one that can repartition the EEE and is intended for use during production programming and for during development and another that can only partition an already erased device and is intended to support initial configuration in normal mode by a boot loader program or by an application downloaded via a boot loader. The partition process must be run once in the lifetime of an application to partition the EEE / D-flash.

NOTE

There is no support in normal mode for repartitioning of the EEE.

If an application does not require EEE, the device should be pro-actively partitioned for zero EEE (ERPART = 0 and DFPART = 128) to avoid it being accidentally partitioned incorrectly at some point in the application.

1.The partition D-flash and EEE Query commands are not available on the xM22E mask sets.

5.1 Full Partition D-Flash

This is not an application level command. It is only available in special modes and should be run during initial device programming.

It erases the full D-flash and initializes the EEE resources.

For production devices it should be run one time only to define the size of the device EEE specific to the application.

During development it can be run multiple times.

CAUTION

This command erases the D-flash so it is important that full partition of the D-flash be executed before any static application data is programmed directly into the D-flash using the Program D-flash command.

5.2 Partition D-Flash

This is an application-level command that can execute in normal modes. It is intended to allow serial bootloaders and applications to configure the EEE once.

It initializes the EEE resources.

It should be run one time only to configure the EEE.

It will only execute successfully in the case that the D-flash partition information is in the default factory shipped state or the device has been mass erased using the FTM Erase All Blocks command (available only in special modes)—i.e. if the EEE is not already partitioned. If the D-flash block is not fully erased it will generate an ACCERR flag.

CAUTION

This command requires that the D-flash be fully erased so it is important that partition of the D-flash be done before any constant application data is programmed directly into the D-flash using the Program D-flash command.

NOTE

The standard shipping condition for the D-flash and P-flash memory is erased with security disabled; however, it is recommended that each block or sector is erased before factory programming to ensure that the full data retention capability is achieved.

5.3 Enable EE Emulation

Following the memory controller reset process or the Disable EE emulation command, the EEE is considered to be in an idle state. Writing to the EEE RAM will increment the pending data count but the memory controller will not program records to the EEE NVM.

After the command has completed, the EEE is considered to be in an active state generating EEE NVM records for new EEE RAM data.

The enable command can be run at any time after the FTM reset activity has completed. It:

- Tidies up any previous EEE NVM activity aborted by a reset
- Allows the memory controller to respond to pending EEE data and program EEE records to the EEE NVM.

There is no specific flag to indicate that the EEE is in the enabled or disabled condition. There is no issue executing the Enable EEE command where it is already enabled.

5.4 Disable EE Emulation

The Disable EE command ensures that the EEE is in an idle state. This is the default state of the EEE after a reset.

After the command has completed, the EEE is considered to be in a paused state. Pending EEE RAM data will not generate EEE NVM records. Executing the Enable EEE command again following the Disable EEE command causes EEE NVM records to be programmed for pending EEE RAM data.

The disable command can be run at any time after the FTM reset activity has completed. It prevents pending EEE RAM data generating EEE NVM records.

There is no specific flag to indicate that the EEE is in the enabled or disabled condition. There is no issue executing the Disable EEE command if it is already disabled.

5.5 EEE Query

The EEE Query command reports a number of EEE parameters to the application.

These parameters indicate the configuration of the EEE resources (i.e. the partition information) and the general status of the EEE record system.

The Query EEE command can be run at any time after the FTM reset activity has completed. It would be logical to run it following any reset and in the case of any EEE error being detected.

It returns EEE status information as detailed in the following five sub-sections.

5.5.1 DFPART Configuration

Indicates the number of D-flash sectors allocated for User D-flash.

This can be checked against that expected by the application. This was primarily intended as a check that the EEE was partitioned as intended and human error had not resulted in a wrong configuration. Returns 0xFF if the D-flash is not partitioned or is partitioned for zero EEE.

5.5.2 ERPART Configuration

Indicates the number of 128 word buffer RAM regions allocated for EEE RAM.

This can be checked against that expected by the application. This was primarily intended as a check that the EEE was partitioned as intended and human error had not resulted in a wrong configuration. Returns 0xFF if the D-flash is not partitioned or is partitioned for zero EEE.

5.5.3 ECOUNT Value (of the Current Ready_0 Sector)

Reports the number of times that the next available Ready sector (see Ready_0 sector, Section 4.3, “EEE NVM Data Flow”) has been erased.

The erase count returned by EEE Query typically reflects the wear level of the D-flash sectors in the EEE NVM. As the ready sectors move through the EEE NVM (as new data is written to the EEE) the erase count returned by different queries will typically be that of different sectors. Under ideal conditions the erase count for all sectors will be similar, incrementing once for each loop through the EEE NVM partition. The ECOUNT value for each sector does not roll over, it will stop incrementing when it reaches 65534 (0xFFFE)—this is in excess of the specified 50K erase limit of the D-flash.

However, in the case that a reset occurs during the updating of the sector in a data clean-up cycle, the erase count for that sector can be lost and is reset to zero following the next EEE enable command. It then proceeds to increment from zero again with each loop through the EEE NVM partition. This is of no consequence to the EEE process because the specific erase count value is of no importance to the EEE flow.

In this case, the value returned by EEE Query for any sectors with counts corrupted by reset will no longer reflect the actual erase level of the EEE NVM. Unless it is certain that resets will not occur during EEE activity, the erase count returned by Query EEE should not be relied upon by an application without some further checking of the value by the application, such as saving a copy of the returned count after each reset and validating the value reported by Query EEE against the copy to ensure that the sector count has not reset, and/or running query EEE periodically and keeping track of the maximum ECOUNT value across multiple sectors.

5.5.4 Current Dropped (or Dead) Sector Count

Reports the number of EEE dropped sectors, sectors where a program or erase operation have recorded a fault and have been dropped from EEE use.

Flagging a system error is recommended if this number exceeds 25% of the total sectors allocated for EEE NVM, i.e. if greater than 0.25 * (128-DPFART). See Section 4.4, “Record System Status,” for further detail relating to the maximum number of dropped sectors.

5.5.5 The Current Ready (or Erased) Sector Count

Reports the number of EEE NVM Ready sectors, sectors available for programming data to during a copy/erase cycle. In most conditions this should be three or four. It should always be at least one.

6 Data Considerations

6.1 Data Coherency Across RAM Word Boundaries

As the EEE RAM is stored to the EEE NVM on a word basis, data coherency needs to be accounted for. Any data element in the EEE RAM that crosses a word boundary is stored across multiple records and has the potential to become corrupted if the device is reset during its update.

For example, a 4 byte long variable on an odd address boundary is stored across three independent EEE NVM records. If the first of the data records is updated and a reset occurs while programming the second, then the recovered value of the long variable will be corrupted. This is no different to current EEPROM implementations and the validity of such data must be managed at the application level, with multiple data copies or NVM flags indicating when such data is being updated, for example.

Single byte data elements and aligned word data elements do not have this constraint as the data coherency for these is taken care of by the EEE design.

CAUTION

Complex high-level data structures that contain single byte and word size data elements only should be arranged and placed so that any word elements fall on even address boundaries.

6.2 Using the EEE for Constant Data

The nature of the data clean-up mechanism means that records for constant data are periodically copied as dynamic data is refreshed.

When a new record needs to be programmed and all of the records in the oldest full sector are active records (i.e. there are no newer records for this data because it is constant data) the whole sector is copied without freeing up any space.

In this way, a large amount of constant data can cause several sectors to be copied and erased consecutively in the data clean-up cycle, causing a relatively long clean-up cycle.

This is primarily an issue when it is important that any pending data be programmed on power down. Because of the repeated copy of constant data, it also results in a write cycle penalty and can impact the achievable cycle limit for the application.

This does not impact the application data access but can impact the worse case time required to for all pending data to be programmed to the EEE NVM before powering down.

In this case, you should store constant data directly into the P-flash or User D-flash in this case as opposed to the EEE.

6.3 Using the D-Flash for Urgent Write Data

Another consideration of the data clean-up behavior is the latency of the EEE for programming urgent data to the EEE NVM. As previously described, data clean-up (copy/erase) cycles are performed when necessary as new EEE NVM data records are programmed and, especially in the presence of constant data, more than one back-to-back cycle may be required. This can result in a long effective program time for an EEE NVM record.

While this activity is ongoing, other new EEE RAM data will not be immediately programmed to the EEE NVM. However, memory controller commands can be issued and the EEE can be paused between consecutive data clean-up cycles by executing the Disable EE emulation command. Urgent data can then be programmed to the P-flash or User D-flash.

Program P-flash or D-flash commands can be launched without disabling the EEE, although this is not as effective for more than one program command as the EEE continues pending activities between CCOB commands and so will introduce delays between consecutive direct flash program commands while it programs EEE records.

Where there is data that must be urgently programmed to the NVM under certain conditions, it is recommended to store this directly into the P-flash or User D-flash as opposed to the EEE.

6.4 Avoiding Constant Data Blocks

The data clean-up cycle copies any records for valid data from the current oldest full sector to the first available ready sector. All constant data records are copied and therefore remain in the same relative sector. By spreading out constant data across the record system, it can minimize the likelihood of multiple sectors requiring sequential data clean-up cycles.

To do this, we need to ensure that this data is not programmed to the EEE NVM consecutively. Typically, following partitioning of the EEE or during production programming, initial values for all EEE data will be programmed in the EEE RAM. The order in which EEE NVM records for this initial data set is written to the EEE NVM can be controlled by the order in which they are written to the EEE RAM.

The simplest option is to interleave the constant data records with the dynamic data records proportionally. There are 63 data records per EEE NVM sector. When initializing the EEE data to the EEE RAM it should be written in a sequence of:

1. Write (% constant data * 63) constant data words
2. Write (% dynamic data * 63) dynamic data words
3. Wait for the EEE to complete programming (following each 63 word write sequence)
4. Repeat from step 1 to initialize all EEE RAM data locations

Step 3 is important because writing the data to the EEE RAM continuously in a loop results in the records being stored along the lines of the EEE address priority and not in the sequence they were written to the EEE RAM.

An example for a data set where 20% of the data is constant and 80% is dynamic is

1. Write 13 constant data words
2. Write 50 dynamic data words
3. Wait for the EEE to complete programming
4. Loop to step 1 until EEE data fully initialized

More complex flows are possible that would distribute constant data over all available EEE NVM sectors at the expense of initialization time.

6.5 Considerations Accessing the User D-Flash

Reads of the D-flash user partition while the EEE is active can be disturbed by EEE activity accessing the EEE NVM.

To avoid such collisions, pause the EEE with the Disable EE command before reading the D-flash array, or alternatively, check that the ETAG value is zero and the MGBUSY flag is low (indicating that the memory controller is idle)— to confirm that there is no pending EEE NVM activity.

Programming of the D-flash user partition while the EEE is active is managed cleanly by the FTM prioritization.

Consider copying any constant User D-flash data to RAM before enabling the EEE.

7 Write Cycling

Traditionally, EEPROM is rated for the number of write / erase cycles because for each write an erase cycle is required. Because of the cyclical nature of the EEE, it is more reasonable to consider the total available number of writes to the EEE RAM, while remaining within the 50,000 erase cycles available per EEE NVM D-flash sector.

A write cycle of the EEE can be defined as any write to an EEE RAM location that generates a new EEE NVM data record.

Misaligned word writes to the EEE RAM generate two data records that must be considered as two write cycles. This is another reason to avoid misaligned word data.

Wear leveling is inherent to the EEE implementation.

At it’s simplest, an EEE NVM to EEE RAM partition ratio of 8:1 results in approximately 4 NVM data records for every EEE RAM word location.

In typical use, the actual EEE write cycle capability will not be a simple function of the EEE NVM to EEE RAM partition ratio, but is more closely related to the number of EEE RAM data words used versus the size of EEE NVM partition. For instance, having only a single word of EEE data and using 16K words of EEE NVM results in an effective EEE NVM to EEE RAM ratio of 16384:1.

However, the EEE data size to EEE NVM ratio does not represent the full picture either. The relative rate that various data in the EEE is updated is also part of the write cycle equation. Data that is updated often generates more new EEE NVM records than data that is not written so often.

Also, due to the cyclical nature of the record system, new copies of EEE NVM records for constant EEE data are also created periodically by the data-clean up process, representing an additional write/erase overhead.

Due to these many interrelated factors, calculating the write loading for any given data requirement is beyond the scope of this document. However, it can be evaluated using the S12XE_EEE_Calculator utility that can be downloaded from the S12XE design tools page at www.freescale.com.

The current MC9S12XEP100 specification for cycling only one aligned word in the EEE RAM using all 16K words of the D-flash (EEE NVM to EEE RAM = 16384), guarantees a minimum of 325 million write cycles with a typical of 3200 million write cycles across the automotive temperature range.

8 MCU System Considerations

8.1 Status Flags

8.1.1 Regarding FTM Fault Coverage

The sophisticated fault detection on the FTM and EEE subsystem is intended to ensure an application can be notified of a failure event in any case that there is a fault. It is not because a significant level of faults are expected from the S12XE flash technology.

At the time of publication, all of the data for this flash strongly supports it being highly reliable with very low fault levels.

8.1.2 EEE Flag Summary

This section provides a summary of the EEE specific error reporting. If using the EEE, it is recommended to enable the flash error interrupt to service the error flags as a number of them indicate critical EEE faults if they occur.

8.1.2.1 Critical EEE Fault Flags

The following flags indicate serious EEE faults intended to flag that a major problem exists. On any of these faults the EEE should be considered unavailable.

The access error (ACCERR) bit will be set by:

- The Enable EEE and Disable EEE commands if there is no EEE partition defined
- Either of the partition D-flash commands if an illegal partition ratio is requested
- The Enable EEE command if there is no EEE partition defined
- Errors encountered while initializing the EEE RAM during the FTM reset sequence

The MGSTAT0 and MGSTAT1 bits will set following reset for:

- The Partition D-flash command if an EEE partition is already defined
- Unrecoverable fault reading EEE partition information —in this case, the EEE feature is disabled with ERPART => 0x0000 and DFPART => 0xFFFF
- Double fault detected reading the EEE protection byte in the flash configuration field—in this case, the EPROT register is loaded with 0x7F = EEE RAM fully protected

On detecting the MGSTAT bits set following reset, consider generating a system reset with the COP or an illegal address access to cause the reset sequence to be repeated.

The following two flags report critical operating conditions and should be treated as a limp home condition for the EEE subsystem. The EEE RAM data can continue to be read and written. Programming of EEE records to the EEE NVM will be blocked while these flags are set.

The EEE Error Interrupt 0 flag (ERSVIF0) is set if

- The memory controller is unable to re-format an EEE NVM sector
- There are no ready sectors available
- The maximum number of data sectors having a program or erase fault has been reached (see Section 4.4, “Record System Status”

The EEE Error Interrupt 1 flag (ERSVIF1) is set if

- The memory controller is unable to change the state of an EEE NVM sector

8.1.2.2 EEE Warning Flags

The following flags typically report conditions that result in dropping a sector from the EEE flow. The EEE can continue to function until the dropped sector limit is reached or until there are no ready sectors available.

The EEE Erase Error Interrupt flag (ERSERIF) is set if

- EEE erase operation fails to verify. The sector will be dropped from the EEE use.

The EEE Erase Error Interrupt flag (PGMERIF) will be set if.

- EEE program operation fails to verify. The sector will be dropped from the EEE use.

On the occurrence of either of these flags, the query EEE command can determine the current number ready and dropped sectors and flag a system error (see Section 5.5, “EEE Query”).

8.1.2.3 Protection Violation

This flag relates to the EEE as system level error flag, as opposed to an EEE subsystem flag.

The EEE Protection Violation flag (EPVIOLIF) will be set if:

- There is an attempt to write an area of the EEE RAM protected by the EPROT setting. The write will be blocked and EEE activity will be unaffected.

The EPROT register defines which EEE RAM areas are protected against writes, it does not protect access to the D-flash. Access to the EEE NVM region of the D-flash is automatically protected.

8.2 EEE and D-Flash ECC

Single bit faults are automatically corrected by the ECC and the corrected data will be read. Single bit faults are not reported by the EEE.

Double bit fault errors on data records are managed by the EEE. They are not flagged directly as ECC read errors.

A double fault can occur in the case of a reset during the programming of any part of a data record. A record contains two 16-bit data fields and two ECC syndromes and a reset can occur during the programming of any of these. The EEE reset recovery flow treats records with ECC faults as records corrupted by a reset and manages them cleanly.

Where programming of a record is interrupted by a reset, this can be considered the same as a pending EEE RAM write that was not programmed to the EEE NVM before the reset occurred (unlike traditional EEPROM a reset while programming data is most likely to cause data corruption or marginal programming of the data).

On detecting a double bit fault, the copy-down process considers the record as damaged and ignores it. The data or address information in a corrupted data record cannot be trusted so there is no way to identify which EEE RAM word location the damaged record related to or what the valid data value might have been.

The D-flash programming operation includes a margin verify and existing reliability data for the D-flash is excellent. As the EEE NVM data is periodically modified, the EEE scheme refreshes all data records over time (by the data-clean up mechanism). The D-flash data retention spec must still be observed, especially where there is a possibility of long periods of EEE inactivity or very low data cycling rates.

8.3 Low-Power Modes

Entry into wait mode or background debug mode has no effect on the FTM.

If the MCU executes a stop instruction:

- Any ongoing CCOB command will be completed before stop mode is entered.
- With EEE enabled, any pending EEE data will also be programmed to the EEE NVM before the memory controller halts activity and the system clocks are stopped.

8.4 Clock Considerations

The FTM state machine is clocked from the system bus clock while low-level NVM timing (for program and erase pulses) is derived from the external oscillator.

FTM timing can be minimized by using a faster bus clock during all NVM operations including all EEE activity and the EEE copy down following reset.

As the IPLL on the S12XE does not require external components, it can be enabled on any hardware target. The low-level (flash hardware) programming and erase timings are not dependent on the bus frequency; therefore, loss of IPLL lock does not affect the FTM program or erase functionality, although it will extend the command operational timing. The time required for the IPLL to achieve lock is always quicker than the EEE data copy-down process and the copy-down time may be reduced significantly by enabling and selecting the IPLL for the bus clock in the start up code, immediately following reset and the initialization of the stack pointer (see Section 3.1.1, “Reset Timing”).

CAUTION

The oscillator clock is required for FTM timing; therefore, any FTM operations requiring program or erase are not allowed in fast-wake or self-clock modes.

Ensure that the EEE remains disabled when using fast-wake mode.

Disable the EEE at the beginning of the self-clock mode interrupt routine.

8.5 Debugging Considerations

In general, working with data in the EEE RAM should be transparent to the debugger (after the copy-down activity is completed). Reading and writing the EEE RAM is the same for the system RAM and a debugger can treat it as such without any problems.

Reads of the EEE RAM via BDM during the EEE copy-down process may not read valid data until after the EEE copy down is completed and the CCIF flag is set. Writes of the EEE RAM via the BDM during the EEE copy-down process should be avoided. This is an unlikely scenario because although debuggers may need to update a display window immediately following reset, they are unlikely to write data to the EEE RAM.

When debugging any EEE NMV management activity, such as checking the ETAG value and MGBUSY flag, having an awareness of the following aspects of the EEE can help avoid confusion.

The memory controller operates independently of the CPU and the XGATE and does not have a freeze mode because if the memory controller clocks are halted with the high voltage enabled on the flash array, it could overstress or damage it. This has always been true of any HC12/S12/S12X NVM state machines including EEPROM programming. This can be observed by writing to the EEE RAM via the BDM with the CPU halted and the EEE enabled. The EEE NVM record is programmed and the new data is retained through a power down.

One side effect of this is that for any activity where the CPU halts during development (on a breakpoint or single step) where it would not halt during the real application flow, then any pending memory controller activity continues independently of the CPU until it is completed. The state of the system following the breakpoint, which can take some time for the debugger read, can be different from the real time state in the application. For example, if the application has a loop testing ((ETAG > 0) || (MGBUSY == 1)) and you single step it, it passes this test on the first loop, whereas running the code without the step results in multiple executions of the loop. This is also true of the S12 EEPROM if you have a loop waiting for CCIF to be reset by the state machine or for an external EEPROM.

Another consideration is the reset activity of the FTM. Resetting the CPU into special single chip mode (i.e. by the BDM cable) causes the CPU to enter BDM active mode awaiting BDM commands. The FTM reset sequence can take some time following the reset to complete the EEE copy down. Development tool activities or scripts that execute reset and immediately try to perform FTM commands or access the EEE RAM must verify the CCIF flag is set before these activities are performed. In the first case the command will not be executed. In the second, the data read from the EEE RAM cannot be trusted until the CCIF flag is set. It is recommended that the error flags in the FSTAT register also be evaluated following any reset.

9 Some Comments on EEE Timing

In normal operation, after the EEE reset process has completed, the access timing to EEE data is that of reading and writing to the EEE RAM. The reset timing is described in Section 3.1.1, "Reset Timing."

At a high level, the background timing for EEE record programming is unpredictable and data dependent due to the copy/erase mechanism. The status of pending data being stored to the NVM can only be judged by the state of the ETAG count and the MGBUSY flag. See Section 2.3, "Monitoring EEE Activity."

At publication time, timing equations for each of the EEE commands and activities were being added to the S12XE data sheet. EEE timing is subject to both NVM write/erase and data processing activities. The first of these is fixed by the physics of the flash array and derived from the fixed oscillator clock frequency. The second is proportional to the bus frequency, so it will always be advantageous to use as fast a bus clock as possible when using the EEE.

At faster bus clocks, the average time for programming a data word record to the EEE NVM is significantly quicker than updating a word in the 0.25µ S12X EEPROM (at 50 MHz fbus, the average write of an NVM record is <1 ms). The FTM operations for programming any individual data record however will be faster or slower than this due to the periodic nature of the data clean-up process (where the copy and erase overhead occurs). This is especially true where there are full sectors containing constant data (this is primarily of concern where EEE data might need to be written before the power supply power is disabled). Where there are few or no data sectors containing constant data, at faster bus speeds updating 2K words in the EEE is generally going to be faster than updating 2K words in a traditional EEPROM.

As described in Section 6.2, "Using the EEE for Constant Data," an EEE NVM sector full of constant data will require an additional copy/erase cycle when reached in the round-robin loop. At 50 MHz the copy/erase cycle for each sector full of constant data record can take approximately 38 ms. Consecutive sectors full of constant data records extend this activity proportionally. Pending CCOB commands will be executed between consecutive copy/erase cycles. This defines the latency of being able to do anything else with the FTM while the EEE is enabled and updating data.

The EEE NVM erase operation is margin verified and may vary from ~5 ms to ~20 ms depending on the number of times the sector has previously been erased. This means that the EEE data clean-up cycle timing can be significantly faster on a new device and extend as the application ages and the EEE NVM erase count increases.

10 Glossary

ACCERR—memory controller Access Error flag.

Aligned word—16-bit element located at an even memory address.

API—Application programming interface.

BDM—Background debug module.

Buffer RAM—Buffer RAM allocated to support EEE operation.

Byte data—8-bit data element.

CCIF—Memory Controller Command Complete Interrupt flag.

CCOB—Memory controller common command object. Common registers used when executing FTM commands.

Constant EEE data—EEE data that is never or rarely updated by an application.

Core active phase—The part of the reset sequence where the CPU core is running.

Core hold phase—The part of the reset sequence where the CPU core is held inactive.

CPU—Central processing unit.

Data clean-up cycle—EEE D-flash sector copy/erase operation.

D-flash—Flash memory optimized for NVM data storage.

DFPART—Number of D-flash sectors allocated for direct access by the application.

ECC—Error correction coding.

ECOUNT—EEE D-flash sector erase count.

ECU—Electronics control unit.

EEE—Emulated EEPROM sub-system.

EEE RAM—Buffer RAM allocated to EEE.

EEE NVM—D-flash allocated to EEE.

EEE Partition—EEE resource configuration.

ERPART—Number of buffer 256 Byte RAM regions allocated as EEE RAM.

EEPROM—Electrically erasable programmable read-only memory.

EPROT—Memory controller EEE Protection register.

ETAG—Memory controller register containing pending EEE data counter.

FTM—S12XE NVM and memory controller module.

FTR—EEE configuration flash to RAM ratio.

FERSTAT—Memory controller Flash Error Status register.

FSTAT—Memory controller Flash Status register.

IP—Intellectual property.

IPLL—Phase-locked loop with internal filter.

MGBUSY—Memory controller Busy flag.

Misaligned Word—16-bit element located at an odd memory address.

NVM—Non-volatile memory.

NVRAM—Non-volatile RAM.

RAM—Random-access memory.

System RAM—Main RAM used for CPU and XGATE data.

User D-flash—D-flash sectors available directly to the application.

User RAM—Buffer RAM not allocated to EEE.

Wear leveling—Technique for maximizing the write/erase capability of the EEE NVM.

Word Data—Data accessed as a 16-bit element.

‘G—S12X global address notation.

0x—Hexadecimal notation.

|| — Logical OR notation.

> —‘Is greater than’ notation.

< —‘Is less than’ notation.

== — Logical equate notation.

Document Number: AN3490
Rev. 0
09/2007