EETS4K
Block User Guide
V02.06

Original Release Date: 22 FEB 2001
Revised: 23 JAN 2003

Motorola, Inc

Revision History

Version Number	Revision Date	Effective Date	Author	Description of Changes
V01.00	22FEB01	15NOV00		Initial Version.
V02.00	22MAY01	27MAR01		Do not set PVIOL for erase verify command if address written to is in a protected area. Allow data writes of $00 and $40 to ECLKDIV.
V02.01	25MAY01	25MAY01		Make formats SRS V2 compliant. Reorder and restructure document. Add overview block diagram.
V02.02	19JUL01	10JUL01		Add document names. Hide names and variable definitions.
V02.03	29OCT01	29OCT01		Minor cleanup.
V02.04	11MAR02	11MAR02		Modify document number. Fix cross references.
V02.05	09JUL02	09JUL02		Modify document number. Rearrange Section **3.2 Module Memory Map**.
V02.06	23JAN03	23JAN03		Add description of EADDR and EDATA registers.

Table of Contents

Section 5 Resets

Section 6 Interrupts

List of Figures

List of Tables

Section 1 Introduction

1.1 Overview

This document describes the EETS4K module which is a 4K byte EEPROM (Non-Volatile) memory. The EETS4K block uses a small sector Flash memory to emulate EEPROM functionality. It is an array of electrically erasable and programmable, non-volatile memory. The EEPROM memory is organized as 2048 rows of 2 bytes (1 word). The EEPROM memory's erase sector size is 2 rows or 2 words (4 bytes).

The EEPROM memory may be read as either bytes, aligned words or misaligned words. Read access time is one bus cycle for byte and aligned word, and two bus cycles for misaligned words.

Program and erase functions are controlled by a command driven interface. Both sector erase and mass erase of the entire EEPROM memory are supported. An erased bit reads '1' and a programmed bit reads '0'. The high voltage required to program and erase is generated internally by on-chip charge pumps.

It is not possible to read from the EEPROM memory while it is being erased or programmed.

The EEPROM memory is ideal for data storage for single-supply applications allowing for field reprogramming without requiring external programming voltage sources.

WARNING

A word must be erased before being programmed. Cumulative programming of bits within a word is not allowed.

1.1.1 Glossary

Command Sequence

A three-step MCU instruction sequence to program, erase or erase-verify the EEPROM.

1.2 Features

- 4K bytes of EEPROM memory.
- Minimum erase sector of 4 bytes.
- Automated program and erase algorithms.
- Interrupts on EEPROM command completion and command buffer empty.
- Fast sector erase and word program operation.
- 2-stage command pipeline.
- Flexible protection scheme for protection against accidental program or erase.
- Single power supply program and erase.

1.3 Modes of Operation

- Program and erase operation (please refer to **4.1** for details).

1.4 Block Diagram

Figure 1-1 shows a block diagram of the EETS4K module.

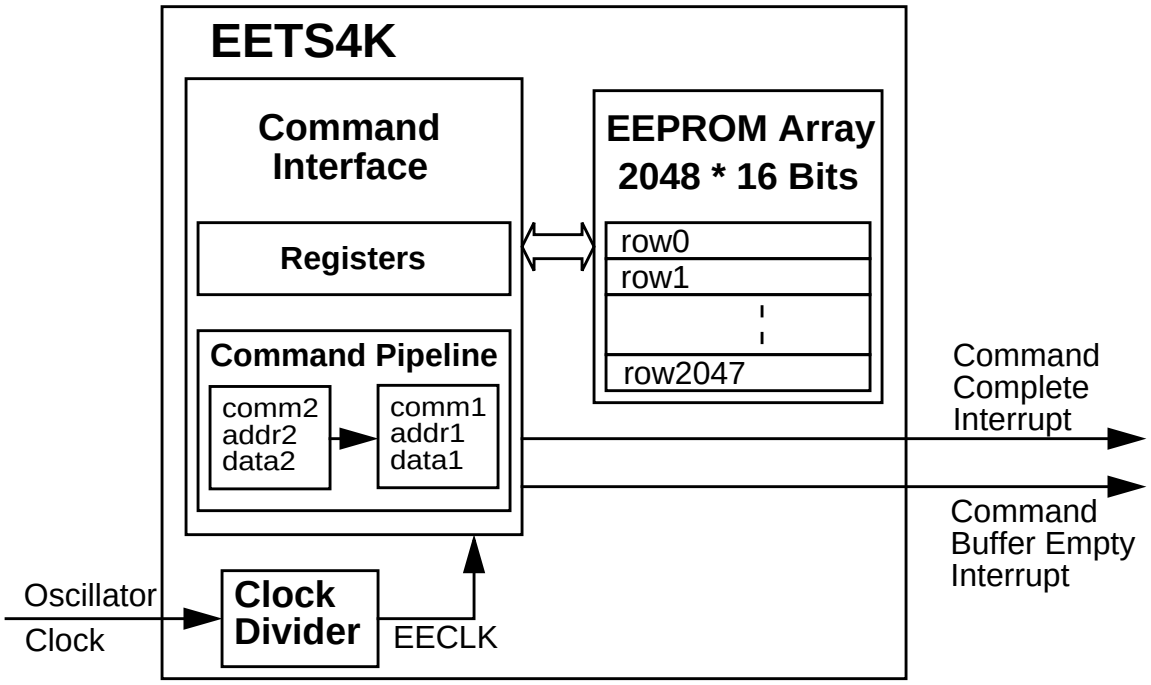

Figure 1-1 Module Block Diagram

Section 2 External Signal Description

2.1 Overview

The EETS4K module contains no signals that connect off chip.

Section 3 Memory Map and Registers

3.1 Overview

This section describes the EETS4K memory map and registers.

3.2 Module Memory Map

Figure 3-1 shows the EETS4K memory map. Location of the EEPROM array in the MCU memory map is defined in the specific MCU Device User Guide and is reflected in the INITEE register contents defined in the HCS12 Core User Guide. Shown within the EEPROM array are a protection/reserved field and user-defined EEPROM protected sectors. The 16 byte protection/reserved field is located in the EEPROM array from address $_FF0 to $_FFF. A description of this protection/reserved field is given in **Table 3-1**.

Table 3-1 EEPROM Protection/Reserved Field

Array Address	Size (bytes)	Description
$_FF0 - $FFC	13	Reserved
$_FFD	1	EEPROM Protection byte
$_FFE - $_FFF	2	Reserved

The EEPROM module has hardware interlocks which protect data from accidental corruption. A protected sector is located at the higher address end of the EEPROM array, just below address $_FFF. The protected sector in the EEPROM array can be sized from 64 bytes to 512 bytes. In addition, the EPOPEN bit in the EPROT register (see section **3.3.5**) can globally protect the entire EEPROM array.

NOTE

Chip security is defined at the MCU level.

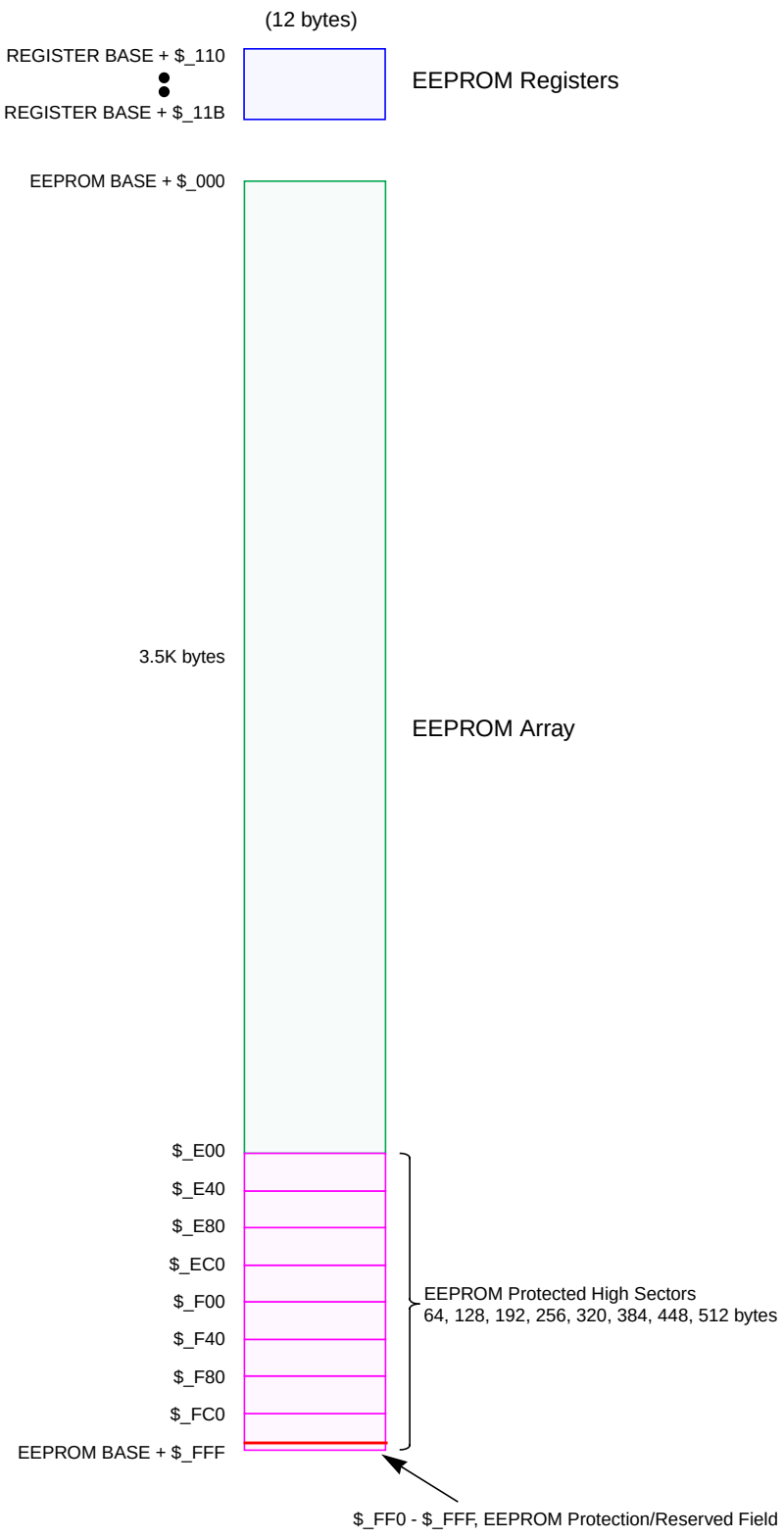

Figure 3-1 EEPROM Memory Map

The EEPROM module also contains a set of 12 control and status registers located in address space BASE + $110 to BASE + $11B.

Table 3-2 gives an overview on all EETS4K registers.

Table 3-2 EEPROM Register Memory Map

Address Offset	Use	Access
$_00	EEPROM Clock Divider Register (ECLKDIV)	R/W
$_01	RESERVED1[1]	R
$_02	RESERVED2[1]	R
$_03	EEPROM Configuration Register (ECNFG)	R/W
$_04	EEPROM Protection Register (EPROT)	R/W
$_05	EEPROM Status Register (ESTAT)	R/W
$_06	EEPROM Command Register (ECMD)	R/W
$_07	RESERVED3[1]	R
$_08	EEPROM High Address Register (EADDRHI)	R/W
$_09	EEPROM Low Address Register (EADDRLO)	R/W
$_0A	EEPROM High Data Register (EDATAHI)	R/W
$_0B	EEPROM Low Data Register (EDATALO)	R/W

NOTES:
1. Intended for factory test purposes only.

NOTE: *Register Address = Register Base Address + $110 + Address Offset, where the Register Base Address is defined by the HCS12 Core INITRG register and the Address Offset is defined by the EEPROM module.*

3.3 Register Descriptions

3.3.1 ECLKDIV — EEPROM Clock Divider Register

The ECLKDIV register is used to control timed events in program and erase algorithms.

Register address BASE + $110

	7	6	5	4	3	2	1	0
R	EDIVLD	PRDIV8	EDIV5	EDIV4	EDIV3	EDIV2	EDIV1	EDIV0
W								
RESET:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 3-2 EEPROM Clock Divider Register (ECLKDIV)

All bits in the ECLKDIV register are readable, bits 6-0 are write once and bit 7 is not writable.

EDIVLD — Clock Divider Loaded.

1 = Register has been written to since the last reset.

0 = Register has not been written.

PRDIV8 — Enable Prescaler by 8.

1 = Enables a prescaler by 8, to divide the EEPROM module input oscillator clock before feeding into the CLKDIV divider.

0 = The input oscillator clock is directly fed into the ECLKDIV divider.

EDIV[5:0] — Clock Divider Bits.

The combination of PRDIV8 and EDIV[5:0] effectively divides the EEPROM module input oscillator clock down to a frequency of 150kHz - 200kHz. The maximum divide ratio is 512. Please refer to section **4.1.1** for more information.

3.3.2 RESERVED1

This register is reserved for factory testing and is not accessible to the user.

Register address BASE + $111

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 3-3 RESERVED1

All bits read zero and are not writable.

3.3.3 RESERVED2

This register is reserved for factory testing and is not accessible to the user.

Register address BASE + $112

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 3-4 RESERVED2

All bits read zero and are not writable.

3.3.4 ECNFG — EEPROM Configuration Register

The ECNFG register enables the EEPROM interrupts.

Register address BASE + $113

	7	6	5	4	3	2	1	0
R	CBEIE	CCIE	0	0	0	0	0	0
W								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 3-5 EEPROM Configuration Register (ECNFG)

CBEIE and CCIE are readable and writable. Bits 5-0 read zero and are not writable.

CBEIE — Command Buffer Empty Interrupt Enable.

The CBEIE bit enables the interrupts in case of an empty command buffer in the EEPROM.
1 = An interrupt will be requested whenever the CBEIF flag, **Figure 3-7**, is set.
0 = Command Buffer Empty interrupts disabled.

CCIE — Command Complete Interrupt Enable.

The CCIE bit enables the interrupts in case of all commands being completed in the EEPROM.
1 = An interrupt will be requested whenever the CCIF, **Figure 3-7**, flag is set.
0 = Command Complete interrupts disabled.

3.3.5 EPROT — EEPROM Protection Register

The EPROT register defines which EEPROM sectors are protected against program or erase.

Register address BASE + $114

	7	6	5	4	3	2	1	0
R	EPOPEN	NV6	NV5	NV4	EPDIS	EP2	EP1	EP0
W								
Reset:	F	F	F	F	F	F	F	F

= Unimplemented or Reserved

Figure 3-6 EEPROM Protection Register (EPROT)

The EPROT register is loaded from EEPROM array address $_FFD during reset, as indicated by the “F” in **Figure 3-6**.

All bits in the EPROT register are readable. Bits NV[6:4] are not writable. The EPOPEN and EPDIS bits in the EPROT register can only be written to the protected state (i.e. 0). The EP[2:0] bits can be written anytime until bit EPDIS is cleared. If the EPOPEN bit is cleared, then the state of the EPDIS and EP[2:0] bits is irrelevant.

To change the EEPROM protection that will be loaded on reset, the upper sector of EEPROM must first be unprotected, then the EEPROM Protect byte located at address $_FFD must be written to.

A protected EEPROM sector is disabled by the EPDIS bit while the size of the protected sector is defined by the EP bits in the EPROT register.

Trying to alter any of the protected areas will result in a protect violation error and bit PVIOL will be set in the EEPROM Status Register ESTAT. A mass erase of a whole EEPROM block is only possible when protection is fully disabled.

EPOPEN — Opens the EEPROM for program or erase.

1 = The EEPROM sectors not protected are enabled for program or erase.

0 = The whole EEPROM array is protected. In this case the EPDIS and EP bits within the protection register are ignored.

EPDIS — EEPROM Protection address range Disable.

The EPDIS bit determines whether there is a protected area in the space of the EEPROM address map.

1 = Protection disabled.

0 = Protection enabled.

EP[2:0] — EEPROM Protection Address Size.

The EP[2:0] bits determine the size of the protected sector. Refer to **Table 3-3**.

Table 3-3 EEPROM Address Range Protection

EP[2:0]	Protected Address Range	Protected Size
000	$_FC0-$_FFF	64 bytes
001	$_F80-$_FFF	128 bytes
010	$_F40-$_FFF	192 bytes
011	$_F00-$_FFF	256 bytes
100	$_EC0-$_FFF	320 bytes
101	$_E80-$_FFF	384 bytes
110	$_E40-$_FFF	448 bytes
111	$_E00-$_FFF	512 bytes

NV[6:4] — Non-Volatile Flag Bits.

These three bits are available to the user as non-volatile flags.

3.3.6 ESTAT — EEPROM Status Register

The ESTAT register defines the EEPROM state machine command status and EEPROM array access, protection and erase verify status.

Register address BASE + $115

	7	6	5	4	3	2	1	0
R	CBEIF	CCIF	PVIOL	ACCERR	0	BLANK	0	0
W								
Reset:	1	1	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 3-7 EEPROM Status Register (ESTAT)

Register bits CBEIF, PVIOL and ACCERR are readable and writable, bits CCIF and BLANK are readable and not writable, bits 3, 1 and 0 read zero and are not writable.

CBEIF — Command Buffer Empty Interrupt Flag.

The CBEIF flag indicates that the address, data and command buffers are empty so that a new command sequence can be started. The CBEIF flag is cleared by writing a “1” to CBEIF. Writing a “0” to the CBEIF flag has no effect on CBEIF but sets ACCERR if written during a command sequence. This bit, CBEIF, is used together with the enable bit CBEIE to generate the interrupt request.

1 = Buffers are ready to accept a new command.

0 = Buffers are full.

CCIF — Command Complete Interrupt Flag.

The CCIF flag indicates that there are no more commands pending. The CCIF flag is cleared when CBEIF is cleared and sets automatically upon completion of all active and pending commands. The CCIF flag does not set when an active command completes and a pending command is fetched from the command buffer. Writing to the CCIF flag has no effect. This bit, CCIF, is used together with the enable bit CCIE to generate the interrupt request.

1 = All commands are completed.
0 = Command in progress.

PVIOL — Protection Violation.

The PVIOL flag indicates an attempt was made to program or erase an address in a protected EEPROM memory area (see **4.1.4 Illegal EEPROM Operations**) . The PVIOL flag is cleared by writing a “1” to PVIOL. Writing a “0” to the PVIOL flag has no effect on PVIOL. While PVIOL is set, it is not possible to launch another command in the EEPROM.

1 = A protection violation has occurred.
0 = No failure.

ACCERR — EEPROM Access Error.

The ACCERR flag indicates an illegal access to the selected EEPROM array (see **4.1.4 Illegal EEPROM Operations**) . This can be either a violation of the command sequence, issuing an illegal command (illegal combination of the CMDBx bits in the ECMD register) or the execution of a CPU STOP instruction while a command is executing (CCIF=0). The ACCERR flag is cleared by writing a “1” to ACCERR. Writing a “0” to the ACCERR flag has no effect on ACCERR. While ACCERR is set, it is not possible to launch another command in the EEPROM.

1 = Access error has occurred.
0 = No failure.

BLANK — Array has been verified as erased.

The BLANK flag indicates that an erase verify command has checked the EEPROM array and found it to be erased. The BLANK flag is cleared by hardware when CBEIF is cleared as part of a new valid command sequence. Writing to the BLANK flag has no effect on BLANK.

1 = EEPROM array verifies as erased.
0 = If an erase verify command has been requested, and the CCIF flag is set, then a zero in BLANK indicates array is not erased.

3.3.7 ECMD — EEPROM Command Register

The ECMD register defines the EEPROM commands.

Register address BASE + $116

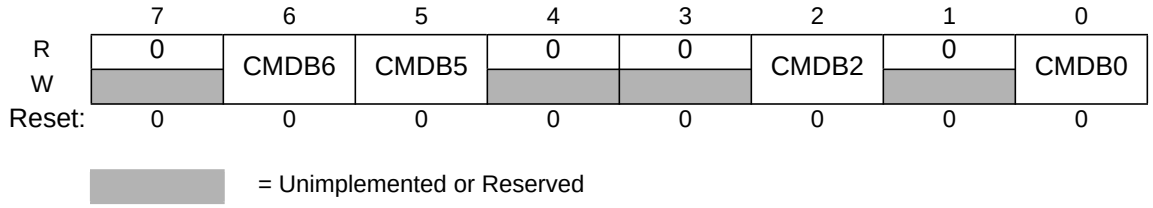

Figure 3-8 EEPROM Command Register (ECMD)

Bits 7, 4, 3 and 1 read zero and are not writable. Bits CMDB6, CMDB5, CMDB2 and CMDB0 are readable and writable during a command sequence.

CMDB — Valid normal mode commands are shown in **Table 3-4**. Any other command than those mentioned in **Table 3-4** sets the ACCERR bit in the ESTAT register (**3.3.6**).

Table 3-4 EEPROM Normal Mode Commands

Command	Meaning
$05	Erase Verify
$20	Word Program
$40	Sector Erase
$41	Mass Erase
$60	Sector Modify

3.3.8 RESERVED3

This register is reserved for factory testing and is not accessible to the user.

Register address BASE + $117

Figure 3-9 RESERVED3

All bits read zero and are not writable.

3.3.9 EADDR — EEPROM Address Register

EADDRHI and EADDRLO are the EEPROM address registers.

Register address Base + $118

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	EABHI		
W								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 3-10 EEPROM Address High Register (EADDRHI)

Register address Base + $119

	7	6	5	4	3	2	1	0
R	EABLO							
W								
Reset:	0	0	0	0	0	0	0	0

Figure 3-11 EEPROM Address Low Register (EADDRLO)

In normal modes, all EADDRHI and EADDRLO bits read zero and are not writable.

In special modes, all EADDRHI and EADDRLO bits are readable and writable except EADDRHI[7:3] which are not writable and always read zero.

For sector erase, the MCU address bits AB[1:0] are ignored.

For mass erase, any address within the block is valid to start the command.

3.3.10 EDATA — EEPROM Data Register

EDATAHI and EDATALO are the EEPROM data registers.

Register address BASE + $11A

	7	6	5	4	3	2	1	0
R	EDHI							
W								
Reset:	0	0	0	0	0	0	0	0

Figure 3-12 EEPROM Data High Register (EDATAHI)

Register address BASE + $11B

	7	6	5	4	3	2	1	0
R W	EDLO							
Reset:	0	0	0	0	0	0	0	0

Figure 3-13 EEPROM Data Low Register (EDATALO)

In normal modes, all EDATAHI and EDATALO bits read zero and are not writable.

In special modes, all EDATAHI and EDATALO bits are readable and writable.

Section 4 Functional Description

4.1 Program and Erase Operation

Write and read operations are both used for the program and erase algorithms described in this section. These algorithms are controlled by a state machine whose timebase EECLK is derived from the oscillator clock via a programmable divider. The command register as well as the associated address and data registers operate as a buffer and a register (2-stage FIFO) so that a new command along with the necessary data and address can be stored to the buffer while the previous command is still in progress. The pipelined operation allows a simplification of command launching. Buffer empty as well as command completion are signalled by flags in the EEPROM status register. Interrupts for the EEPROM will be generated if enabled.

The next four subsections describe:

- How to write the ECLKDIV register.
- The write sequences used to program and erase the EEPROM, but also to perform more sophisticated commands like sector modify and erase verify.
- Valid EEPROM commands.
- Errors resulting from illegal EEPROM operations.

4.1.1 Writing the ECLKDIV Register

Prior to issuing any program or erase command, it is first necessary to write the ECLKDIV register to divide the oscillator down to within 150kHz to 200kHz range. The program and erase timings are also a function of the bus clock, such that the ECLKDIV determination must take this information into account. If we define:

- EECLK as the clock of the EEPROM timing control block
- Tbus as the period of the bus clock
- INT(x) as taking the integer part of x (e.g. INT(4.323)=4),

then ECLKDIV register bits PRDIV8 and EDIV[5:0] are to be set as described in **Figure 4-1**.

For example, if the oscillator clock is 950kHz and the bus clock is 10MHz, ECLKDIV bits EDIV[5:0] should be set to 4 (binary 000100) and bit PRDIV8 set to 0. The resulting EECLK is then 190kHz. As a result, the EEPROM algorithm timings are increased over optimum target by:

$$(200 - 190)/200 \times 100 = 5\%$$

NOTE

Command execution time will increase proportionally with the period of EECLK.

WARNING

Because of the impact of clock synchronization on the accuracy of the functional timings, programming or erasing the EEPROM cannot be performed if the bus clock runs at less than 1 MHz. Programming the EEPROM with an oscillator clock < 150kHz should be avoided. Setting ECLKDIV to a value such that EECLK < 150kHz can reduce the lifetime of the EEPROM due to overstress. Setting ECLKDIV to a value such that (1/EECLK+Tbus) < 5μs can result in incomplete programming or erasure of the memory array cells.

If the ECLKDIV register is written, the bit EDIVLD is set automatically. If this bit is zero, the register has not been written since the last reset. Program and erase commands will not be executed if this register has not been written to.

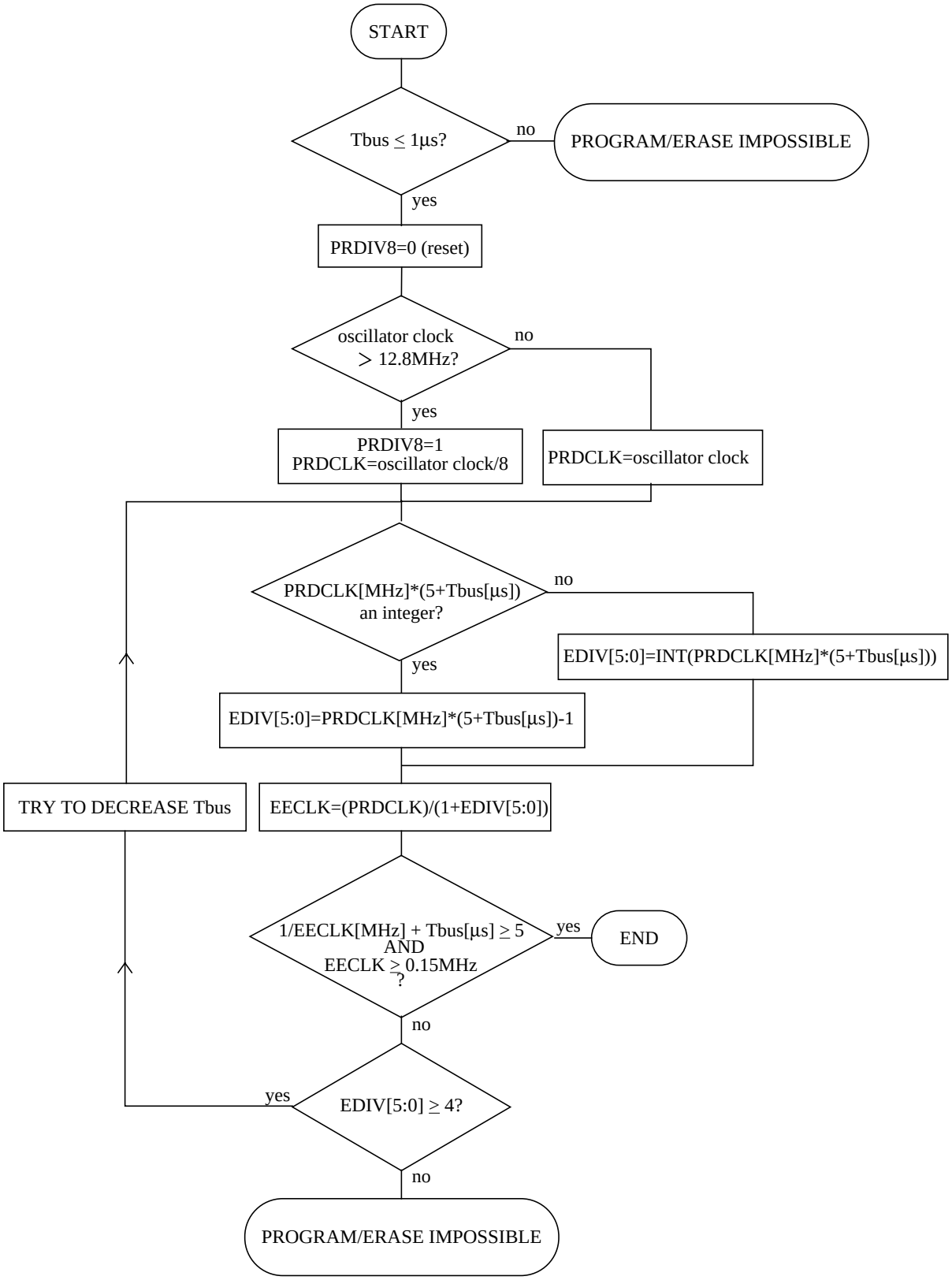

Figure 4-1 PRDIV8 and EDIV bits Determination Procedure

4.1.2 Program and Erase

A Command State Machine is used to supervise the write sequencing for program and erase. More specialized commands like sector modify or erase verify follow the same flow. Before starting a command sequence, it is necessary to check that there is no pending access error or protection violation (the ACCERR and PVIOL flags should be cleared in the ESTAT register).

After this initial step, the CBEIF flag should be tested to ensure that the address, data and command buffers are empty. If so, the command sequence can be started. The following 3-step command write sequence must be strictly adhered to and no intermediate access to the EEPROM array is permitted between the 3 steps. It is possible to read any EEPROM register during a command sequence. The command sequence is as follows:

1. Write the aligned data word to be programmed to the valid EEPROM address space. The address and data will be stored in internal buffers. For program, all address bits are valid. For erase, the value of the data bytes is don't care. For mass erase, the address can be anywhere in the available address space of the array. For sector erase, the address bits[1:0] are ignored.
2. Write the program or erase command to the command buffer. These commands are listed in **Table 4-1**.
3. Clear the CBEIF flag by writing a "1" to it to launch the command. When the CBEIF flag is cleared, the CCIF flag is cleared by hardware indicating that the command was successfully launched. The CBEIF flag will be set again indicating the address, data and command buffers are ready for a new command sequence to begin.

The completion of the command is indicated by the CCIF flag setting. The CCIF flag only sets when all active and pending commands have been completed.

NOTE

The Command State Machine will flag errors in program or erase write sequences by means of the ACCERR (access error) and PVIOL (protection violation) flags in the ESTAT register. An erroneous command write sequence will abort and set the appropriate flag. If set, the user must clear the ACCERR or PVIOL flags before commencing another command write sequence. By writing a 0 to the CBEIF flag the command sequence can be aborted after the word write to the EEPROM address space or after writing a command to the ECMD register and before the command is launched. Writing a "0" to the CBEIF flag in this way will set the ACCERR flag.

A summary of the program algorithm is shown in **Figure 4-2**. For the erase algorithm, the user writes either a mass erase or sector erase command to the ECMD register.

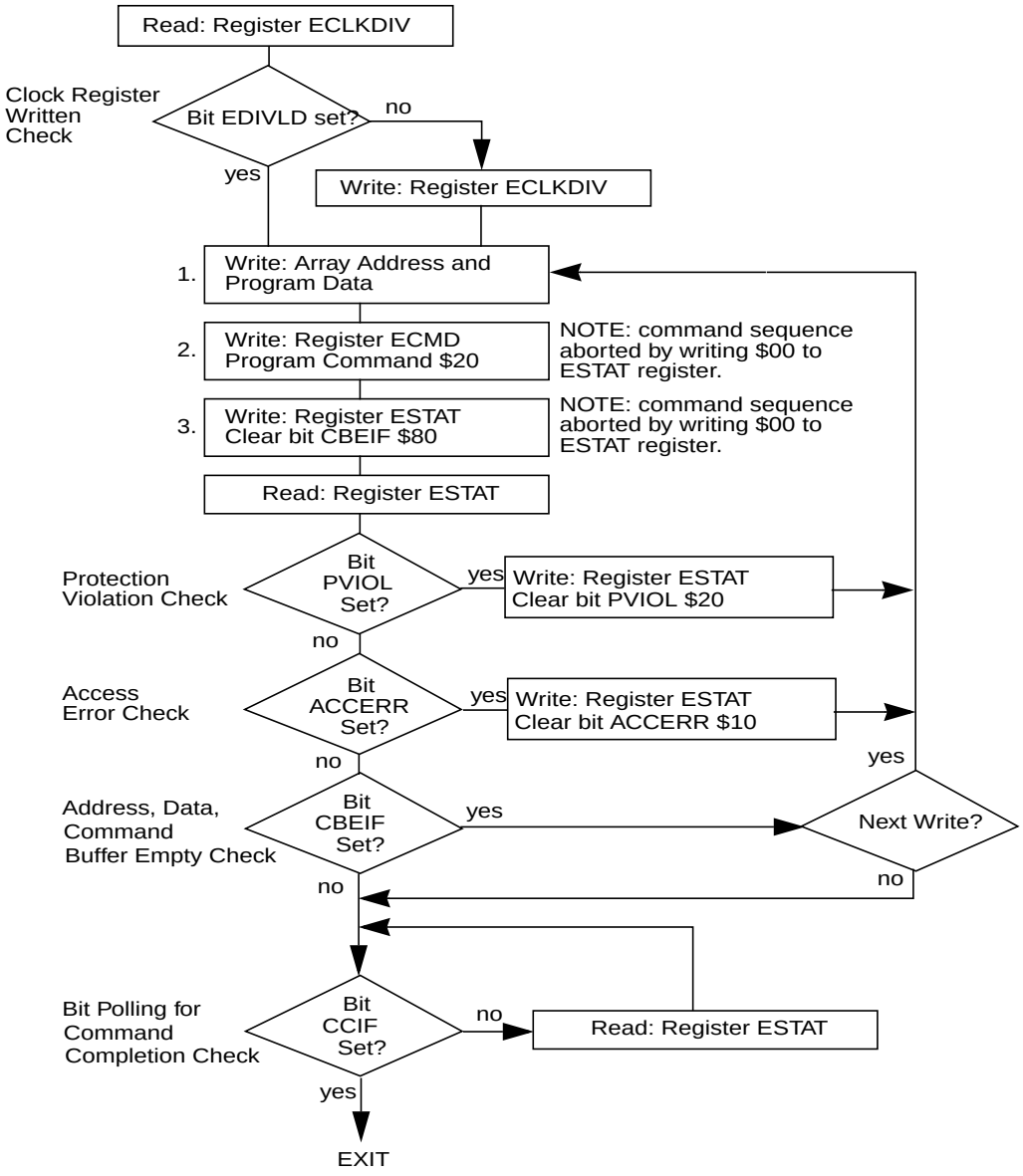

Figure 4-2 Example Program Algorithm

4.1.3 Valid EEPROM Commands

Table 4-1 summarizes the valid EEPROM commands. Also shown are the effects of the commands on the EEPROM array.

Table 4-1 Valid EEPROM Commands

ECMD	Meaning	Function on EEPROM Array
$05	Erase Verify	Verify all memory bytes of the EEPROM array are erased. If the array is erased, the BLANK bit will set in the ESTAT register upon command completion.
$20	Program	Program a word (two bytes).
$40	Sector Erase	Erase two words (four bytes) of EEPROM array.
$41	Mass Erase	Erase all of the EEPROM array. A mass erase of the full array is only possible when EPDIS and EPOPEN are set.
$60	Sector Modify	Erase two words of EEPROM, re-program one word.

WARNING

It is not permitted to program an EEPROM word without first erasing the sector in which that word resides.

The sector modify command executes a two-step algorithm which first erases a sector (2 words) of EEPROM array and then re-programs one of the words in that sector. The EEPROM sector which is erased by the sector modify command is the sector containing the address of the aligned array write which starts the valid command sequence. That same address is re-programmed with the data that was written. By launching a sector modify command and then pipelining a program command, it is possible to completely replace the contents of an EEPROM sector.

4.1.4 Illegal EEPROM Operations

The ACCERR flag will be set during the command write sequence if any of the following illegal operations are performed causing the command write sequence to immediately abort:

1. Writing to the EEPROM address space before initializing ECLKDIV.
2. Writing a misaligned word or a byte to the valid EEPROM address space.
3. Writing to the EEPROM address space while CBEIF is not set.
4. Writing a second word to the EEPROM address space before executing a program or erase command on the previously written word.
5. Writing to any EEPROM register other than ECMD after writing a word to the EEPROM address space.
6. Writing a second command to the ECMD register before executing the previously written

command.

7. Writing an invalid command to the ECMD register in normal mode.
8. Writing to any EEPROM register other than ESTAT (to clear CBEIF) after writing to the command register (ECMD).
9. The part enters STOP mode and a program or erase command is in progress. The command is aborted and any pending command is killed.
10. A “0” is written to the CBEIF bit in the ESTAT register.

The ACCERR flag will not be set if any EEPROM register is read during the command sequence.

If the EEPROM array is read during execution of an algorithm (i.e. CCIF bit in the ESTAT register is low), the read will return non-valid data and the ACCERR flag will not be set.

When an ACCERR flag is set in the ESTAT register, the Command State Machine is locked. It is not possible to launch another command until the ACCERR flag is cleared.

The PVIOL flag will be set during the command write sequence after the word write to the EEPROM address space and the command sequence will be aborted if any of the following illegal operations are performed.

1. Writing a EEPROM address to program in a protected area of the EEPROM.
2. Writing a EEPROM address to erase in a protected area of the EEPROM.
3. Writing the mass erase command to ECMD while any protection is enabled.

When the PVIOL flag is set in the ESTAT register the Command State Machine is locked. It is not possible to launch another command until the PVIOL flag is cleared.

4.2 Wait Mode

When the MCU enters WAIT mode and if any command is active (CCIF=0), that command and any pending command will be completed.

The EETS4K module can recover the MCU from WAIT if the interrupts are enabled (see **Section 6**).

4.3 Stop Mode

If a command is active (CCIF = 0) when the MCU enters the STOP mode, the command will be aborted, and the data being programmed or erased is lost. The high voltage circuitry to the EEPROM array will be switched off when entering STOP mode. CCIF and ACCERR flags will be set. Upon exit from STOP, the CBEIF flag is set and any pending command will not be executed. The ACCERR flag must be cleared before returning to normal operation.

WARNING

As active commands are immediately aborted when the MCU enters STOP mode, it is strongly recommended that the user does not use the STOP command during program and erase execution.

4.4 Background Debug Mode

In Background Debug Mode (BDM), the EPROT register is writable. If the chip is unsecured then all EEPROM commands listed in **Table 4-1** can be executed. In special single chip mode if the chip is secured then the only possible command to execute is mass erase.

Section 5 Resets

5.1 General

If a reset occurs while any command is in progress that command will be immediately aborted. The state of the word being programmed or the sector / block being erased is not guaranteed.

Section 6 Interrupts

6.1 General

The EETS4K block can generate an interrupt when all commands are completed or the address, data and command buffers are empty.

Table 6-1 EEPROM Interrupt Sources

Interrupt Source	Interrupt Flag	Local Enable	Global (CCR) Mask
EEPROM Address, Data and Command Buffers empty	CBEIF (ESTAT register)	CBEIE	I Bit
All Commands are completed on EEPROM	CCIF (ESTAT register)	CCIE	I Bit

NOTE

Vector addresses and their relative interrupt priority are determined at the MCU level.

6.2 Description of Interrupt Operation

For a detailed description of the register bits, refer to the EEPROM Configuration register and EEPROM Status register sections (respectively **3.3.4** and **3.3.6**).

Block Guide End Sheet

FINAL PAGE OF
38
PAGES