To all our customers

Regarding the change of names mentioned in the document, such as Hitachi Electric and Hitachi XX, to Renesas Technology Corp.

The semiconductor operations of Mitsubishi Electric and Hitachi were transferred to Renesas Technology Corporation on April 1st 2003. These operations include microcomputer, logic, analog and discrete devices, and memory chips other than DRAMs (flash memory, SRAMs etc.) Accordingly, although Hitachi, Hitachi, Ltd., Hitachi Semiconductors, and other Hitachi brand names are mentioned in the document, these names have in fact all been changed to Renesas Technology Corp. Thank you for your understanding. Except for our corporate trademark, logo and corporate statement, no changes whatsoever have been made to the contents of the document, and these changes do not constitute any alteration to the contents of the document itself.

Renesas Technology Home Page: http://www.renesas.com

Renesas Technology Corp.
Customer Support Dept.
April 1, 2003

Cautions

Keep safety first in your circuit designs!

1. Renesas Technology Corporation puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage.
   Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corporation product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corporation or a third party.
2. Renesas Technology Corporation assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corporation without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor for the latest product information before purchasing a product listed herein.
   The information described here may contain technical inaccuracies or typographical errors.
   Renesas Technology Corporation assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
   Please also pay attention to information published by Renesas Technology Corporation by various means, including the Renesas Technology Corporation Semiconductor home page (http://www.renesas.com).
4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corporation assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corporation semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corporation or an authorized Renesas Technology Corporation product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
6. The prior written approval of Renesas Technology Corporation is necessary to reprint or reproduce in whole or in part these materials.
7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
   Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
8. Please contact Renesas Technology Corporation for further details on these materials or the products contained therein.

Flash Application Design Guidelines

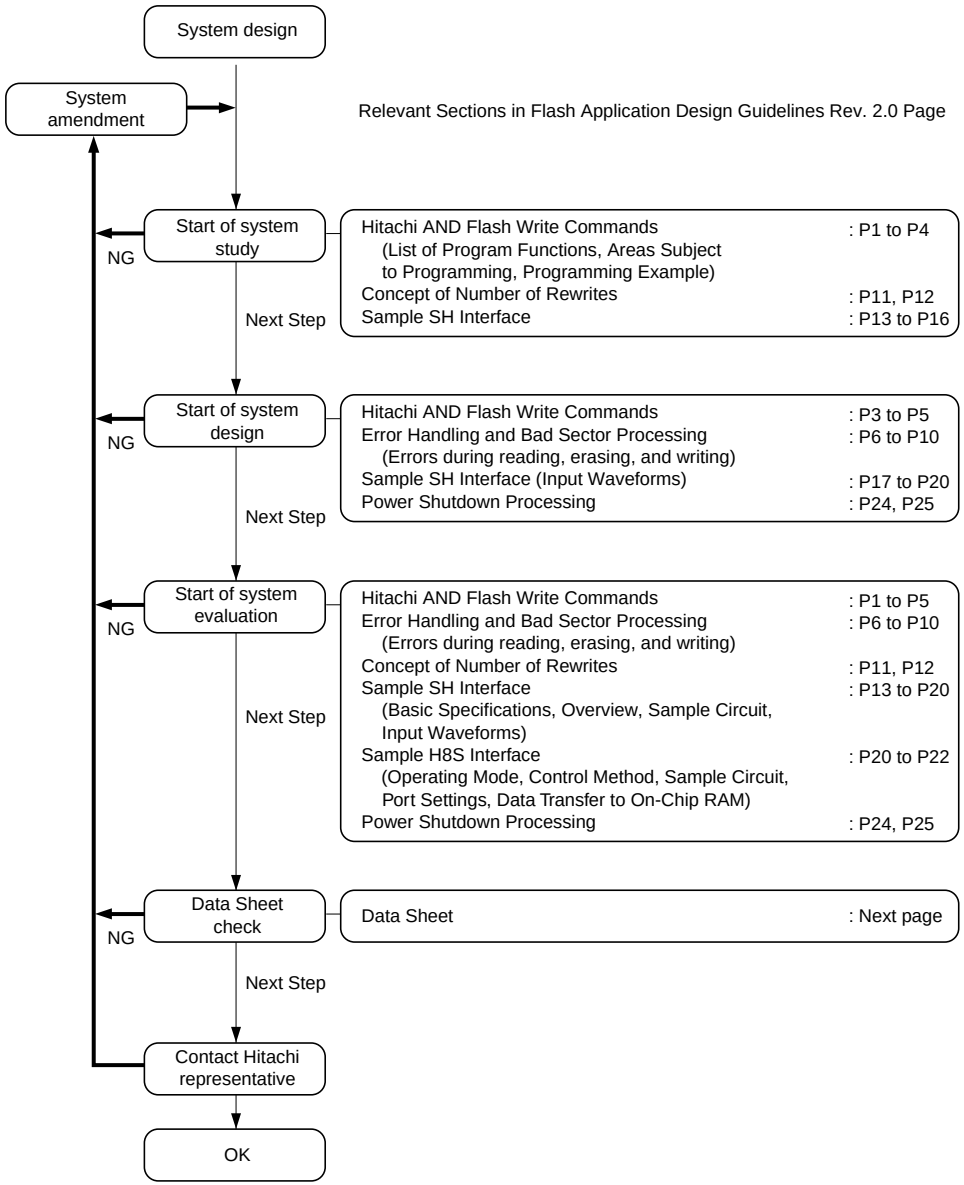
ADE-603-010A
Rev. 2.0
12/5/00
Hitachi, Ltd.

Cautions

1. Hitachi neither warrants nor grants licenses of any rights of Hitachi's or any third party's patent, copyright, trademark, or other intellectual property rights for information contained in this document. Hitachi bears no responsibility for problems that may arise with third party's rights, including intellectual property rights, in connection with use of the information contained in this document.
2. Products and product specifications may be subject to change without notice. Confirm that you have received the latest product standards or specifications before final design, purchase or use.
3. Hitachi makes every attempt to ensure that its products are of high quality and reliability. However, contact Hitachi's sales office before using the product in an application that demands especially high quality and reliability or where its failure or malfunction may directly threaten human life or cause risk of bodily injury, such as aerospace, aeronautics, nuclear power, combustion control, transportation, traffic, safety equipment or medical equipment for life support.
4. Design your application so that the product is used within the ranges guaranteed by Hitachi particularly for maximum rating, operating supply voltage range, heat radiation characteristics, installation conditions and other characteristics. Hitachi bears no responsibility for failure or damage when used beyond the guaranteed ranges. Even within the guaranteed ranges, consider normally foreseeable failure rates or failure modes in semiconductor devices and employ systemic measures such as fail-safes, so that the equipment incorporating Hitachi product does not cause bodily injury, fire or other consequential damage due to operation of the Hitachi product.
5. This product is not designed to be radiation resistant.
6. No one is permitted to reproduce or duplicate, in any form, the whole or part of this document without written approval from Hitachi.
7. Contact Hitachi's sales office for any questions regarding this document or Hitachi semiconductor products.

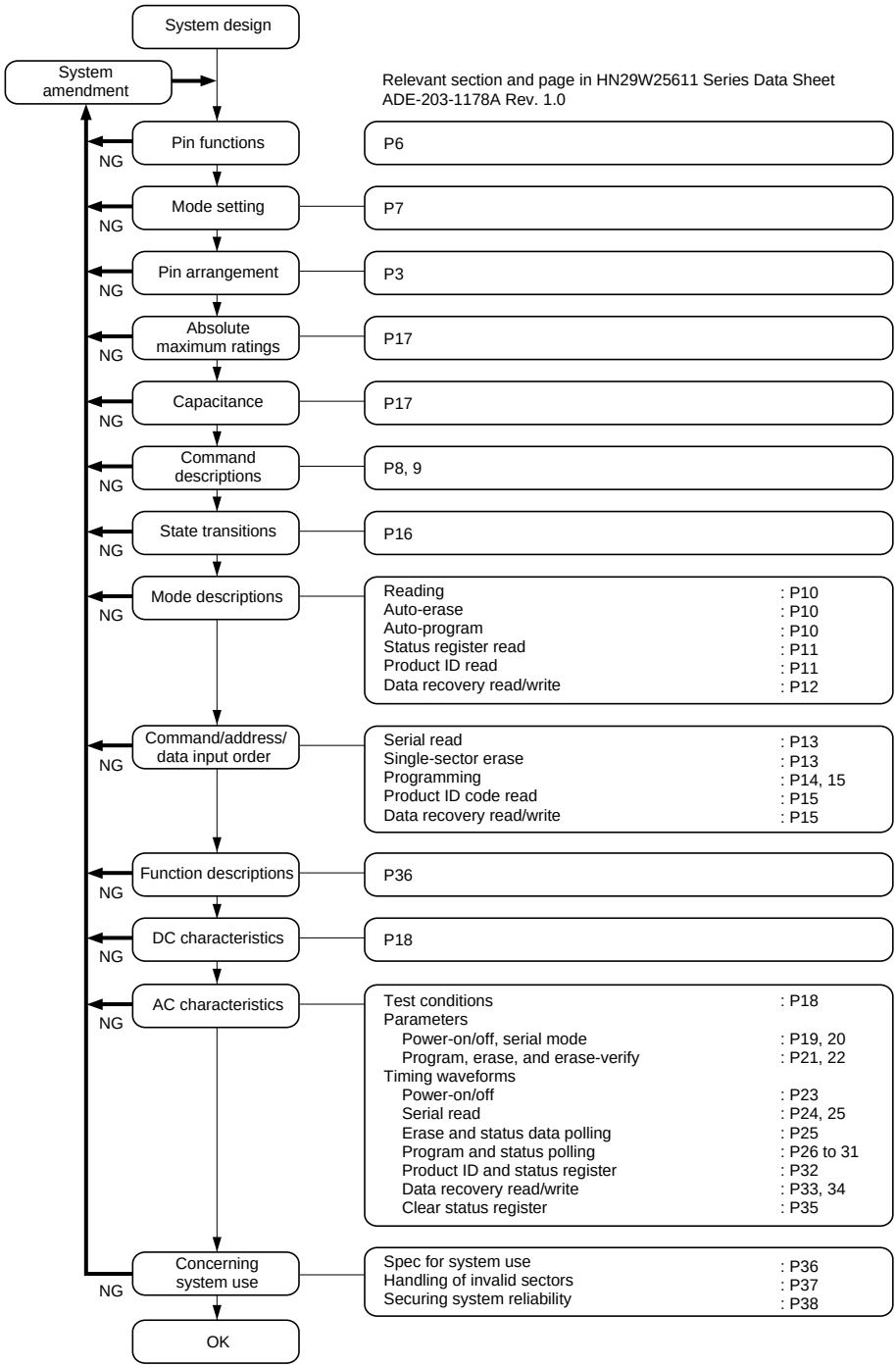
Preface

This document presents flash application guidelines, including points requiring special attention, for developing systems using Hitachi AND flash memory.



Please contact a Hitachi representative in the event of problems relating to flash memory. Early diagnosis is vital for solving problems.

Data Sheet pages for reference during the design of a system using Hitachi AND flash memory are shown below.



Contents

1. Hitachi AND Flash Write Commands

1.1 Overview

This section describes the differences between the four kinds of auto-program commands, Program (1) to Program (4).

Programming operations can be broadly classified into the following three kinds:

- Additional write

  Performs additional writing to arbitrary bytes in the erased state (FFH) within a sector. (Program1, Program3)
- Write

  Performs writing to a sector in the erased state. (Program2)
- Rewrite

  Rewrites data regardless of the write target byte data. (Program4)

1.2 List of Hitachi AND Flash Program Functions

Program No.	Target Area (Column Address CA Range)	Column Address	Function	Figure No.
Program(1) Additional write	Data area (CA0 to CA2047) + Control area (CA2048 to CA2111)	Yes	• Performs additional writes starting at byte indicated by input column address. • Additional writes are possible on unwritten (FFH) bytes. • FFH is input for bytes on which writes are not performed.	Figure 1.2
		No	• Performs additional writes on sector. • Additional writes are possible on unwritten (FFH) bytes. • FFH is input for bytes on which writes are not performed.	Figure 1.3
Program(2) Write	Data area (CA0 to CA2047) + Control area (CA2048 to CA2111)	No	• Performs additional writes on sectors in erased state (all FFH). • FFH is input for bytes on which writes are not performed.	Figure 1.4
Program(3) Control area additional write	Control area (CA2048 to CA2111)	No	• Performs additional writes on control area. • Additional writes are possible on unwritten (FFH) bytes. • FFH is input for bytes on which writes are not performed.	Figure 1.5
Program(4) Rewrite	Data area (CA0 to CA2047) + Control area (CA2048 to CA2111)	Yes	• Performs rewrites starting at byte indicated by input column address. • Rewrites with input data (including FFH data).	Figure 1.6
		No	• Performs rewrites on sector. • Rewrites with input data (including FFH data).	Figure 1.7

Notes on Use of Additional Writes

- As additional writes by Program(1) and Program(3) are also counted in the number of rewrites, the stipulated number of rewrites should not be exceeded. For the method of counting rewrites, see section 3, Concept of Number of Rewrites.
- Additional rewrites are performed in byte units.

1.3 Areas Subject to Hitachi AND Flash Programming

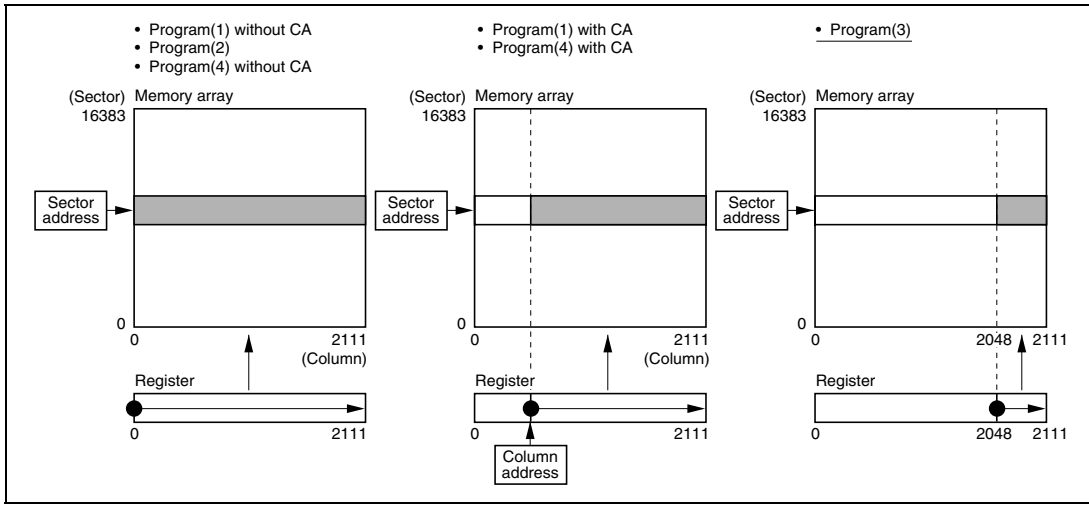


Figure 1.1 Areas Subject to Hitachi AND Flash Programming

1.4 Examples of Use of Hitachi AND Flash Program Modes

Examples of the use of the various program modes are shown below.

1. Program(1) with CA (additional write)
   Writes are performed on bytes with a value of FFH. (Byte-unit additional writes)
   Note: When FFH is input, the result is no change to the existing stored data.

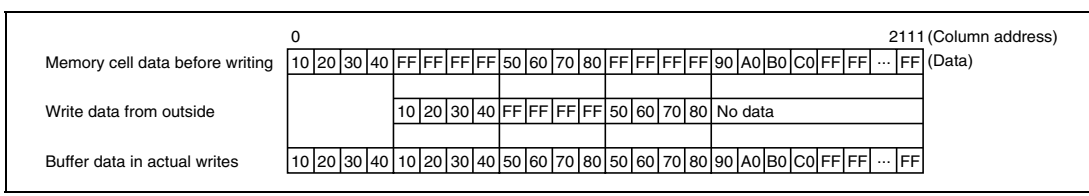


Figure 1.2 Program Mode Example 1

2. Program(1) without CA (additional write)

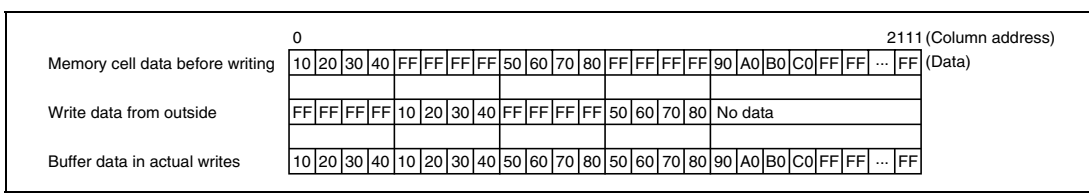


Figure 1.3 Program Mode Example 2

3. Program(2) (write for pre-erased sectors)

   Writing is performed on sectors in the erased state.

   Note: When FFH is input, the result is no change to the existing stored data.

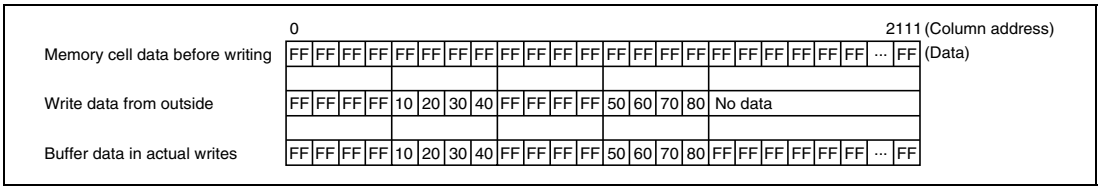


Figure 1.4 Program Mode Example 3

4. Program(3) (additional write to control area)

   Writing is performed on bytes with a value of FFH in the control area.

   Note: When FFH is input, the result is no change to the existing stored data.

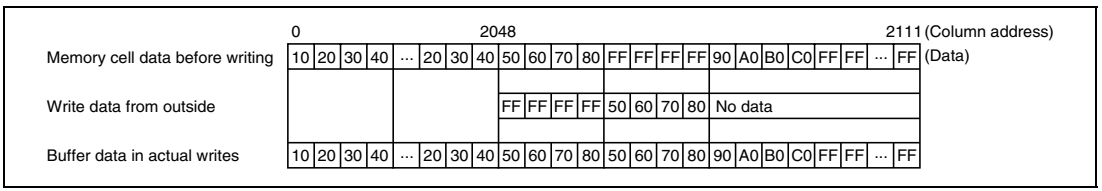


Figure 1.5 Program Mode Example 4

5. Program(4) with CA (rewrite)

   Performs rewriting.

   Addresses for which data has been entered from outside are rewritten. (FFH data rewriting is also performed.)

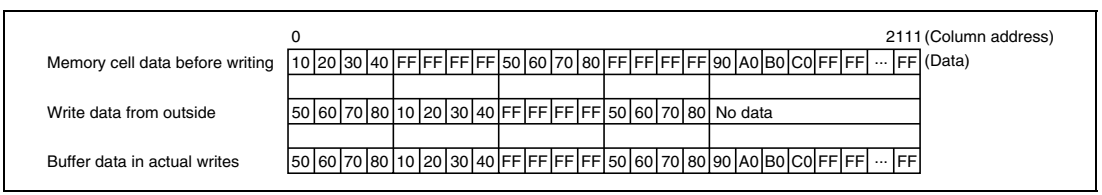


Figure 1.6 Program Mode Example 5

6. Program(4) without CA (rewrite)

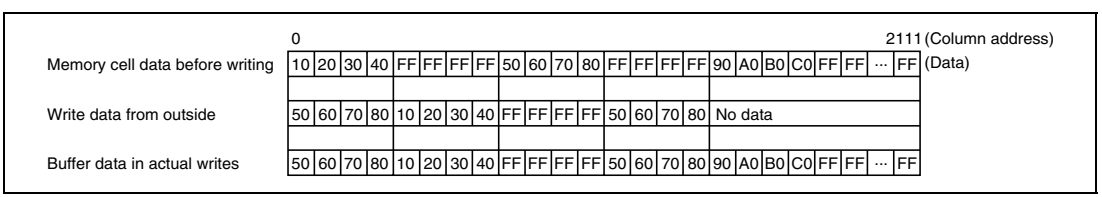


Figure 1.7 Program Mode Example 6

Conditions for the use of the Program(1) to Program(4) modes are summarized below.

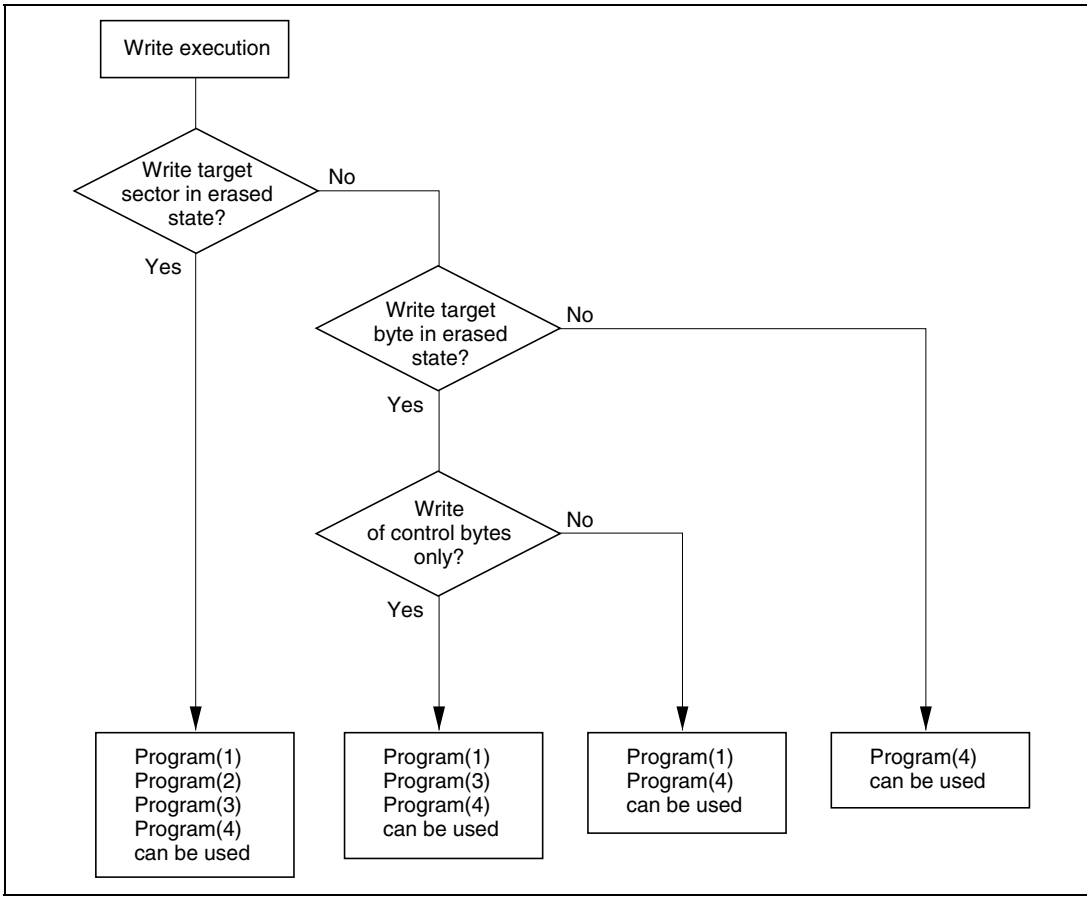


Figure 1.8 Conditions for Use of Each Program Mode

2. Error Handling and Bad Sector Processing

A read, erase, or rewrite error may occur when Hitachi AND flash memory is programmed. To secure system reliability, provisions must be made to handle such errors when they occur.

The approach to error handling and bad sector processing is outlined below. The most appropriate processing should be incorporated into the system after checking details of flash memory state transitions, status information, etc., in the Data Sheet.

2.1 Errors during Reading

When a data read is performed, the read data may differ from the written data. Therefore, an ECC correction function of 3 or more bits per sector (2 kbytes) should be provided.

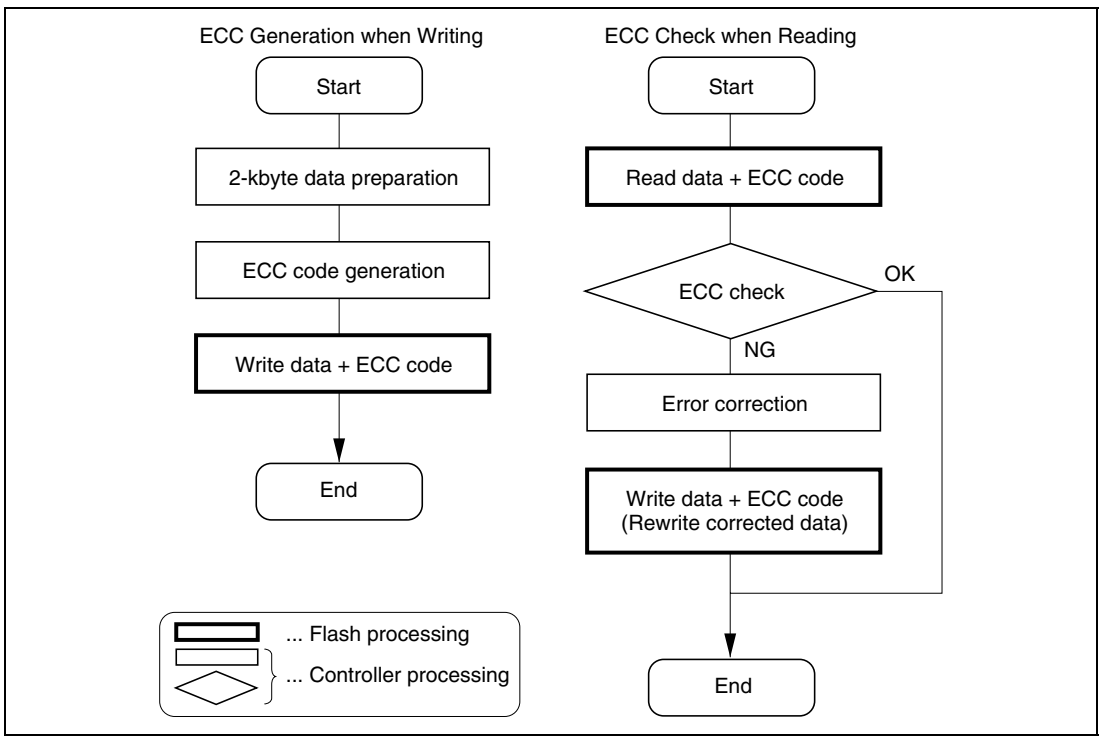


Figure 2.1 Example of Read Error Handling

2.2 Errors during Erasing

The main points for attention when an error occurs during erasing are given below.

(1) Factory shipped bad sectors, or sectors that have become bad due to erasing or writing, should not be erased or written.

(2) Carry out a good sector code check before writing or erasing. One option may be to create a check table in which such sectors are recorded.

(3) Data in a sector in which an error has occurred during erasing will be undefined, and meaningless data will be returned if the sector is read.

(4) If an error occurs, record the bad sector, and then issue a Clear Status Register command and restore the flash memory to the state in which commands can be received.

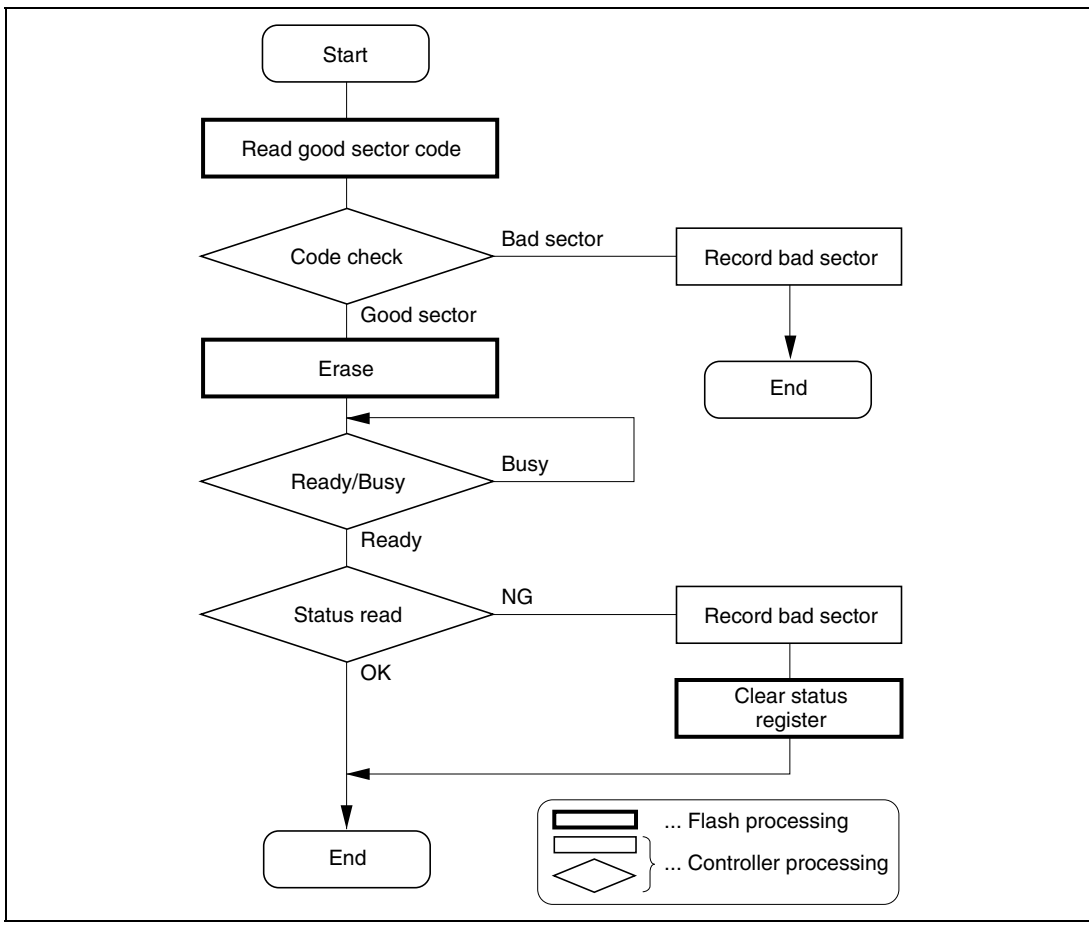


Figure 2.2 Example of Erase Error Handling

2.3 Errors during Writing

The main points for attention when an error occurs during writing are given below.

(1) Factory shipped bad sectors, or sectors that have become bad due to erasing or writing, should not be erased or written.

(2) Carry out a good sector code check before writing or erasing. One option may be to create a check table in which such sectors are recorded.

(3) Error confirmation can be achieved by performing a status register read after writing.

(4) Data in a sector in which a write error has occurred will be indeterminate, and should not be used for rewriting to a spare sector.

(5) Any of the following three methods can be used for rewriting.

(a) Writing data reloaded from an external buffer (See figure 2.4.)
Requires host to maintain data buffer until successful completion of write command.

(b) Data recovery read (See figure 2.5.)

(c) Data recovery write (See figure 2.6.)

(6) Do not perform any further accesses to a sector in which an error has occurred.

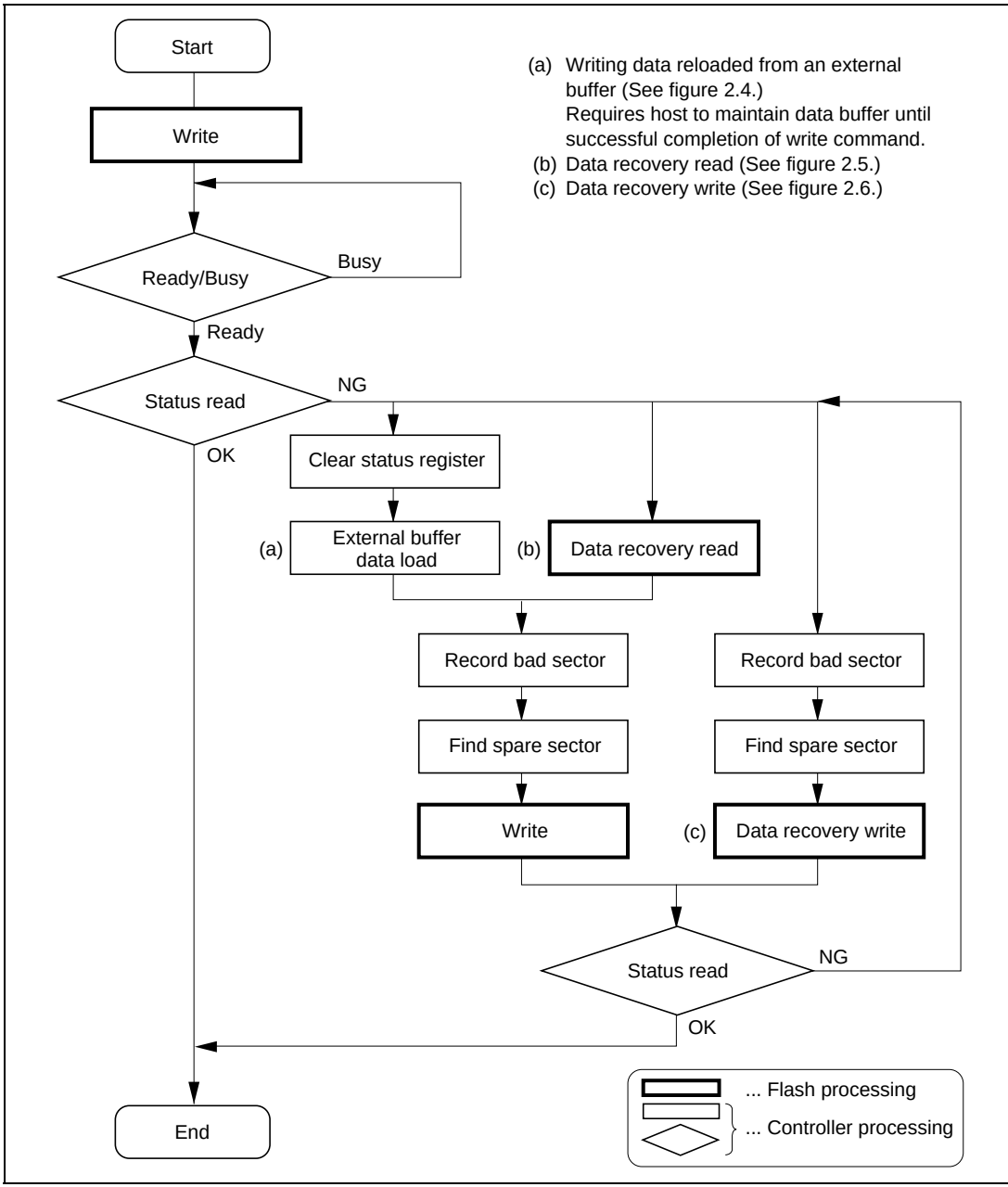


Figure 2.3 Example of Write Error Handling

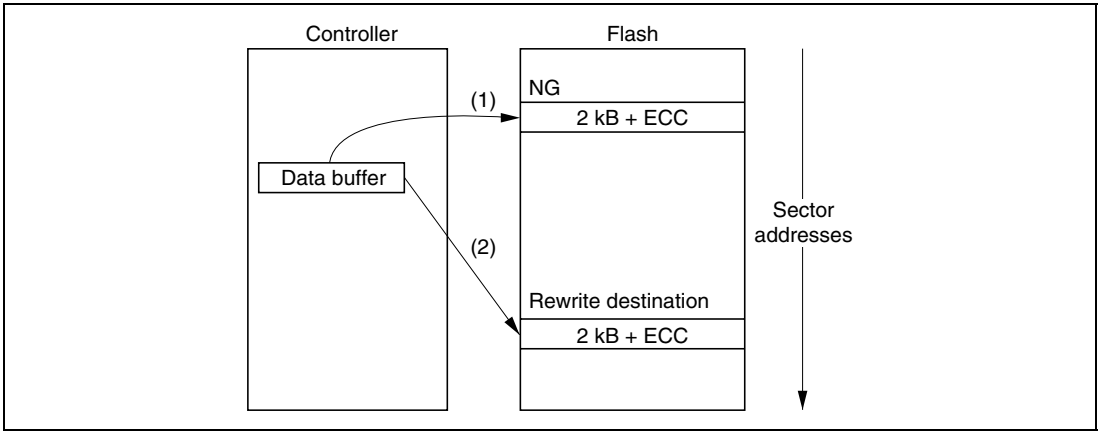


Figure 2.4 Writing Data Reloaded from an External Buffer

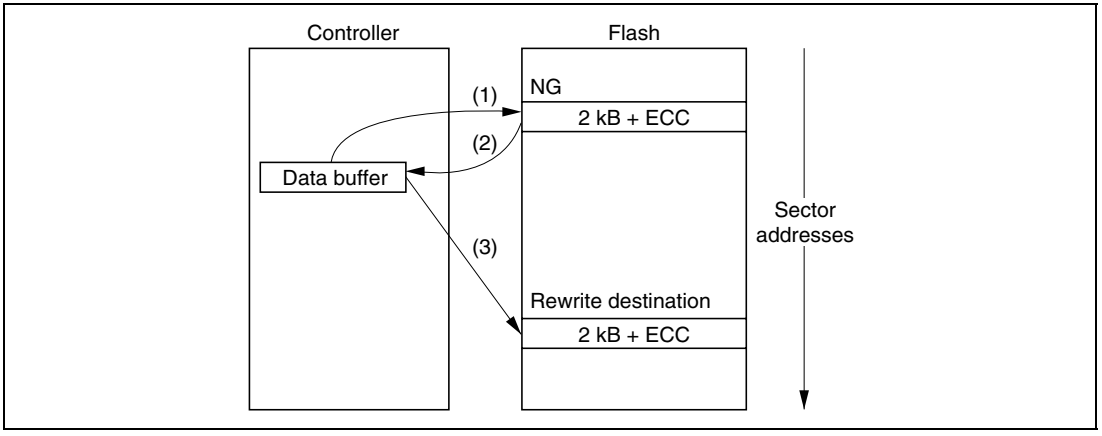


Figure 2.5 Data Recovery Read

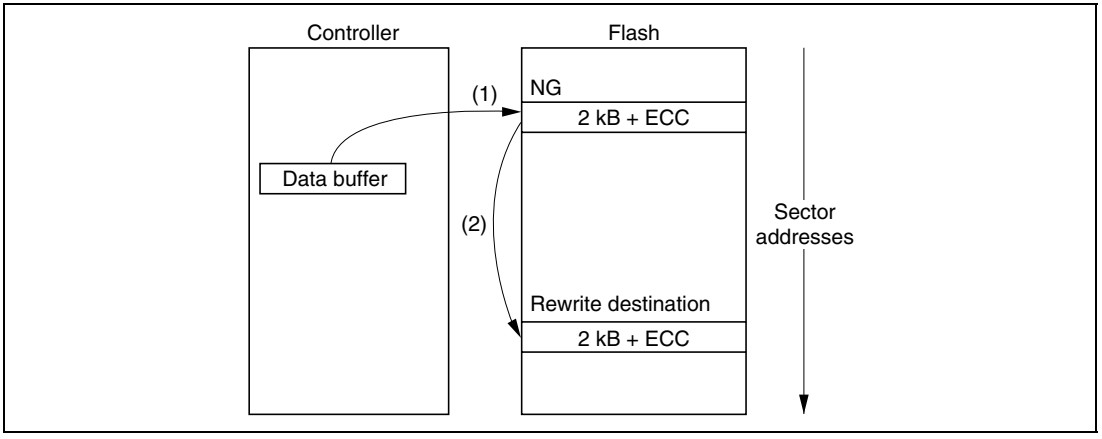


Figure 2.6 Data Recovery Write

3. Concept of Number of Rewrites

3.1 Definition of Number of Rewrites

This is calculated by totaling the number of operations for each process per sector unit (see below).

Erase + Write	1.0
Additional write	1.0
Rewrite	1.0

3.2 Definition of Bad Sector Occurrence Rate [= 1.8(%)]

The probability of a defect occurring when the same sector is rewritten 300,000 times. (See figure 3.1.)

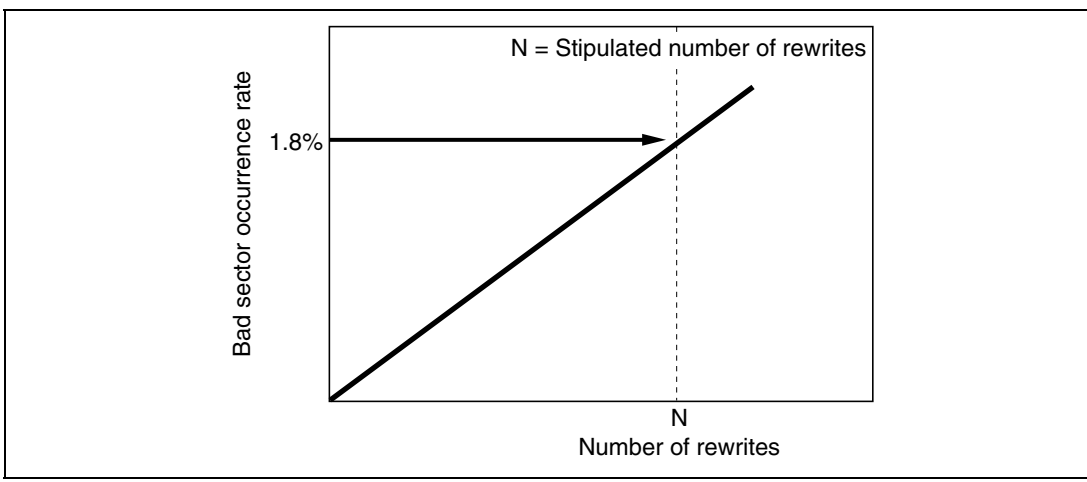


Figure 3.1 Definition of Bad Sector Occurrence Rate

3.3 Sample Calculation of Number of Spare Sectors

Example: Number of spare sectors needed when the entire area of 98% MGM flash is rewritten 300,000 times
Defect rate of 1.8(%) for 300,000 rewrites with total of 16,384 sectors and MGM rate of 98(%)

Number of spare sectors needed = $16{,}384 \times 0.98 \times 0.018 = 290$ (sectors) or more

3.4 Notes on Number of Rewrites

(1) Do not exceed the stipulated number of rewrites on the same sector.

(2) If the stipulated number of rewrites is exceeded, unexpected errors may occur in that sector.

(3) Execute processing on the system side to prevent a system crash in the event of such an error.

4. Sample Microcomputer Interfaces

4.1 Sample SH Interface

This section shows examples of the interface to a microcomputer. Operation of the sample SH interface circuit has been confirmed with an SH7709.

4.1.1 Basic Specifications

Embedded flash (AND) control is performed from the SH by means of simple flash control logic (TTLIC, GA, etc.).

(1) Interfacing is performed via area n (CSn).

(2) A register to control the flash pins is provided at an area n (CSn) address (Address: AP). (External ports: $\overline{CE}$, $\overline{WE}$, $\overline{CDE}$, SC)

(3) Flash is accessed by area n (CSn) address (Address: AF).

(4) 2.5 kbytes of RAM are necessary (5 kbytes recommended) as a buffer for providing a spare sector in the event of a flash memory write or read error.

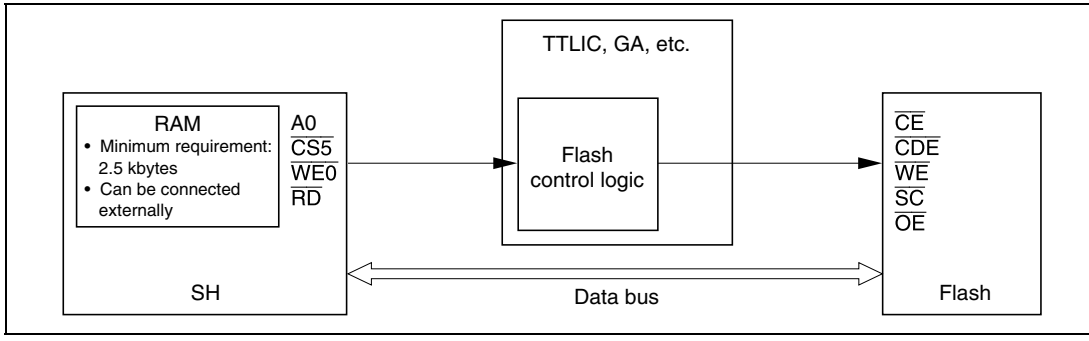


Figure 4.1 Flash Memory Interface Block Configuration

4.1.2 Overview

(1) $\overline{CDE}$ performs command/address switching, and so can be low during polling.

(2) $\overline{CE}$ disables all input (including SC), so is overlapped with $\overline{WE}$, SC, etc.

(3) Operation is not started by a command not stipulated in the Data Sheet.

Ex: 02H, 03H, ... 07H, etc.

(4) SC becomes valid after the prescribed procedure.

After SA(2) + first access when reading; after SA(2) $\overline{CDE}$ fall when writing.

(5) Polling is performed by I/O7. (R/B not used.)

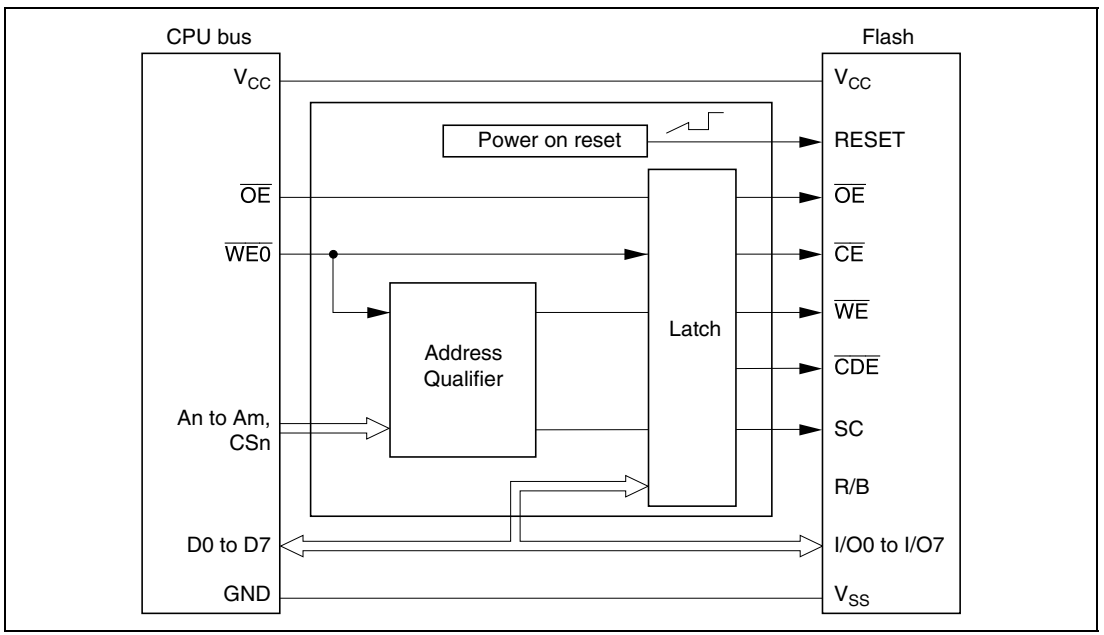


Figure 4.2 Flash Memory Interface Block Configuration

Description of Blocks

- Address Qualifier

  Selects whether an input value from the data bus is to be latched as a signal for control signal setting, or the latch data is to be held.

  During the latch data hold period, control signals are output to the flash memory, so command, sector address, or other input is performed from the data bus to the flash memory.

- Latch

  Latches data input from the data bus, and provides the necessary control signal to the flash memory on the basis of that latched data.

4.1.3 Sample Circuit

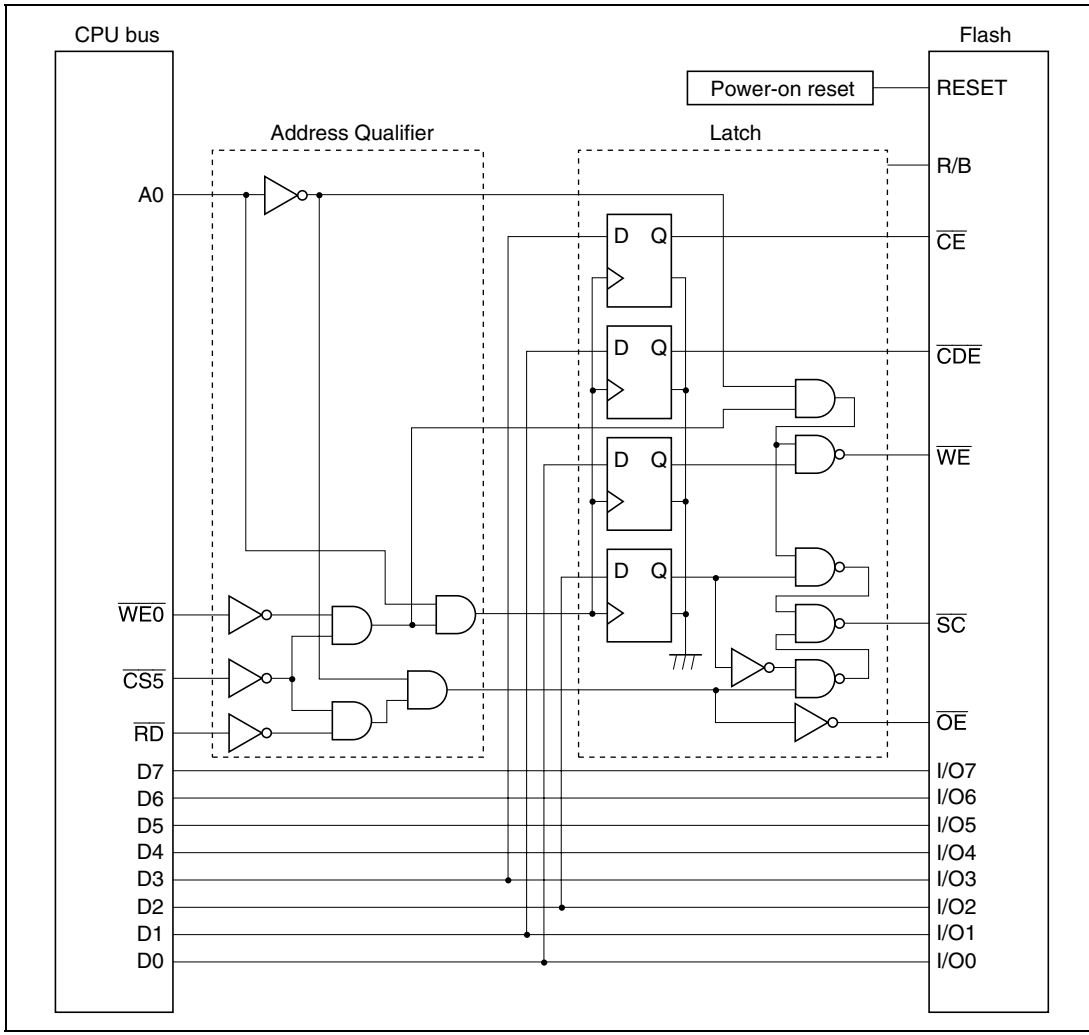


Figure 4.3 Flash Control Logic

Various kinds of processing are performed on the flash memory by combining the states in (1) and (2) below (see section 4.1.4, Input Waveforms, for the flow).

(1) Address A0 = High & $\overline{CSn}$ = Low

Flash control signals are set in the latch (command setting, sector address setting, data read/write $\overline{CE}$, $\overline{OE}$, $\overline{WE}$, $\overline{CDE}$, etc.).

	Latch Input Signal Name	Latch Data	Output Signal State
1)	D0	1;	$\overline{WE0} \rightarrow \overline{WE}$
		0;	$\overline{WE}$ Fixed high
2)	D1	1;	$\overline{CDE}$ Fixed high
		0;	$\overline{CDE}$ Fixed low
3)	D2	1;	$\overline{WE0} \rightarrow$ SC
		0;	$\overline{RD} \rightarrow$ SC
4)	D3	1;	$\overline{CE}$ Fixed high
		0;	$\overline{CE}$ Fixed low

(2) Address A0 = Low & $\overline{CSn}$ = High

Latch hold state; command setting and data read/write access executed on the flash memory.

4.1.4 Input Waveforms

Sample input waveforms in read, program, and erase operations are shown below.

- Serial Read (1) → (2)
- Program (1) → (3)
- Erase (1) → (4)

(1) Command and Address Setting

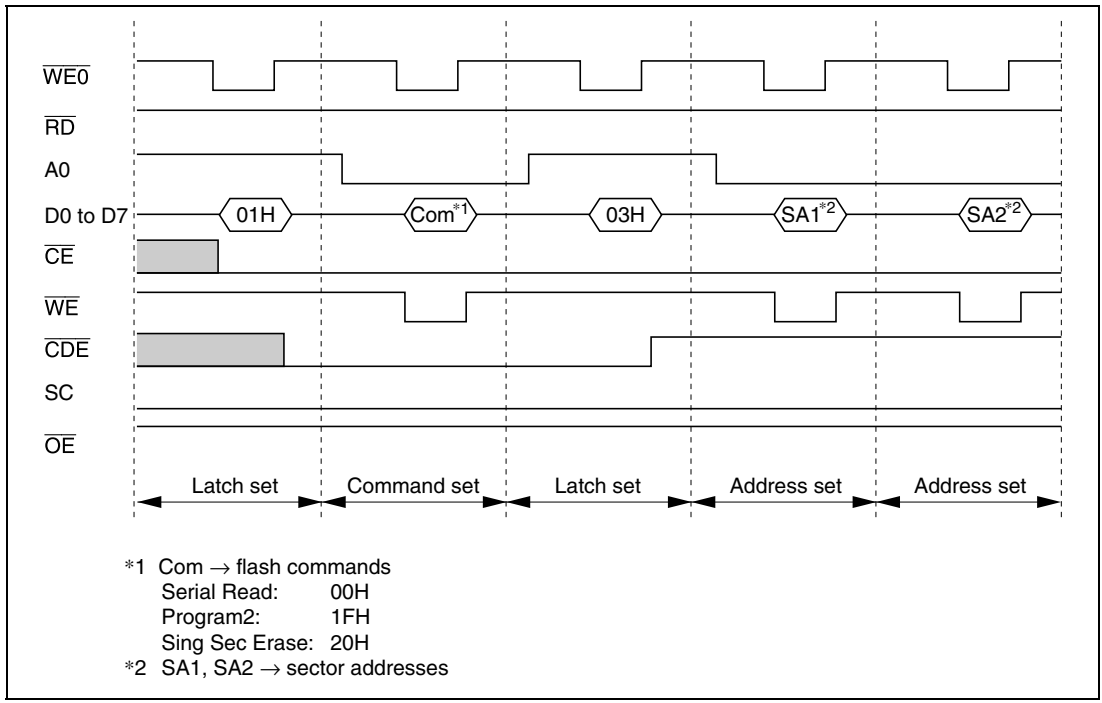


Figure 4.4 Example of Timing Waveforms in Command and Address Setting

(2) Read

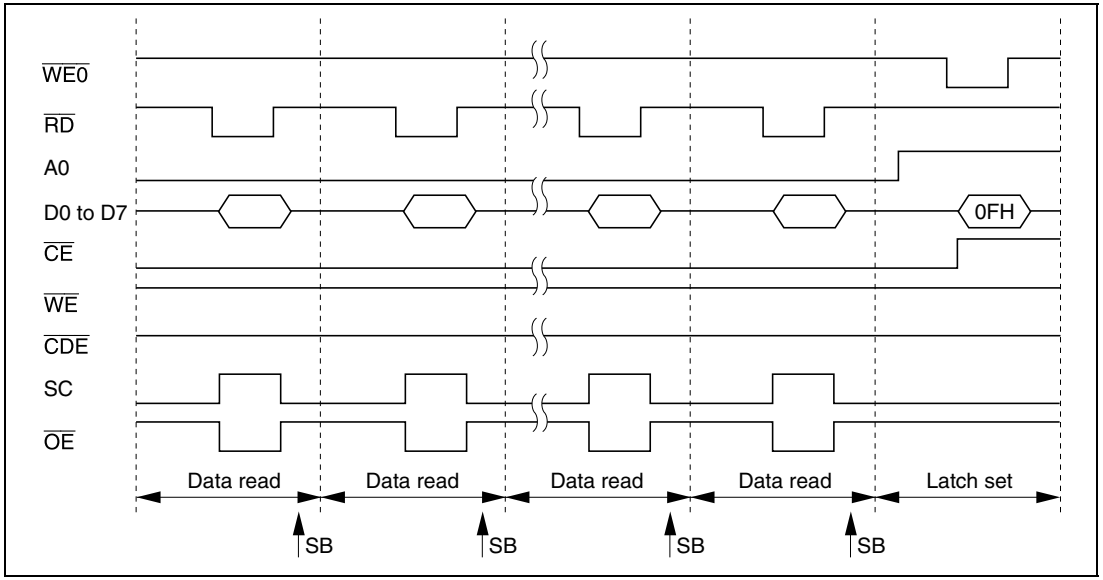


Figure 4.5 Example of Read Timing Waveforms

- Read flow

I/O	A0	D0 to D7
O	1	0FH (Clear)
O	1	01H (Com Latch)
O	0	00H (Serial Read)
O	1	03H (Add Latch)
O	0	AAH (Sector Add 1)
O	0	AAH (Sector Add 2)
I	0	Data (Read Data)
:	:	:
O	1	0FH (Clear)

(3) Program

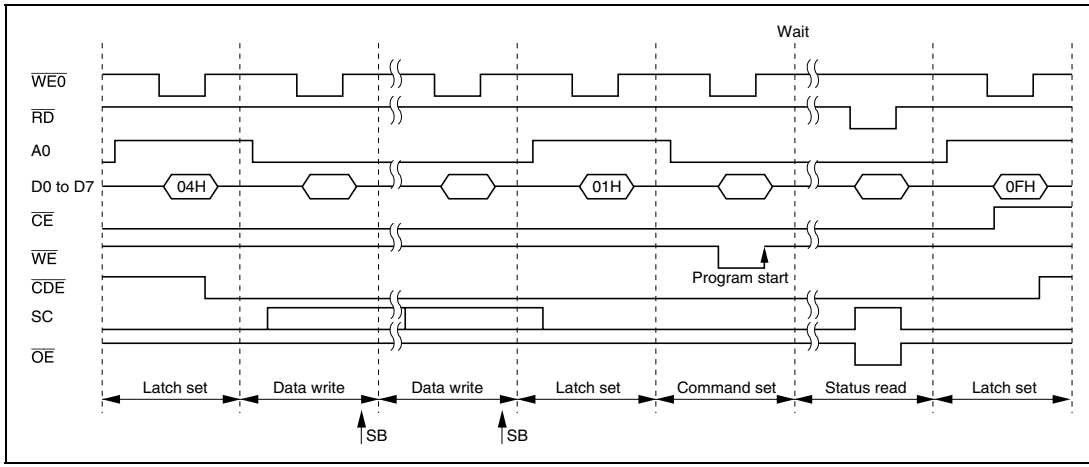


Figure 4.6 Example of Program Timing Waveforms

- Programming flow

I/O	A0	D0 to D7
O	1	0FH (Clear)
O	1	01H (Com Latch)
O	0	11H (Program 4)
O	1	03H (Add Latch)
O	0	AAH (Sector Add 1)
O	0	AAH (Sector Add 2)
O	1	04H (SC Latch)
O	0	DDH (Data Input)
:	:	:
O	1	01H (Com Latch)
O	0	40H (Prg Start) (Status Read)
O	1	0FH (Clear)

(4) Erase

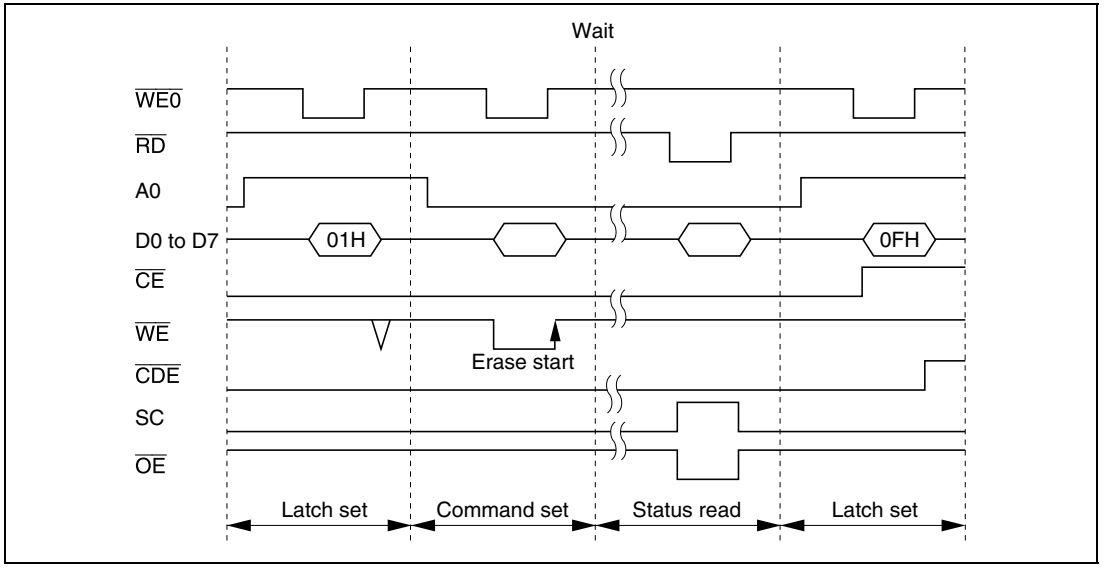


Figure 4.7 Example of Erase Timing Waveforms

- Erase flow

I/O	A0	D0 to D7
O	1	0FH (Clear)
O	1	01H (Com Latch)
O	0	20H (Single Erase)
O	1	03H (Add Latch)
O	0	AAH (Sector Add 1)
O	0	AAH (Sector Add 2)
O	1	01H (Com Latch)
O	0	B0H (Erase Start) (Status Read)
O	1	0FH (Clear)

4.2 Sample H8S Interface

4.2.1 Operating Mode

Mode 3/EXPE = 0 normal single-chip mode is used.

4.2.2 Control Method

Control is performed using ports in order to implement simple signal connection and control.

Port 1 is used for signal control.

Port 2 is used for data input/output.

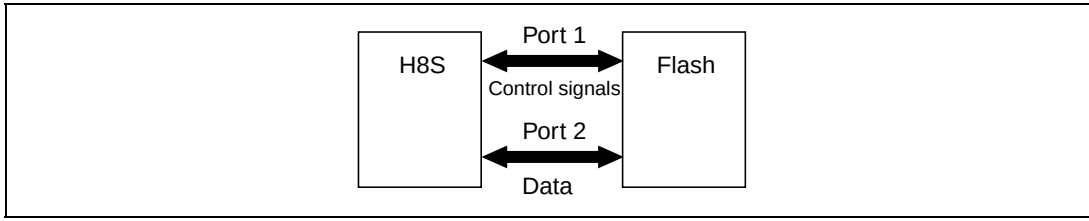


Figure 4.8 Control Method

4.2.3 Sample Circuit

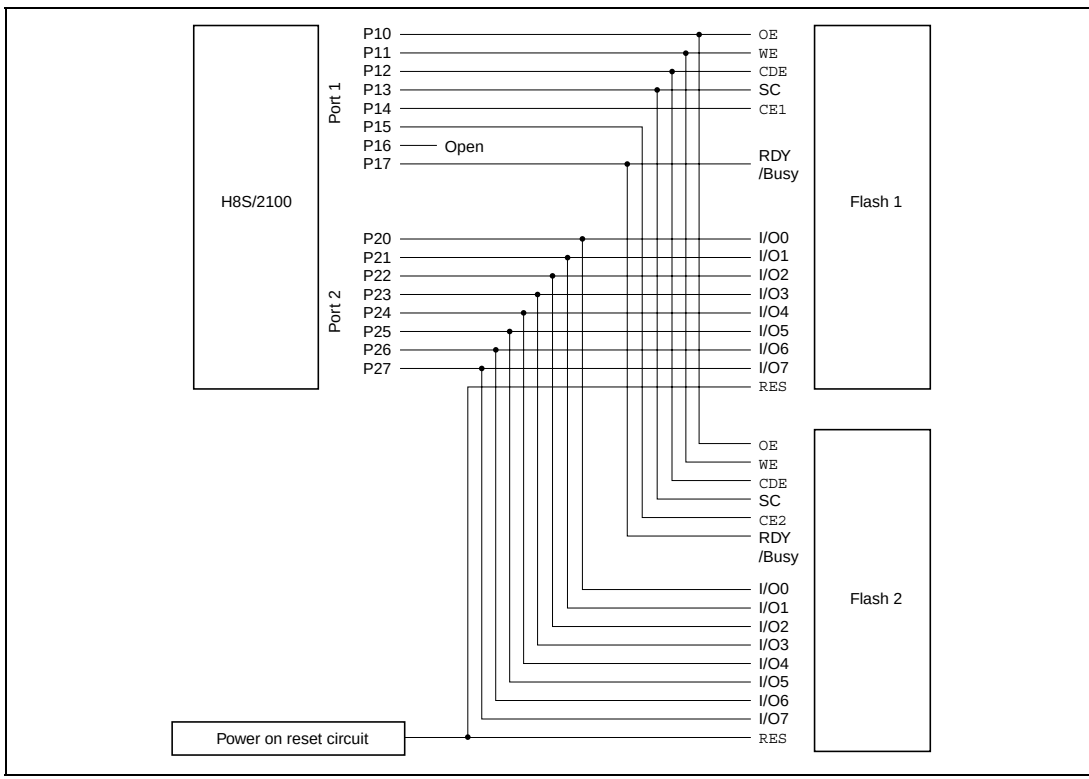


Figure 4.9 Sample Circuit

4.2.4 Port Settings

Port 1 Settings

Port 1	P10	P11	P12	P13	P14	P15	P16	P17
Flash memory	$\overline{OE}$	$\overline{WE}$	$\overline{CDE}$	SC	$\overline{CE1}$	$\overline{CE2}$	—	RDY/Busy
Input/output	Out	Out	Out	Out	Out	Out	Out	In
Pull up	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes

Port 2 Settings

Port 2	P20	P21	P22	P23	P24	P25	P26	P27
Flash memory	I/O0	I/O1	I/O2	I/O3	I/O4	I/O5	I/O6	I/O7
Input/output	I/O	I/O	I/O	I/O	I/O	I/O	I/O	I/O
Pull up	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

4.2.5 Data Transfer to On-Chip RAM

If the on-chip RAM capacity is not sufficient for one flash sector of data to be transferred, one sector of data should be divided for transfer as shown in the example below.

Example: One sector of data is divided into four parts, which are transferred separately.

Taking 2048 + 64 bytes as (512 (data area) + 16 (management area)) × 4 bytes, 528 bytes are transferred at a time.

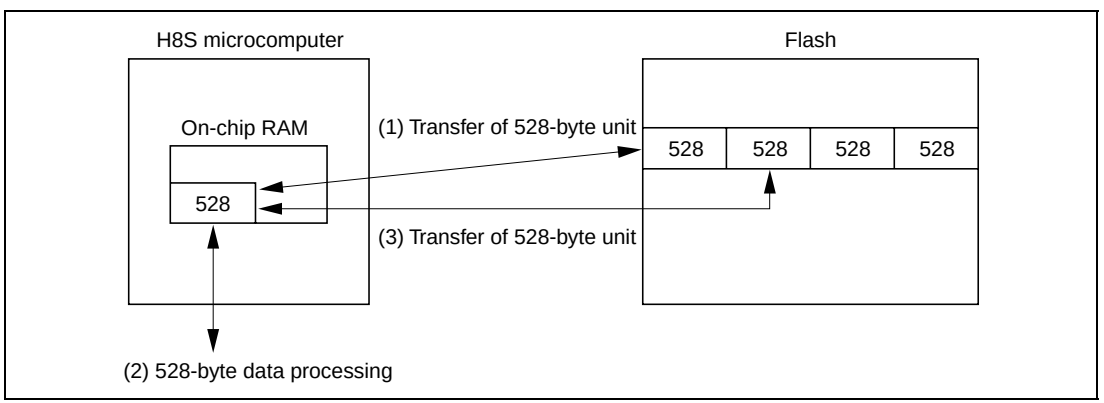


Figure 4.10 Example of Data Transfer

4.3 Reference Materials

(1) 256-Mbit flash memory (HN29W25611 Series) Data Sheet

(2) SH Series Hardware Manual

(3) H8S Series Hardware Manual

5. Power Shutdown Recovery Method

5.1 Internal Circuitry that Executes Erasing/Writing

Figure 5.1 shows the 256-Mbit flash memory circuit configuration and memory cell erase/write operations. Memory cells are linked by a source line, word line, and data line, with each signal line driven by peripheral circuitry.

In a write, a charge is injected into the floating gate by setting the word line to positive potential.

In an erase, the floating gate charge is discharged by setting the word line to negative potential.

When power is shut down, the memory cells themselves in the flash memory are not damaged, but memory cell data may become invalid due to faulty operation of the peripheral circuitry that controls memory cell erasing, writing, etc. (see figure 5.1).

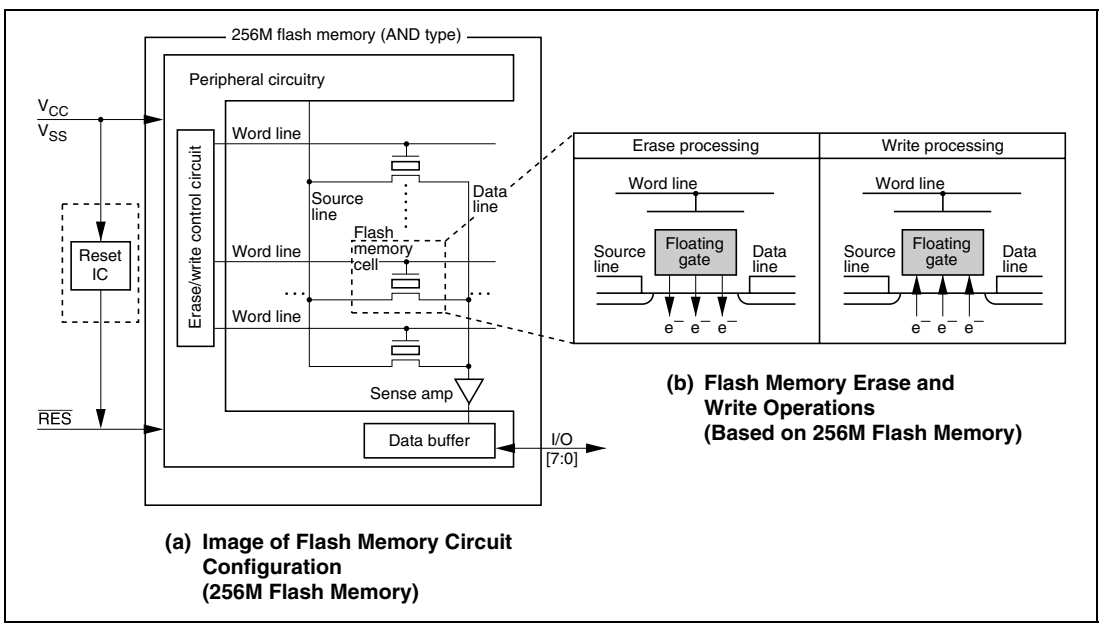


Figure 5.1 Flash Memory Circuit Configuration Image and Erase and Write Operations

5.2 Processing when Power Shutdown Occurs during Erase or Write Operations

If power is shut down during erase or write processing, the following processing should be executed.

(1) Execute a reset by activating the $\overline{\text{RES}}$ signal. (The peripheral circuitry will be initialized.)
(2) Perform erase/write processing again on any sector in which the data has been corrupted due to erasing/writing when the power shutdown occurred.
(3) Read the good sector codes of all sectors, and perform erase/write processing again on any other sectors that have become bad.

Note that the processing in (1) and (3) above can be eliminated by connecting an external Reset IC (see figure 5.1). In this case, only the processing in (2) should be carried out.

5.3 When Power Shutdown Occurs during Reset, Read Transfer, or Other Processing

Execute the processing in (1) and (3) above.

Note that this processing can be eliminated by connecting an external Reset IC.

5.4 If sectors in which a fault has occurred cannot be restored by means of the above processing, processing should be carried out to prevent use of those sectors in the system.

6 Examples of Sector Management Method

6.1 Flash Configuration

The flash configuration in the case of a 256-Mbit product is shown in figure 6.1. The (2048 + 64) bytes constituting a sector are divided into four, giving (512 + 16) bytes as the minimum unit of data. In this section, (512 + 16) bytes is called a sector, and (2048 + 64) bytes is called a block.

256-Mbit flash is composed of 16,384 blocks, with each block consisting of four (512 + 16)-byte sectors.

For data writing, the Program (4) program command is used. An erase command is only used when erasing data as a block.

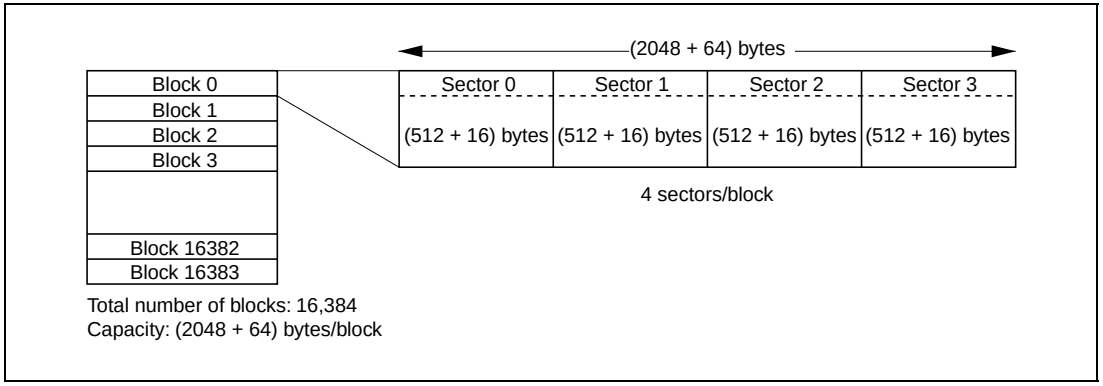


Figure 6.1 Flash Configuration

Each sector is composed of a 512-byte data area, ECC bytes, and sector management bytes.

512 bytes Data	16 bytes ECC + management	512 bytes Data	16 bytes ECC + management	512 bytes Data	16 bytes ECC + management	512 bytes Data	16 bytes ECC + management

6.2 Block Management Method

The block configuration is shown in table 6.1.

Table 6.1 Block Configuration

Item	Number of Blocks Used	Description
Bad block recording table	1 block	Stores good/bad block information
Block management table	16 blocks	Manages correspondence between physical block numbers and logical block numbers. Logical block numbers are consecutive
Spare block area (1)	m blocks	Area reserved for bad blocks. For 256-Mbyte flash, from Data Sheet, m = 16,384 × 0.98 × 0.018 ≈ 290 blocks
Spare block area (2)	n blocks	Area reserved for initial bad blocks. n = 0 to 328
Data block area	16384 –(m +n +16 +1)	Area that can be used as data area. For 256-Mbyte flash, min = 15,750 blocks, max = 16,078 blocks
Block information	1 block	Stores total number of blocks in data block area

(1) Bad block recording table

This table performs management of good/bad flash blocks. The management information is created when flash formatting is carried out. This data is saved as a backup for the initial good/bad block information.

Good/bad information is stored as 1-bit data, and data is held for each physical block.

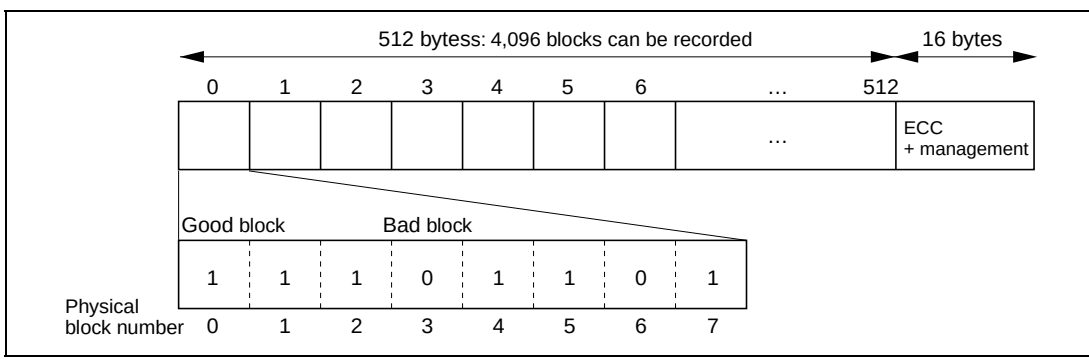


Figure 6.2 Block Recording Table

(2) Block management table

This table manages the correspondence between physical block numbers and logical block numbers for usable blocks. The management information is created when flash formatting is carried out.

The table is updated whenever a bad block occurs during use.

The block information in the block management table is shown in figure 6.3. Two bytes of data are provided for each logical block, holding the physical block number corresponding to the logical block number. The effective capacity information bit indicates whether a logical block corresponds to a data area.

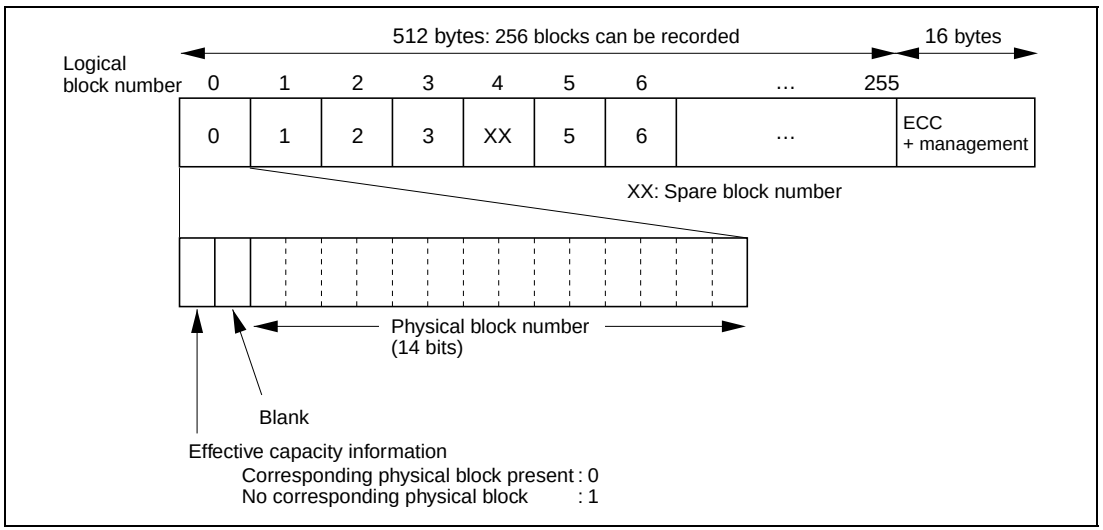


Figure 6.3 Block Management Table

(3) Spare block areas

The spare block areas are spare areas in the event of a bad block.

Figure 6.4 shows the location of each area. The data area is located starting at flash physical block number 0000H.

A bad block in the data area is switched to spare block area (1).

Starting from the highest physical block number, the bad block recording table, block management table, and spare block areas, are located in that order. When there is a bad block, it is placed in the lower good block. For example, if physical block number 3FFFH is a bad block, the bad block recording table is placed at 3FFEH.

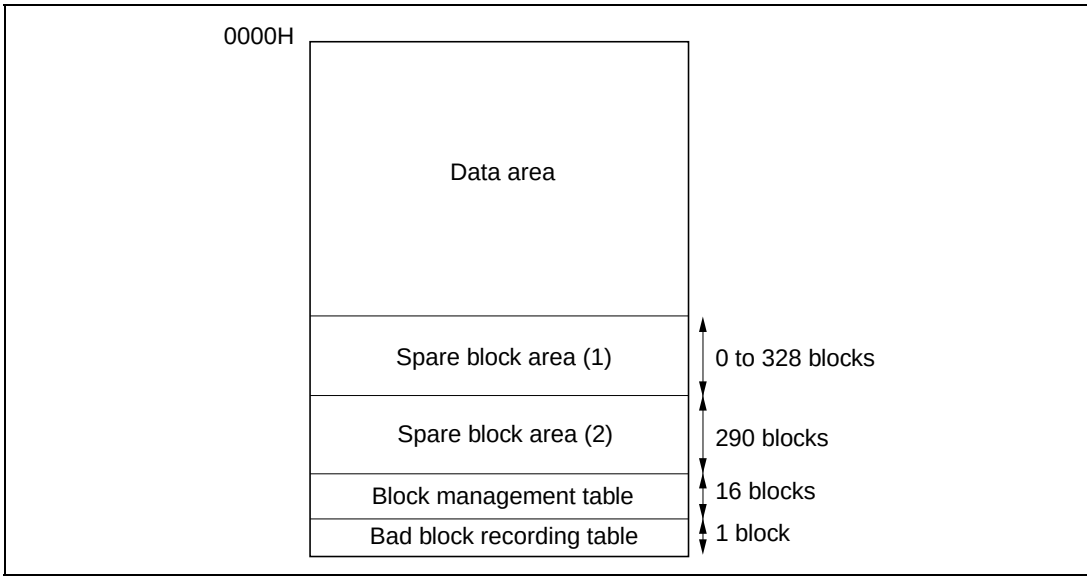


Figure 6.4 Alternate Block Areas

6.3 Sector Management Codes

Sectors are managed by means of 16-byte management data

The management data consists of a sector identification code, block number, good sector identification code, and ECC code.

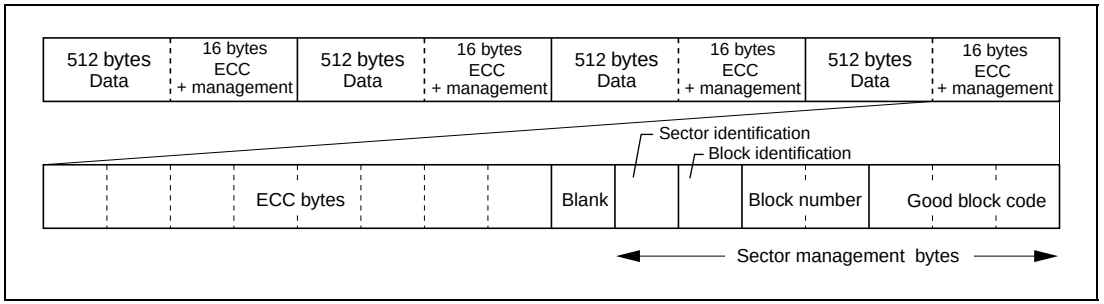


Figure 6.5 Sector Management Codes

(1) Good block code

Stores 3 bytes (1C71C7H) of the ex-factory good block code.

(2) Block number

Stores the logical block number to which the sector belongs.

(3) Block identification code

Indicates which area (data area, spare block area, block management table, or bad block recording table) the block belongs to. The same code is stored for sectors within the same block (see table 6.2).

(4) Sector identification

Indicates the sector status (in use or not used) (see table 6.3).

(5) ECC bytes

These bytes store the EEC code for 512 bytes of data.

Table 6.2 Examples of Block Identification Code

Block Type		Example of Code
Data area		0FH
Spare block area (1)		01H
Spare block area (2)	Substituted	10H
	Not used	1FH
Block management table		00H
Bad block recording table		F0H
Ex-factory bad block		Undefined
Acquired bad block		FFH

Table 6.3 Examples of Sector Identification Code

Sector Status	Example of Data
In use	00H
Not used	FFH

6.4 Sector Access Methods

Data area sectors are accessed by sector number. Sector numbers start from 0.

The quotient resulting from dividing a sector number by 4 is the flash logical block number, and the remainder is the sector position within the block.

Figure 6.6 shows a comparison between physical block numbers and logical block numbers.

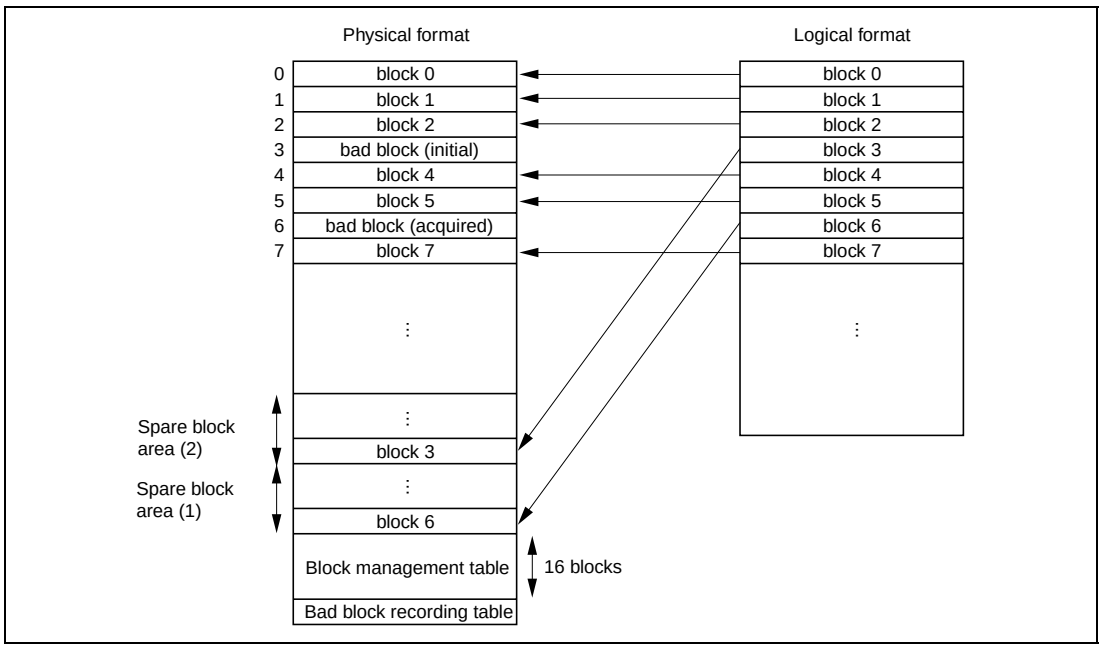


Figure 6.6 Comparison of Physical and Logical Formats

With this sector management method, except for bad sectors, logical block number = physical block number.

Therefore, a method whereby a sector is accessed by performing physical/logical block number conversion, and a method whereby a sector is accessed by performing only bad block physical/logical block number conversion, are possible.

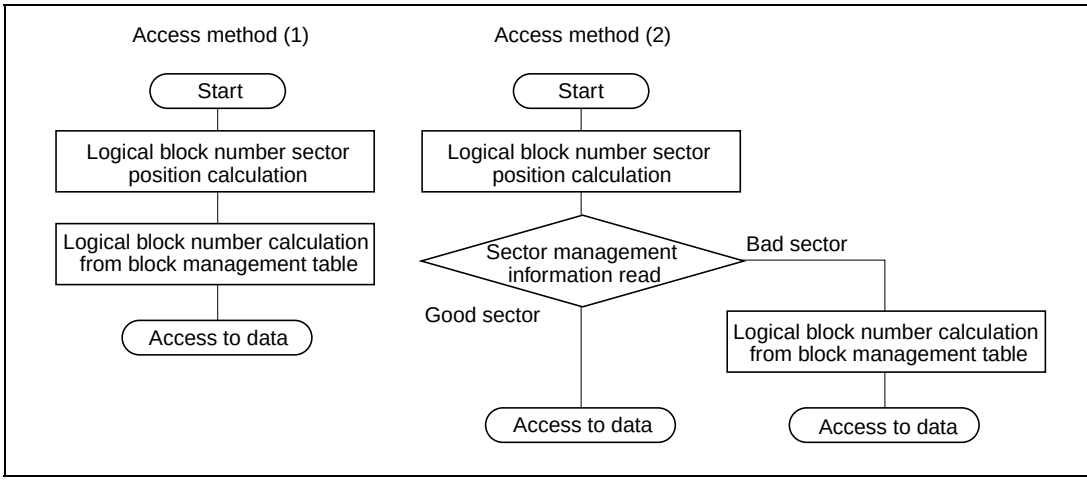


Figure 6.7 Sector Access Methods

If a sector error occurs, the block to which the relevant sector belongs is identified as a bad block, and substitution is performed.

7 Some Common Questions

This section presents a number of technical questions that have been received concerning AND flash memory, in a Q&A format.

7.1 Concerning Flash Memory

[Question]

What kind of memory is flash memory?

[Answer]

Flash memory is a kind of nonvolatile memory (NVM), which retains its data even if power is cut, and features both a high level of integration and electrical programming functions.

Flash memory is broadly divided into two kinds: random access type and serial access type. AND flash is serial access flash memory.

[Question]

What is the difference between AND flash and NAND flash?

[Answer]

Both are serial access type flash memory for data storage use, featuring large capacity and high-speed reading, wiring, and erasing.

A major difference between the two relates to the write/erase units.

With AND flash, the write unit and erase unit are the same. For 256-Mbyte flash, this is the (2k + 64)-byte sector unit.

With NAND flash, the write unit and erase unit are different, with a (512 + 16)-byte page used for writing, and an 8- to 16-kbyte block unit for erasing.

[Question]

What is MGM?

[Answer]

MGM is an abbreviation of Mostly Good Memory. 98% MGM refers to memory in which fewer than 2% of all sectors are invalid (bad) when the product is shipped. This increases chip yield and helps to lower costs.

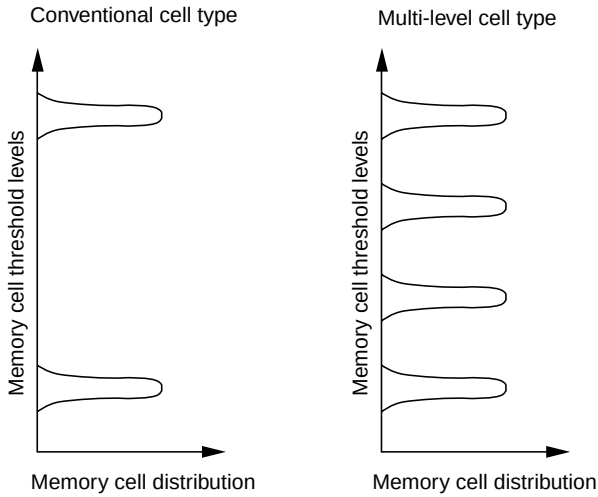
[Question]

What is multi-level technology?

[Answer]

In contrast to the two threshold levels, 0 and 1, of conventional memory, this technology controls four threshold levels, 00, 01, 10, and 11, enabling two memory cells' worth of data to be stored in a single memory cell.

As multi-level technology enables twice the capacity of conventional technology to be achieved with the same number of memory cells, it is effective in cutting costs by increasing capacity and reducing chip size.



7.2 Concerning the Interface

[Question]

Is it possible to design a system that can support future large-capacity products (512 Mbytes and up)?

[Answer]

Yes. The functions, command system, and electrical characteristics are compatible with current 512-Mbyte flash, but the increase in addresses has to be taken into consideration. The package, also, is identical to the 48-pin TSOP (I), and the pins necessary for control are also the same.

[Question]

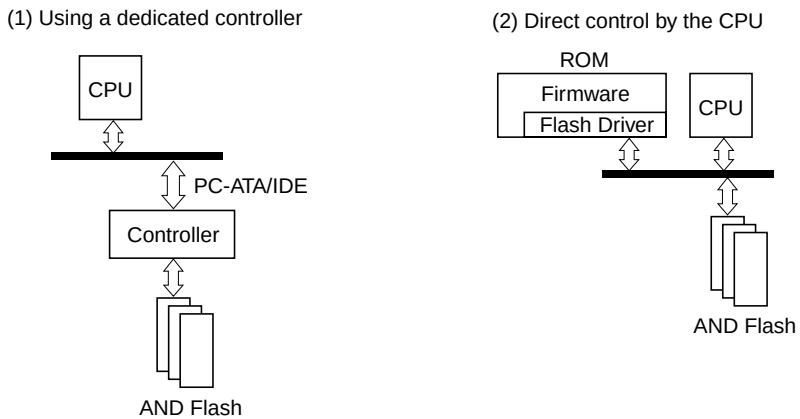
How should AND flash be used?

[Answer]

The optimum method of use is determined by the requirements for the target system flash. Generally speaking, there are two methods of use, involving a trade-off between mounting area and performance:

1) Using a dedicated controller and a software driver for its interface (PC-ATA or IDE)

2) Incorporating the flash driver in the host CPU firmware, and performing direct control with a CPU of adequate performance (SH-3 80 MHz level)



7.3 Concerning Power Shutdown

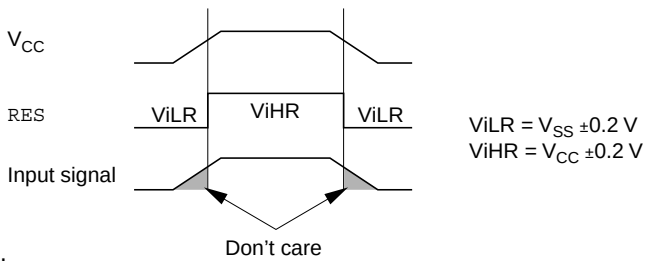
[Question]

Are there any points requiring attention when powering on and off?

[Answer]

The $\overline{RES}$ pin should be driven low before turning on the power.

When powering off, check that the chip is in the Ready state, then drive the $\overline{RES}$ pin low and cut the power. Using this procedure will ensure that data is protected even if the input signal becomes unstable when powering on or off



After power-on, the chip performs internal initialization processing, and automatically goes to the standby state. Operation is possible as soon as the status changes to Ready.

[Question]

Won't data be lost if a power cut (momentary power interruption) occurs?

[Answer]

Data will not be lost if the flash status is Ready.

In addition, data will not be lost when the status is Busy during reading.

If the status is Busy during writing/erasing, operation is forcibly terminated before the memory cells go to the normal write/erase state, and so the data in the relevant sector will be undefined. For the sector recovery method in this case, see section 5, Power Shutdown Recovery Method.

7.4 Concerning Reset Operations

[Question]

Can operation be forcibly reset by issuing a reset command (FFH) in the Busy state?

[Answer]

No. No commands are accepted in the Busy state.

In the case of writing/erasing, it is possible to transit to the standby state by using a reset command (FFH) during the setup state before a start command (writing: 40H, erasing: B0H) is issued.

[Question]

What happens if a hardware reset ($\overline{\text{RES}}$: low) is performed in the Busy state?

[Answer]

All operations are forcibly terminated and a transition is made to the deep standby state.

In the case of writing/erasing, the data in the relevant sector will be undefined.

The recovery method for a sector with undefined data due to a hardware reset is the same as the sector recovery method after power is cut. See section 5, Power Shutdown Recovery Method.

7.5 Concerning Write Operations

[Question]

How many times can Program (1) and (3) additional writes be performed?

[Answer]

There is no limit to the number of writes that can be performed with the additional write function. However, an additional write should be counted as one rewrite (programming) operation.

For example, when performing additional writes with one sector (2 kbytes) divided into four 512-byte units, the number of rewrite operations for that sector should be counted as 4.

[Question]

What is the difference between Program (1) and (3) additional writes and Program (4) rewriting?

[Answer]

An additional write can only be performed on FFH data (i.e. data in the erased state) in a sector. Rewriting (programming) is a data contents update and overwrite function.

For details, see section 1, Hitachi AND Flash Write Commands.

[Question]

If writing is performed with Program (4) by specifying a column address, what happens to data outside the specified range?

[Answer]

For data outside the specified range—that is, areas with no data input from outside—the data prior to the write is retained.

7.6 Concerning Bad Sectors

[Question]

What is a bad sector (invalid sector) in AND flash?

[Answer]

AND flash memory is based on the MGM (Mostly Good Memory) concept, and allows the existence of sectors that do not operate normally—that is, defective sectors—within the chip. These defective sectors are called bad sectors.

Both bad sectors present when the product is shipped from the factory, and bad sectors that arise later within the user system (acquired bad sectors), are permitted.

[Question]

How many bad sectors can be expected?

[Answer]

When flash is shipped from the factory, fewer than 2% of all sectors are bad. With 256-Mbyte flash this means $16{,}384 \times 2\% \approx 328$ sectors.

[Question]

Do bad sectors occur during use in the system?

[Answer]

Yes. Of the 98% or more ex-factory good sectors, it is possible that 1.8% (with 256-Mbyte flash: 16,384 (total number of sectors) $\times$ 98% $\times$ 1.8% $\approx$ 290 sectors) may become bad.

[Question]

How are ex-factory bad sectors identified?

[Answer]

The data shown in the table below is written in good sectors. If column addresses 820H to 825H are as shown in the table, the sector is good. Data in a bad sector is undefined.

Column address	**000H to 81FH**	**820H**	**821H**	**822H**	**823H**	**824H**	**825H**	**826H to 83FH**
Data	00H	1CH	71H	17H	1CH	71H	C7H	FFH

[Question]

How can the occurrence of bad sectors be identified during use in the system?

[Answer]

An error during writing/erasing is identified by reading the status register.

After writing/erasing has been performed and the Ready state is entered, the relevant bit of the status register (bit 4 for writing, bit 5 for erasing) is checked, and processing ends normally if the bit is 0, or ends with an error if the bit is 1. In a normal end, the memory data values are as expected.

Read errors should be detected using a method such as ECCs.

Mode	Detection Method
Erase error	Status register read
Write error	Status register read
Read error	ECC check, etc.

[Question]

When a sector that did not have a good sector code at the time of shipment was rewritten, the processing ended normally. Can this sector be used as a good sector?

[Answer]

A sector that does not have a good sector code at the time of shipment is a bad sector. A bad sector is one with a defect of some kind, or one that has a high probability of developing a defect. Even if rewriting is performed normally, there is still a possibility of some kind of problem arising, such as with the reliability of the data, for instance. Therefore, bad sectors should not be accessed.

[Question]

When a bad sector is read, data of some kind is obtained. Will the same data be obtained however many times that sector is read?

[Answer]

As data in a bad sector lacks reliability, there is no guarantee that the same data will be obtained again.

7.7 Concerning Sector Information (Good/Bad Sectors)

[Question]

Is it possible to recover ex-factory sector information if it has been erased?

[Answer]

It is extremely difficult to do so. Ex-factory bad sectors are identified as the result of various tests conducted during the manufacturing process. In addition to simple write/erase/read tests, a test mode is also used that takes past experience into consideration.

There are many modes for the occurrence of a bad sector, and only some of these are tested in ordinary write/erase tests.

It is important to save sector information in some way, and ensure that it is not lost.

[Question]

Is it OK for sector codes written at the time of shipment to be overwritten with different codes?

[Answer]

Yes, this is possible. As long as good/bad sector information can be saved, the method is immaterial. The most appropriate method for the system should be used.

[Question]

Will any problems occur if ex-factory sector information is not used?

[Answer]

Bad sectors will be used. If a bad sector is used, the data in that sector will be unreliable. Even if writing/erasing ends normally, there is a possibility of problems occurring such as data become corrupted when left for a short while.

[Question]

Can bad sectors be managed by writing a bad sector code?

[Answer]

No. No data whatever can be written to a bad sector. Even if a write is performed successfully, the data will be unreliable.

7.8 Concerning Sector Management

[Question]

What kind of management is necessary?

[Answer]

The following three kinds of management should be carried out.

(1) Preventing ex-factory bad sectors from being accessed
(2) Identifying bad sectors that occur during use in the system, and preventing their subsequent access
(3) Performing sector replacement processing

[Question]

If a power interruption occurs after erasing is completed when rewriting data in the system, the good sector code is lost, and when power is restored the relevant sector becomes a bad sector. Is there any way of preventing this conversion to a bad sector?

[Answer]

One method is to hold a sector management table in the flash together with the good sector codes.

7.9 Concerning Error Handling and Sector Substitution Processing

[Question]

When a write error occurs, is it OK to perform a retry on the same sector?

[Answer]

There is a possibility of an error occurring again. The retry should be performed on a different sector.

[Question]

How is the spare block decided when a sector error occurs and sector replacement is performed?

[Answer]

There is no fixed procedure.

A method appropriate to the system should be used, such as reserving a certain area as a spare block area, or finding an empty sector when an error occurs and using that sector as the spare block.

7.10 Concerning ECCs

[Question]

The sector data area is divided into four 512-byte units for use. Is a 3-bit ECC necessary for 512 bytes in this case?

[Answer]

Yes.

[Question]

Is it OK to write a code generated by the ECC function to a management area?

[Answer]

Yes, this can be done.

[Question]

Why are ECCs necessary?

[Answer]

They are used for detecting/correcting random bit errors due to data retention, and preventing read errors.

A write error can easily be identified by checking the status, but a read error may occur due to data retention after the write has ended normally. As flash outputs data without detecting changes that may have occurred due to data retention, verification of read data is necessary on the system side.

7.11 Concerning Disturbance

[Question]

Are there any write patterns that are susceptible to disturbance?

[Answer]

No. There is no disturbance problem with good sectors.

8. Check List

When designing a system incorporating Hitachi AND flash memory, the specifications in the following check list should be confirmed before starting the design work.

Item				Relevant Page of Data Sheet (ADE-203-1178A(Z))	Check Column	Remarks
Must-do items	Hardware design	Basic operational condition	Pin arrangement		P3, 6		
			Absolute ratings		P17		
			Operating environment				
			V_{CC}	3.3 ± 0.3 V	P1, 18		
			T_{opr}	0 to 70˚C	P17		
			E/W	300K times	P36 (290 alternate sectors when using 3-bit ECC)		
			I_{CC2} (Typ.)	30 mA	P18 (f = 20 MHz)		
			V_{IH} (Min.)	2.0 V	P18		
				V_{CC} – 0.2 V	P18 ($\overline{RES}$)		
			V_{IL} (Max.)	0.8 V	P18		
				0.2 V	P18 ($\overline{RES}$)		
			t_r/t_f	5 ns ↓	P19		
			Other				
		Timing	Mode setting		P6, 10, 11, 12		
			Power-on sequence		P23		
			Operation command setting	Manufacturer's code read	P32		
				Product code read	P32		
				Status register read	P32		
				Clear status register	P35		
				Serial read (1)	P24, 25		
				Serial read (2)	P24, 25		
				Programming (1)	P26, 27		
				Programming (2)	P28		
				Programming (3)	P29		
				Programming (4)	P30, 31		
				Recovery read	P33		
				Recovery write	P34		
			Power-off sequence		P23		
			All timing items		P19 to 22		
	Basic software design		Understood that this is MGM. Possibility of up to 2% random bad sectors out of 16,384 sectors.				
			Only good sectors must be accessed.				
			System should not stop even if a good sector becomes a bad sector when accessed.				
			Good sector code must not be deleted except in case of error occurrence.				
			If power supply (3.0 V min.) cannot be sustained, Hitachi's recommended power-off sequence must be executed.				
	System reliability software design		Error provisions				
			(1) Provide alternate sectors (at least 290).				
			(2) Perform at least 3-bit ECC.				
			Power shutdown provisions In case of emergency, perform chip-internal processing with /RES signal before powering off.				
			If there is chain data, system must ensure that there is no problem if chain is broken midway.				

Flash Application Design Guidelines

Publication Date: 1st Edition, March 2000
2nd Edition, December 2000
Published by: Electronic Devices Sales & Marketing Group
Semiconductor & Integrated Circuits
Hitachi, Ltd.
Edited by: Technical Documentation Group
Hitachi Kodaira Semiconductor Co., Ltd.