To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: http://www.renesas.com

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

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

Application Note

F-ZTAT Reprogramming by On-Chip CAN

Application Note

Renesas Electronics
www.renesas.com

Rev.1.0 2003.03

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.

General Precautions on Handling of Product

1. Treatment of NC Pins

Note: Do not connect anything to the NC pins.
The NC (not connected) pins are either not connected to any of the internal circuitry or are used as test pins or to reduce noise. If something is connected to the NC pins, the operation of the LSI is not guaranteed.

2. Treatment of Unused Input Pins

Note: Fix all unused input pins to high or low level.
Generally, the input pins of CMOS products are high-impedance input pins. If unused pins are in their open states, intermediate levels are induced by noise in the vicinity, a pass-through current flows internally, and a malfunction may occur.

3. Processing before Initialization

Note: When power is first supplied, the product's state is undefined.
The states of internal circuits are undefined until full power is supplied throughout the chip and a low level is input on the reset pin. During the period where the states are undefined, the register settings and the output state of each pin are also undefined. Design your system so that it does not malfunction because of processing while it is in this undefined state. For those products which have a reset function, reset the LSI immediately after the power supply has been turned on.

4. Prohibition of Access to Undefined or Reserved Addresses

Note: Access to undefined or reserved addresses is prohibited.
The undefined or reserved addresses may be used to expand functions, or test registers may have been be allocated to these addresses. Do not access these registers; the system's operation is not guaranteed if they are accessed.

Preface

This application note describes how to reprogram the flash memory using user program mode. The program data can be provided by using the on-chip HCAN (Hitachi Controller Area Network) of the H8S Series.

For details on the flash memory and HCAN, refer to the following sections in the H8S/2612F Hardware Manual.

- ROM
- HCAN

Although operations of programs and circuit examples, etc. described in this application note are confirmed, be sure to confirm them again before actual use. (Note that examples of programs in this application note are for the on-chip HCAN of the H8S/2612F.)

Contents

Section 1 Overview

This application note describes how to reprogram the H8S/2612F on-chip flash memory using user program mode. The program data can be provided by using the on-chip HCAN of the H8S/2612F.

As a sample system, this application note describes how to reprogram the flash memory in the system configuration shown in figure 1.1. Also this application note describes how to customize the programs shown in the application note according to the operating frequency and the CAN bus specifications for the user system.

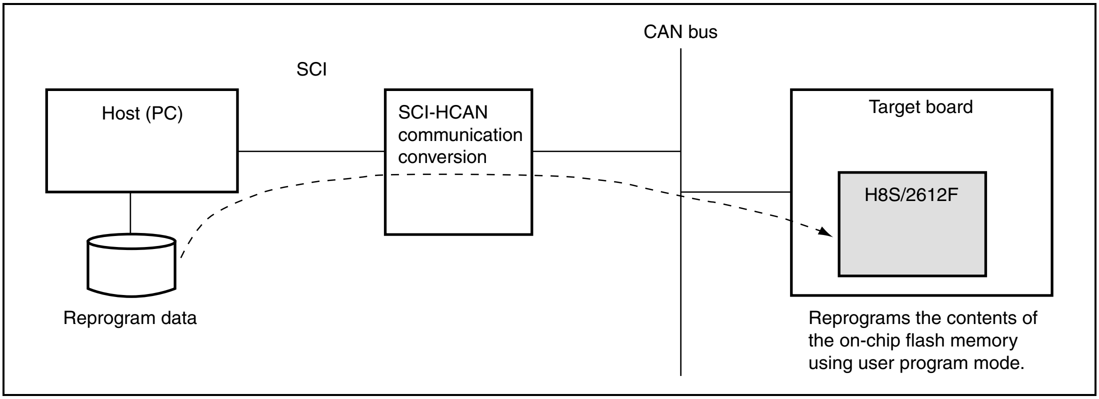


Figure 1.1 Sample System Configuration

1.1 List of F-ZTAT™* Microcomputers (H8S Series) Including HCAN Unit

This application note applies to the following devices:

- H8S/2612F
- H8S/2623F
- H8S/2626F
- H8S/2636F

When using this application note as a reference for using microcomputers other than the H8S/2612F, note the differences in:

- Addresses, bit positions, and functionality of the on-chip registers
- Control method for erasing and programming the flash memory (such as application time for E and P bits)

Note: * F-ZTAT is a trademark of Hitachi, Ltd.

1.2 Overview of User Program Mode

User program mode enables on-board reprogramming of the flash memory. This mode allows the user to reprogram the contents of the on-chip flash memory with the F-ZTAT microcomputer mounted on the user's board.

Two modes are available for on-board reprogramming of the flash memory: boot mode and user program mode. In boot mode, a boot program included in the F-ZTAT microcomputer is executed to implement programming to the flash memory. In user program mode, an application program in the flash memory (on-chip ROM) is executed. For this reason, user program mode requires the reprogram processing to be installed in the application program in advance.

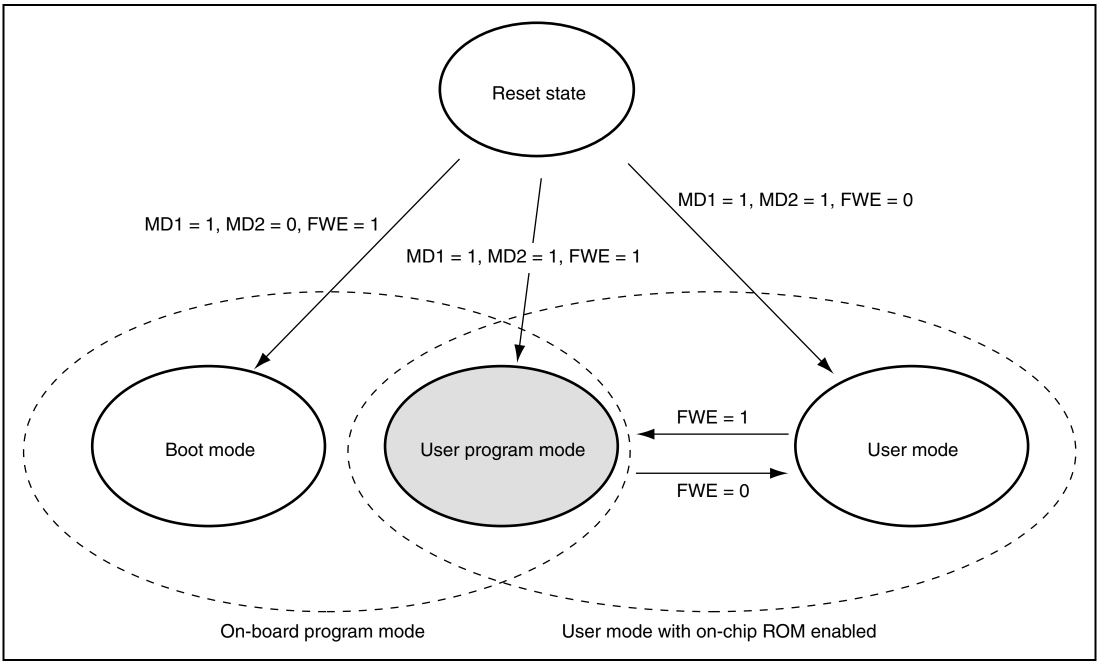


Figure 1.2 Transition of User Program Mode

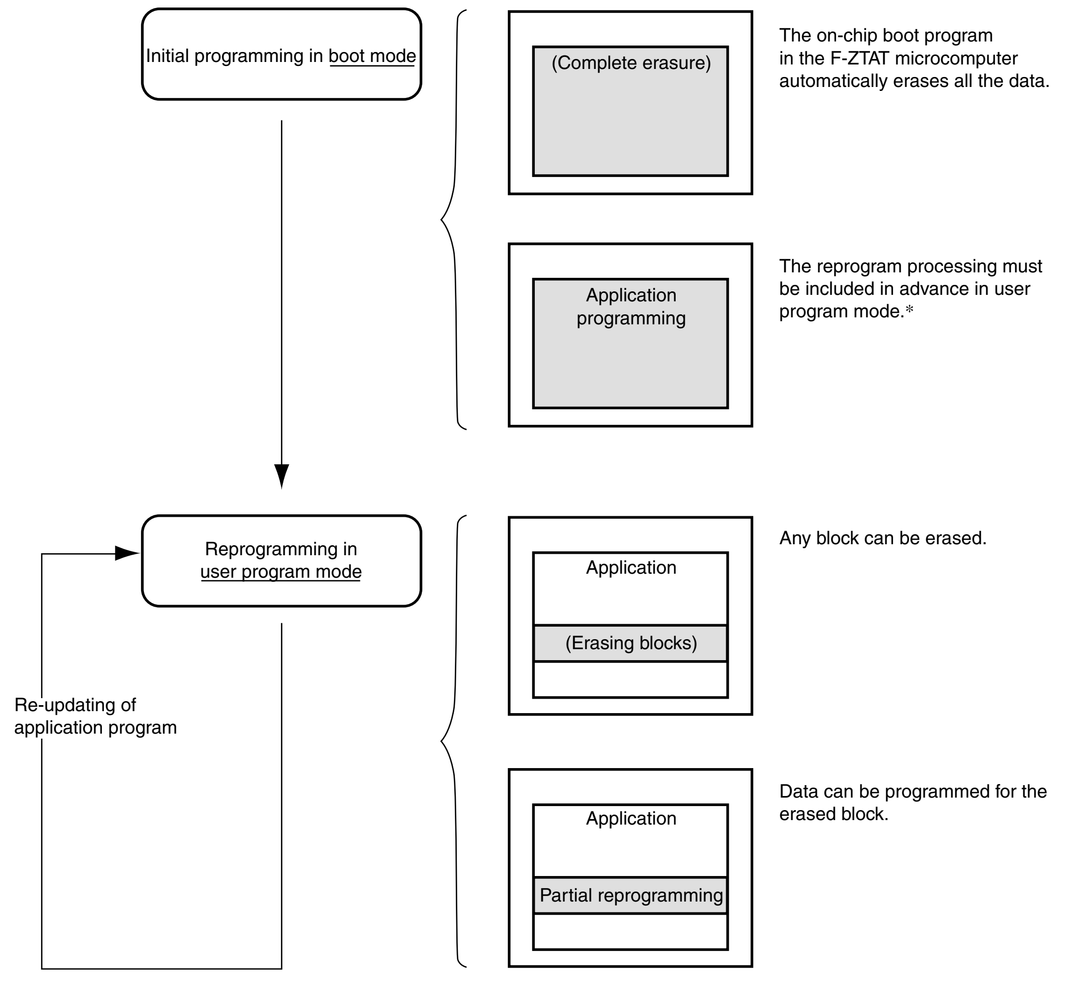
In boot mode, the on-chip boot program automatically erases the entire flash memory before new data is programmed. For this reason, a complete reprogramming of the application program is required even if the user only wants to reprogram it partially.

User program mode allows the user to erase and program any data according to the user system, allowing a partial reprogram for individual erased blocks.

Note: The word "program" of user program mode means "programming to" the flash memory.

1.3 Overview of Reprogramming Method in User Program Mode

Since the on-chip ROM is enabled in user program mode, the application program including the reprogram processing must be programmed to the on-chip flash memory in advance. Use boot mode or ROM programmer mode to perform this initial programming. If the reprogramming in user program mode is disabled due to accidental erasure of the application program including the reprogram processing, the user can forcibly program the data in boot mode.



Note: * Contents of the reprogram processing that must be installed into the application
The processing that detects the transition conditions for reprogram processing, and the processing that transfers the program/erase control program to RAM and then jumps to RAM must be installed into the application in advance.

Figure 1.3 Reprogramming Method in User Program Mode (Overview)

2.1 Hardware List

The following lists the components which are required for executing the sample program described in this application note.

Table 2.1 Hardware List

No.	Hardware Name	Specifications	Remarks
1	Host (PC)	Executes FlashCAN.exe. Must have the serial interface.	OS for DOS/V personal computers: Windows®*[1] 98(S), Windows® 2000, Windows® NT4.0, Windows® Me, or Windows® XP
2	SCI-CAN communication conversion board	Converts serial communications from the host to CAN communications to communicate with the target board. It also converts CAN communications from the target board to serial communications.	LIN-CAN Starter Kit (H8S/2612F) manufactured by Hokuto Denshi co., ltd.
3	Target board	Contains on-board H8S/2612F that has the on-chip flash memory to which the data will be programmed.	LIN-CAN Starter Kit (H8S/2612F) manufactured by Hokuto Denshi co., ltd. requires a switch for the FWE pin to support user program mode.*[2]
4	Serial cable	Connects the 9-pin host to the J3 connector on the SCI-CAN communication conversion board	Attached to the LIN-CAN Starter Kit (H8S/2612F) manufactured by Hokuto Denshi co., ltd.*[2]
5	CAN bus cable	Connects the J7 connector on the 3-pin SCI-CAN communication conversion board to the J7 connector on the target board	Attached to the LIN-CAN Starter Kit (H8S/2612F) manufactured by Hokuto Denshi co., ltd.

Notes: 1. Windows® is a registered trademark of Microsoft Co., in the U.S. and other countries.

2. A transition to user program mode requires turning on and off the FWE pin.
The LIN-CAN Starter Kit (H8S/2612F) manufactured by Hokuto Denshi co., ltd. uses the slide switch (SW11) on the board to turn on and off the FWE pin.
For actual operations on changing modes for the LIN-CAN Starter Kit (H8S/2612F), see the attached manual.

2.2 Software List

Table 2.2 Software List

No.	File Name	Program Name	Remarks
1	FlashCAN.exe	F-ZTAT Microcomputer H-CAN Program	This program runs in the host (PC). It is an evaluation version for this application note.
2	SCI2612F3.sub	Program control program (SCI communications)	Use this program for the initial programming in boot mode. When executing this program, it should be sent from FlashCAN.exe to the boot program in the target board via serial transmission, and then store it in RAM.
3	HCAN2612F3.sub*[1]	Program/erase control program (HCAN communications)	Use this program for the reprogramming in user program mode. When executing this program, it should be sent from FlashCAN.exe to the processing included in the application on the target board via serial transmission, and then store it in RAM.
4	SCItoCAN.mot	SCI-CAN communication conversion program	This program runs in the SCI-CAN communication conversion board.
5	Sample1.mot*[1]	Application (sample) program	This program runs in the target board. This program includes the reprogram processing in user program mode.

Note: * To use the sample program provided in this application note according to the user system, you can customize the items listed below. Change the source files, and then compile and assemble them.
You do not need to customize anything if you use the sample system without any changes.

2.3 Items to be Customized for Sample Program

No.	Items	Defaults	Sections to Change
1	CAN communication settings	See the next page.	• Sample1.mot InitHCAN() function in the HCAN_up.c file
2	Mailbox numbers for transmission and reception used by the on-chip HCAN	Receive: MB4 Transmit: MB5	• The following functions in the Sample1.motHCAN_up.src file: InitHCAN(), PowerON_Reset(), CAN_MB4_rcv1byte(), and CAN_MB5_trs1byte() • HCAN2612f3.mot RCV1BYTE() and TRS1BYTE() subroutines in the HCAN2612f3.src file

To compile or assemble the program, use the batch file or Hew project of the DOS prompt. The attached batch file and Hew project found in the samples show an example of using the compiler package Ver. 5.0.02.

Change the following files according to the version of your compiler package.

- To change the HCAN2612f3.mot file, use the file HCAN2612f3.bat or Hew2.hws in the HCAN2612f3 folder.
- To change the Sample1.mot file, use the file Sample1.bat or Hew2.hws in the Sample1 folder.

2.4 Installing Programs to be Used in Application Note

Execute setup.exe to install the programs. The following shows the directory configuration after installation.

```
[Directory Configuration after Standard Installation]

[c:\FlashCAN] ──┬── FlashCAN.exe
                ├── SCI2612F3.sub
                ├── SCI2612F3.inf
                ├── HCAN2612F3.sub
                ├── HCAN2612F3.inf
                ├── [HCAN2612F3]               ; Program/erase control program
                │        ├── HCAN2612F3.src . . . (*1) Program that can be customized
                │        ├── HCAN2612F3.bat
                │        ├── [Hew2] ──────┬── [Debug] ─────────────── Debug.hdp
                │        ├── Hew2.hws     ├── [Release] ──────────┬── Release.hdp
                │        ├── Hew2.tws     ├── [SimDebug_H8S-2600A] ├── HCAN2612f3.abs
                │        └── Hew2.hbp     ├── Hew2.hwp             ├── HCAN2612f3.h8a
                │                         ├── Hew2.pgs             ├── HCAN2612f3.hlk
                │                         ├── Hew2.tps             ├── HCAN2612f3.obj
                │                         ├── DefaultSession.hsf   └── Hew2.map
                │                         └── SimSessionH8S-2600A.hsf
                ├── [Sample1]                  ; Application (sample) program
                │        ├── Sample1.mot . . . (*1) Program that can be customized
                │        ├── HCAN_up.c
                │        ├── Sample1.c
                │        ├── io_2612.h
                │        ├── Sample1.bat
                │        ├── [Hew2] ──────┬── [Debug] ─────────────── Debug.hdp
                │        ├── Hew2.hws     ├── [Release] ──────────┬── Release.hdp
                │        ├── Hew2.tws     ├── [SimDebug_H8S-2600A] ├── Sample1.abs
                │        ├── Hew2.hbp     ├── Hew2.hwp             ├── HCAN_up.h8c
                │        ├── Data_00.mot  ├── Hew2.pgs             ├── Sample1.h8c
                │        └── Data_FF.mot  ├── Hew2.tps             ├── Sample1.hlk
                │                         ├── DefaultSession.hsf   ├── HCAN_up.obj
                │                         └── SimSessionH8S-2600A.hsf ├── Sample1.obj
                │                                                  └── Hew2.map
                └── [SCItoCAN]                 ; SCI-CAN communication conversion program
                         └── SCItoCAN.mot
```

You can start up the program tool (FlashCAN.exe) from the Windows® Start menu as follows:

From the Start menu, select Programs, FlashCAN, and then FlashCAN.

2.5 CAN Bus Interface (Example)

A bus transceiver IC is necessary to connect this chip to a CAN bus. A Philips PCA82C250 transceiver IC is recommended. Any other product must be compatible with the PCA82C250. Figure 2.1 shows a sample connection diagram.

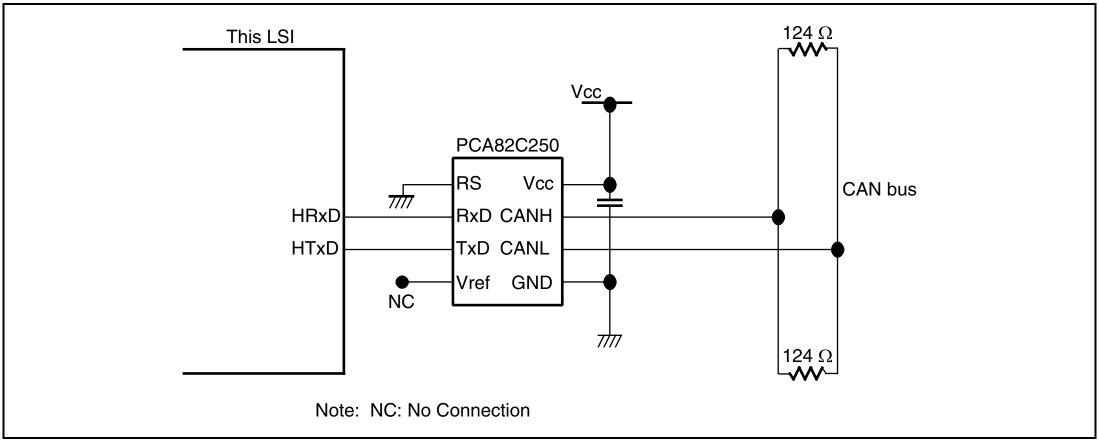


Figure 2.1 High-Speed Interface Using PCA82C250

Table 2.3 LED Pin Assignment

Microcomputer Pin	LED
PD0	D1
PD1	D2
PD2	D3
PD3	D4
PD4	D5
PD5	D6
PD6	D7
PD7	D8

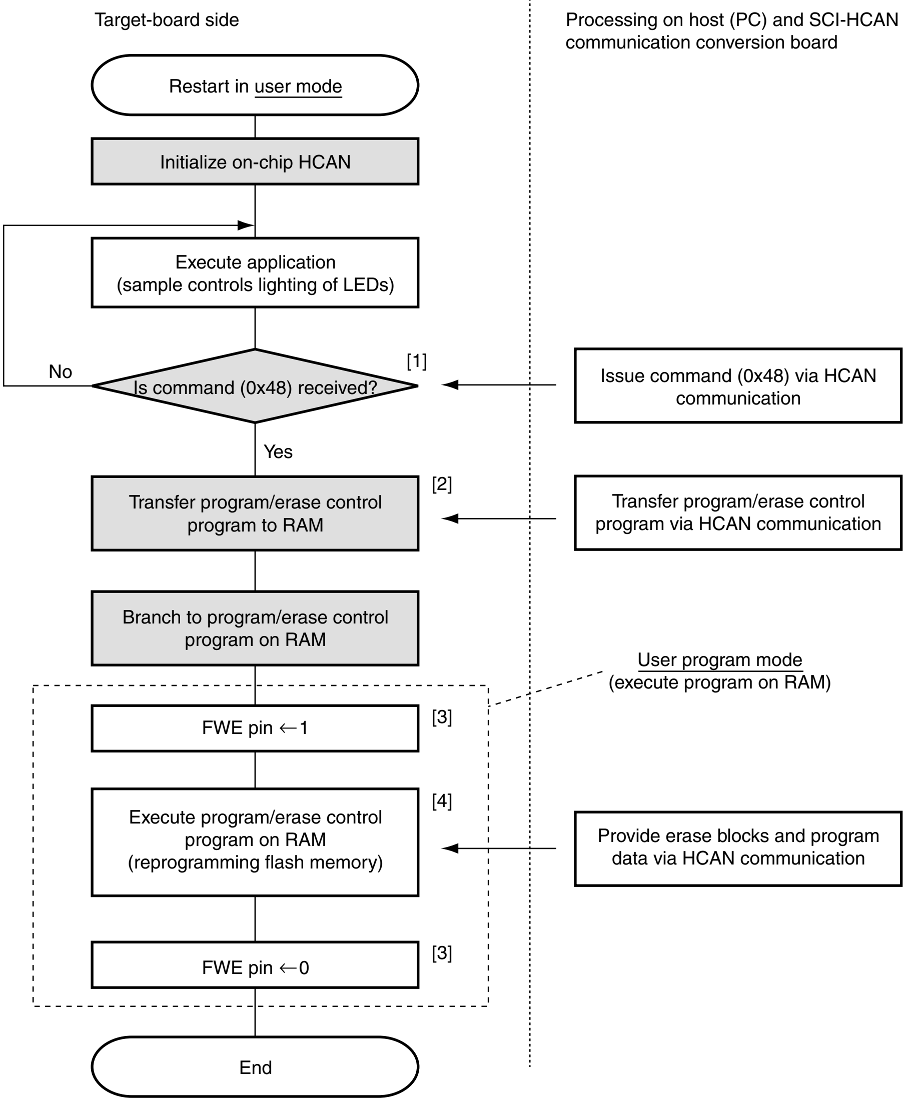
The LED port is turned on by the bit of 0.

Section 3 Procedure for Reprogramming Flash Memory in User Program Mode

This application note uses the following procedure for reprogramming the flash memory:

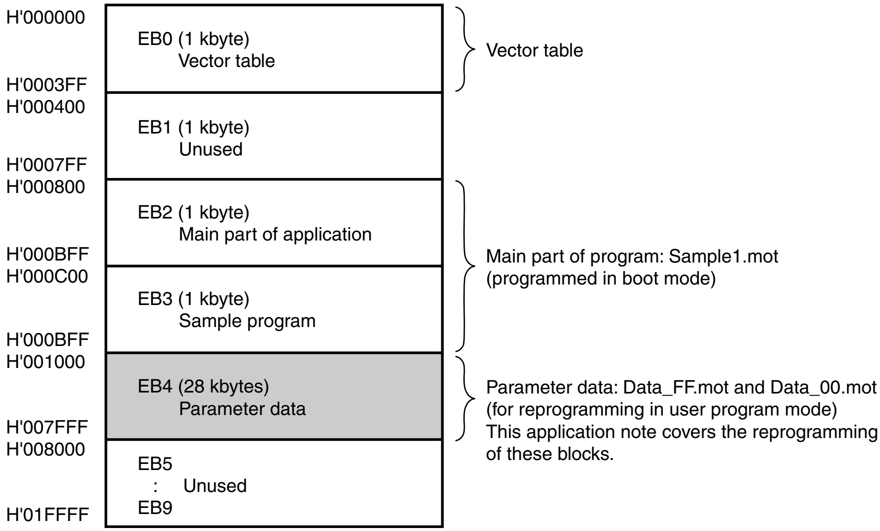
1. Use the reception of a command via HCAN communication as a trigger to start the reprogramming of the flash memory.
2. Transfer the program/erase control program from outside to RAM via HCAN communication.
3. Switch the FWE pin on the target board (set FWE = 1 to enter user program mode).
4. Transfer the program data from outside via HCAN communication.

The hosts used in steps 1, 2, and 4 above are the host (PC) and SCI-HCAN communication conversion board.



Note: The shaded section in the flowchart indicates the processing that must be installed into the application in advance.

3.1 Mapping Flash Memory for Application (Sample) Program



Note: Download Sample1.mot into the flash memory on the target board in boot mode beforehand. This application note describes the reprogramming of Data_FF.mot or Data_00.mot into the flash memory in user program mode. Since the application turns on or blinks the LEDs according to this data, you can visually check that the data has been reprogrammed.

3.2 Programming Programs to Target Board and SCI-HCAN Communication Conversion Board

Before reprogramming the flash memory in user program mode, you need to program the programs in boot mode. Use 'Standard Mode' of FlashCAN.exe for programming in boot mode.

Program the following programs in boot mode:

- Program Sample1.mot into the target board.
- Program SCItoCAN.mot into the SCI-HCAN communication conversion board.

For details on how to use FlashCAN.exe, see section 8.

3.3 CAN Communication Settings in Application Note

The application note requires the following settings for CAN communication:

CAN baud rate: 1000 kbps

(CPU operating frequency = 20 MHz and BCR setting = H′0034.)

- BRP setting: 0 (1 time quantum/ 2 system clocks)
- TSEG1 setting: 4 (5 time quanta)
- TSEG2 setting: 3 (4 time quanta)

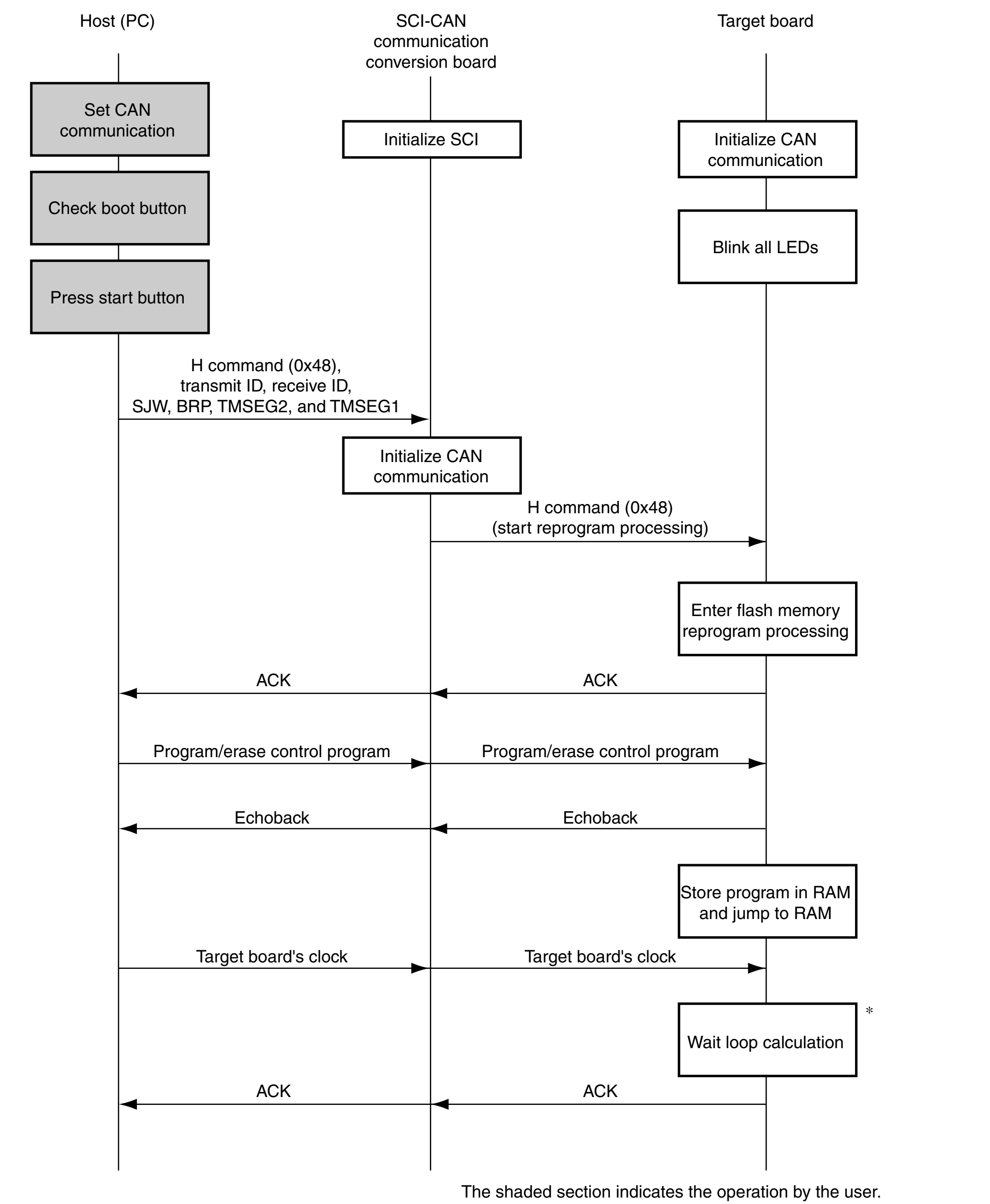
This results in: 1 Mbps = 20 MHz/{2 × ($\boxed{0}$ + 1) × (3 + $\boxed{4}$ + $\boxed{3}$)}

↑ BRP ↑ TSEG1 ↑ TSEG2

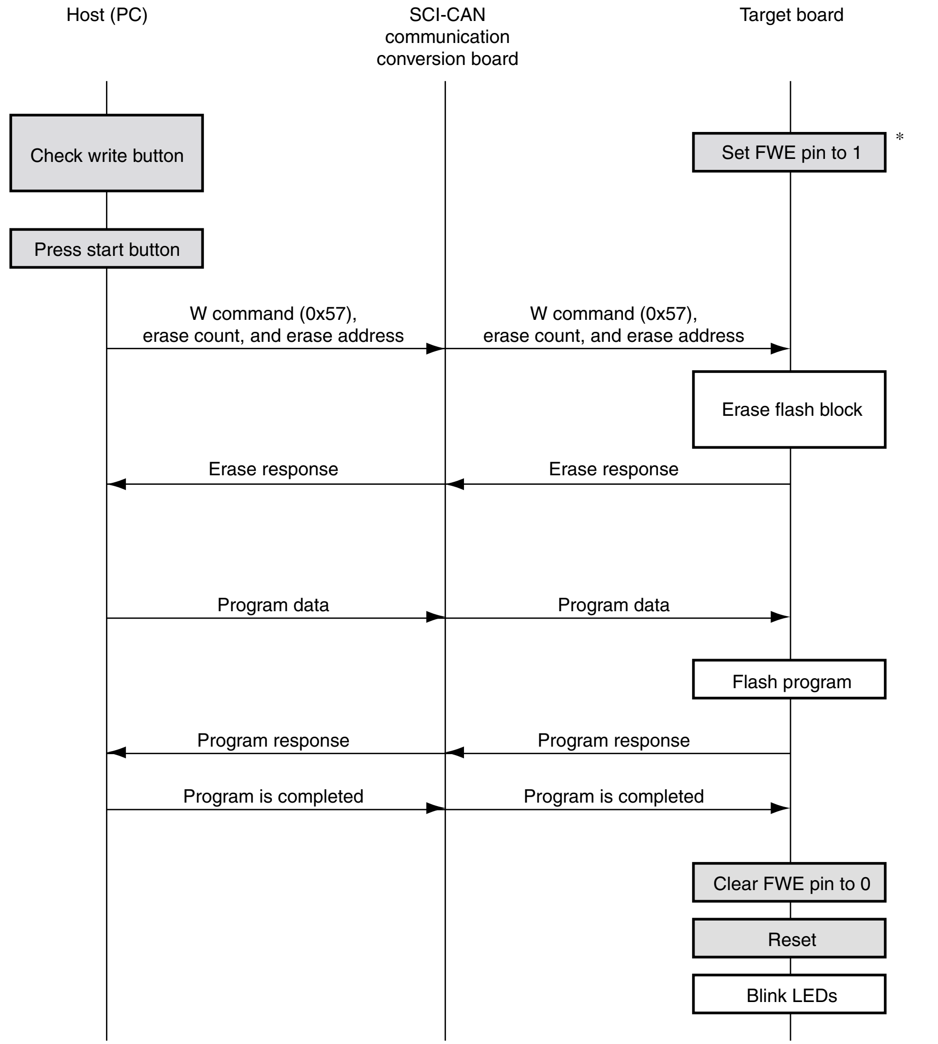
Data frame ID (standard format: 11 bits)

- Receive data ID on the communication conversion board: H′03F9 (11-bit display: 01111111001)
- Receive data ID on the target board: H′0602 (11-bit display: 11000000010)

3.4 Flash Memory Reprogramming Sequence in User Program Mode



Note: * The program/erase control program uses the software loop count to control the application time, etc. for bits E and P. Increment or decrement the loop count according to the operating frequency of the F-ZTAT microcomputer.



Note: * Before pressing the start button for programming, set the FWE pin to 1 to enter user program mode.

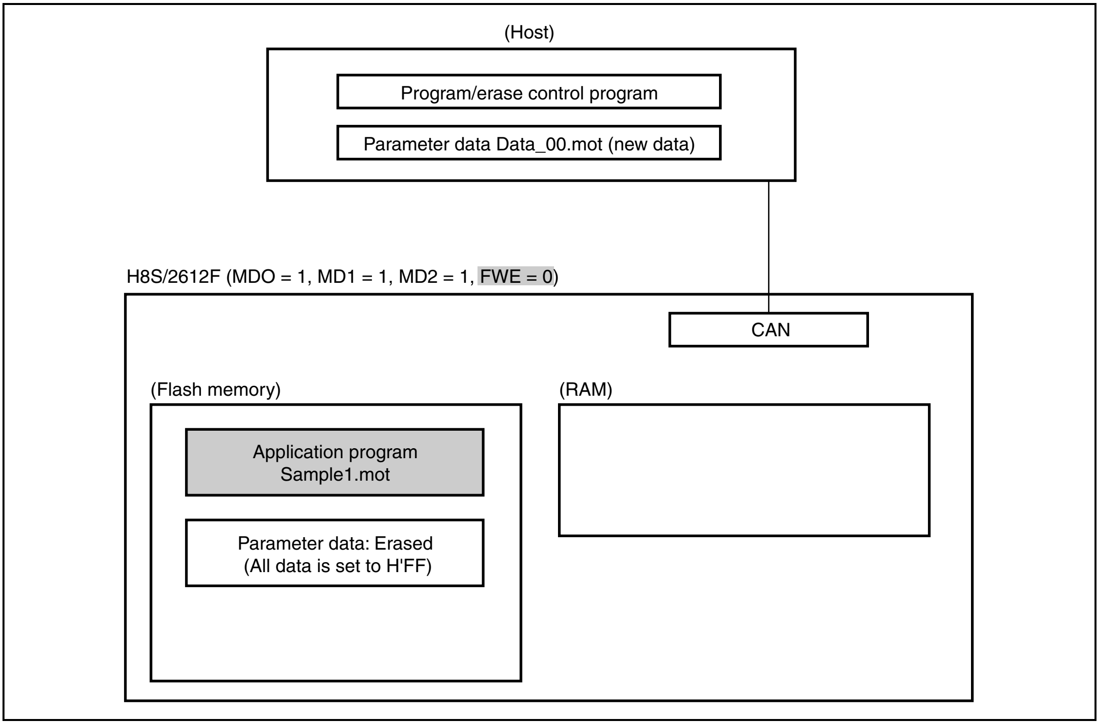
Remarks:

The settings of SCI communication between the host (PC) and SCI-CAN communication conversion board should be as follows:

- Mode: Asynchronous communication method
- Data format: 8-bit data, without parity, one stop bit
- Bit rate: 57.600 bit/sec

Section 4 Reprogramming Flash Memory in User Program Mode

4.1 Initial State

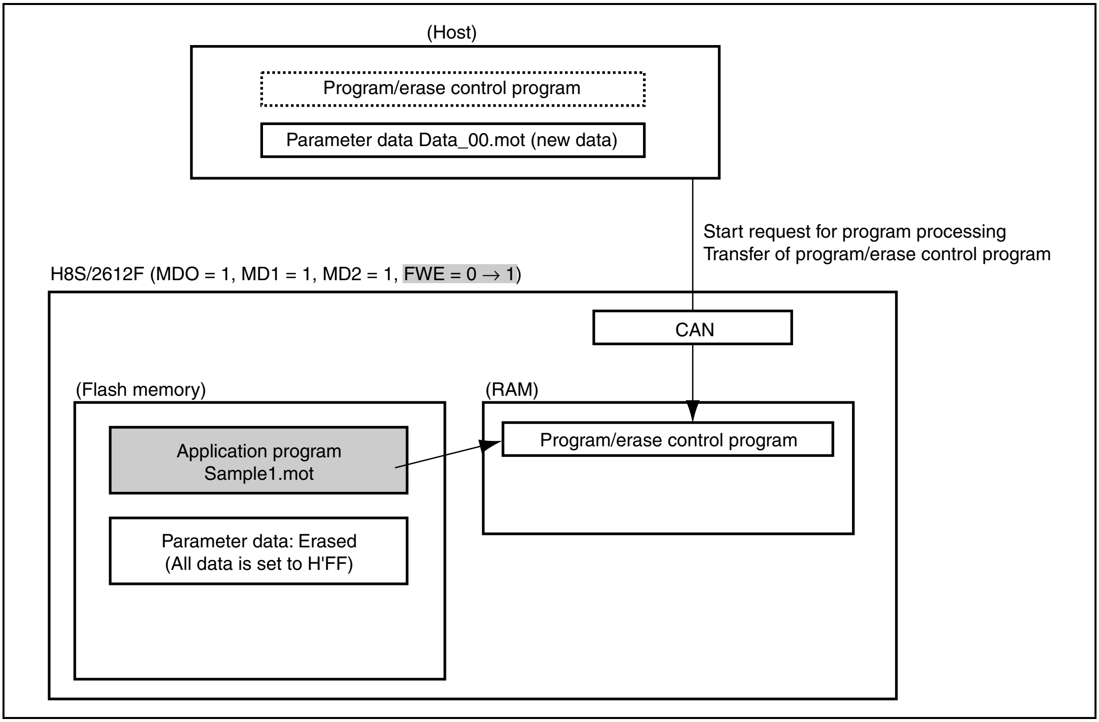


Description:

1. Restart the H8S/2612F in user mode (MD0 = 1, MD1 = 1, MD2 = 1, and FWE = 0).
2. The application program initializes the HCAN unit to accept the conditions for entering the program processing.
3. Since the parameter data area has been erased in boot mode, the data is set to H′FF. The application program lights all the LEDs.

Note: For details on how the application program operates the LEDs, see section 5.3, Application (Sample) Program.

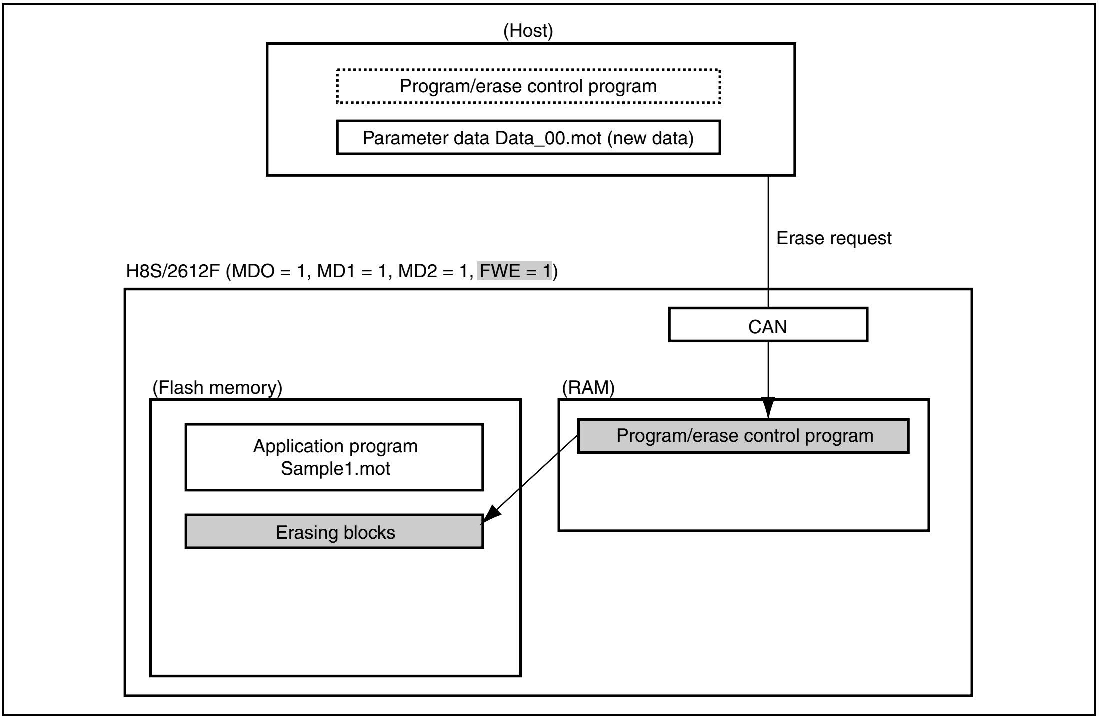
4.2 Transferring Program/Erase Control Program



Description:

1. The host uses HCAN communication to issue a request to start the reprogram processing.
2. The application program accepts this request, and then enters the transfer processing of the program/erase control program.
3. The host transfers the program/erase control program.
4. The application program stores the program/erase control program to be transferred in RAM. After completion of the transfer, the application program jumps to the program/erase control program in RAM.
5. The user uses the switch on the target board to set the FWE pin to on (1). Setting FWE = 1 clears the hardware protect to enable erasing and programming of the flash memory.

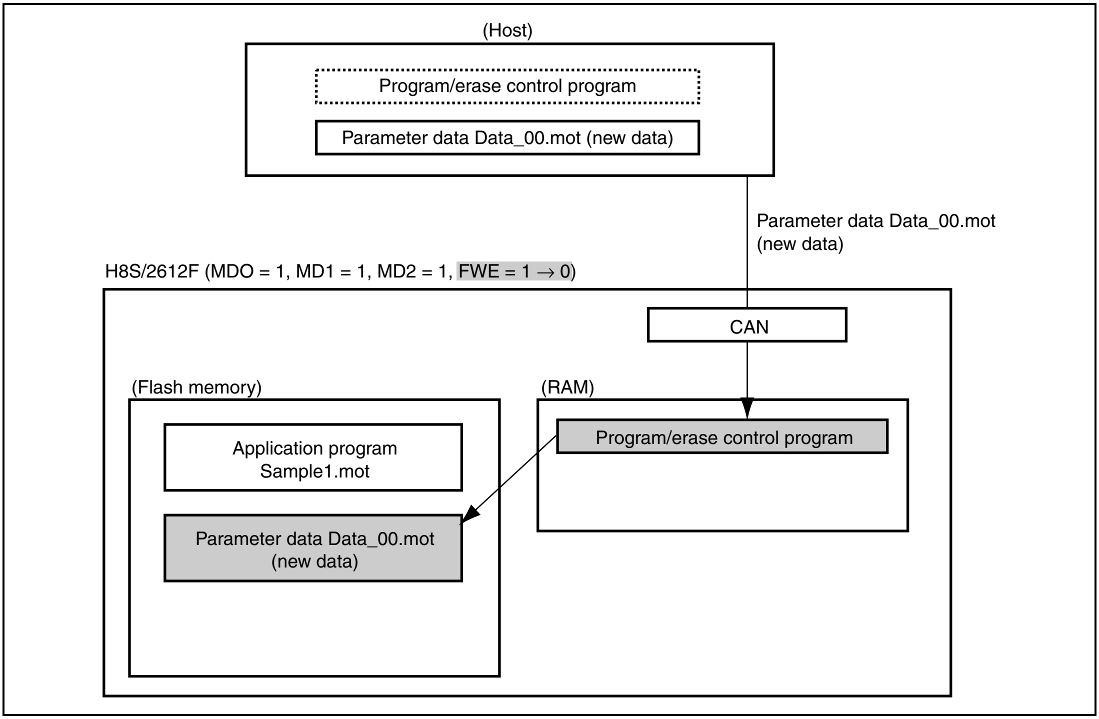
4.3 Erasing Blocks of Flash Memory



Description:

1. The host uses HCAN communication to issue a request for erasing blocks in the area in which the new data is to be programmed.
2. The program/erase control program in RAM erases the requested blocks. However, the program in the initial state only executes erase-verify without executing erase because all blocks have been erased.

4.4 Programming New Application Program



Description:

1. The host provides the program data (Data_00.mot) via HCAN communication.
2. The program/erase control program in RAM receives the program data and then programs it to the flash memory.
3. After the data has been programmed, set the FWE pin to off (0).
4. Restart the application program. Then, the application program refers to the parameter data (new data) and blinks the LEDs.

Note: For details on how the application program operates the LEDs, see section 5.3, Application (Sample) Program.

Section 5 Details of Software

5.1 F-ZTAT Microcomputer H-CAN Program

The F-ZTAT Microcomputer H-CAN Program (FlashCAN.exe) runs in the host (PC).

No.	File Name	Description
1	FlashCAN.exe	Main section of the F-ZTAT Microcomputer H-CAN Program
2	SCI2612F3.inf	Information file (for SCI communication) for the H8S/2612F microcomputer
3	SCI2612F3.sub	Load module of the program control program
4	HCAN2612F3.inf	Information file (for H-CAN communication) for the H8S/2612F microcomputer
5	HCAN2612F3.sub	Load module of the program/erase control program
6	HCAN2612F3.src	Source file of the program/erase control program
7	HCAN2612F3.bat	DOS prompt batch file that assembles and links the program/erase control program
8	Hew2.hws	Hew project file that assembles and links the program/erase control program

Example of the FlashCAN.exe window (Main window)

(Window for setting the CAN communication specifications)

5.2 SCI-HCAN Communication Conversion Program

The SCI-HCAN communication conversion program (SCItoCAN.mot) runs in the SCI-HCAN communication conversion board.

No.	File Name	Description
1	SCItoCAN.mot	On-chip load module for the H8S/2612F on the SCI-HCAN communication conversion board. This module converts the interface between the SCI and HCAN. It uses the on-chip SCI and HCAN for the H8S/2612F on the SCI-HCAN communication conversion board.

Note: Download the SCI-HCAN communication conversion program to the SCI-HCAN communication conversion board in boot mode in advance.

5.3 Application (Sample) Program

The application (sample) program (Sample1.mot) runs on the target board.

No.	File Name	Description
1	Sample1.mot	On-chip load module for the H8S/2612F on the target board. This module includes the flash-memory reprogram processing in user program mode. It provides the sample function that lights and blinks the LEDs to let you visually check the operation of the application.
2	Data_FF.motData_00.mot	Parameter data of the pattern by which the Sample1.mot lights and blinks the LEDs. H′FF (erased): Lights all the LEDs. H′00: Blinks the LEDs at one-second intervals.
3	HCAN_up.c	Sample1.mot source file that includes the flash-memory reprogram processing
4	Sample1.c	Sample1.mot source file that includes the function for lighting and blinking the LEDs
5	io_2612.h	Include file that defines the on-chip I/O register. This file has been included from HCAN_up.c and Sample.c.
6	Sample1.bat	DOS prompt batch file that compiles and links the program.
7	Hew2.hws	Hew project file that compiles and links the program.

Section 6 Example of Creating Application (Sample) Program

6.1 Functions and Variables

1. List of Functions

Source name: HCAN_up.c

Abbreviation	Module Name	Description
PowerON_Reset	Power-on reset processing	Performs initialization. This function checks whether the H command (0x48) is received, and then switches the user program processing and sample program.
UserProgramMode_Main	User program processing	Transfers the program/erase control program.
InitHCAN	HCAN initialization	Initializes HCAN communication.
CAN_MB5_trs1byte	Transmitting CAN one-byte	Transmits data (HCAN communication).
CAN_MB4_rcv1byte	Receiving CAN one-byte	Receives data (HCAN communication).

Source name: Sample1.c

Abbreviation	Module Name	Description
UserApli	LED display switching	Checks the parameters and controls the LEDs (lights or blinks all LEDs).

2. Variables

All the variables in use are internal variables.

6.2 Example of Changing CAN Communication Settings

1. Change the bit rate from 1000 kbit/s to 500 kbit/s.
   In (1) HCAN.BCR.WORD = 0x0034;, change 0x0034 to 0x0134;.
   The baud rate prescaler is set to 4 system clocks, and the bit rate is set to 500 kbit/s.
   See the register description for BCR in the hardware manual.
2. Change the data length from 1 byte to 8 bytes.
   In (2) HCAN.MC[4][1-1] = 0x01;, change 0x01; to 0x08;.
   The receive data length is set to 8 bytes.
   In (5) HCAN.MC[5][1-1] = 0x01;, change 0x01; to 0x08;.
   The transmit data length is set to 8 bytes.
   See the register description for MC0 to MC15 in the hardware manual.
3. Change the receive mailbox ID from 0x0602 to 0x0005.
   In (3) HCAN.MC[4][5-1] = 0x20;, change 0x20; to 0xA0;.
   In (4) HCAN.MC[4][6-1] = 0x7F;, change 0x7F; to 0x00;.
   The receive mailbox ID (ID28 to ID18) is set to 0x0005.
   In (6) HCAN.MC[5][5-1] = 0x40;, change 0x40; to 0x80;.
   In (7) HCAN.MC[5][6-1] = 0xC0;, change 0xC0; to 0x00;.
   The transmit mailbox ID (ID28 to ID18) is set to 0x0004.
   See the register description for MC0 to MC15 in the hardware manual.

Program lists (1) to (3)

Source name: HCAN_up.c

Module name: InitHCAN

```
/**********************************************************************************************/
/*  HCAN initialization                                                                       */
/*  • Baud rate 1000kbps, TSEG1=4,TSEG2=3,BRP=0,SJW=0                                        */
/*  • MB4: Receiving "flash-memory reprogram request" [ID(11bit):03F9]                       */
/*  • MB5: Transmitting "flash-memory reprogram response" [ID(11bit):0602]                   */
/*  Remarks: Since communication will take place during operation in RAM, set program to     */
/*  prohibit any interrupt.                                                                   */
/**********************************************************************************************/
void InitHCAN( void )
{
  int i, j;                           /* Loop counter */
  /* Clear HCAN module stop bit (MSTPCRC) */
  System.MSTPCRC.BIT.HCANCKSTP = 0;
(1) HCAN.BCR.WORD = 0x0034; →0x0134; /* 1000-kbps (at 20 MHz) baud rate setting            */
  HCAN.MBCR.WORD   = 0x1100;          /* Transmission and reception setting for mailboxes
  MB4=Rcv */
  /* Clear MC [4-5] [0-7] and MD [4-5] [0-7] to all 0 */
  for ( j = 4; j <= 5; j++ ) {
      for ( i = 0; i < 8; i++ ) {
           HCAN.MC[j][i] = 0x00;      /* MC[4][0] to MC[5][7] */
           HCAN.MD[j][i] = 0x00;      /* MD[4][0] to MD[5][7] */
      }
  }
  HCAN.MCR.BYTE = 0x04;               /* Transmission method: Order of mailboxes */
  while ( HCAN.GSR.BIT.GSR3 == 1 );   /* While not in HCAN normal mode */
                                      /* Enter HCAN normal mode */
  /* Settings for mailbox 4 (receive: flash-memory reprogram request) */
(2) HCAN.MC[4][1-1] = 0x01; →0x08;    /* MB4 data length = 1 byte                          */
  HCAN.MC[4][5-1] = 0x00;             /*     Data frame, standard format                   */
(3) HCAN.MC[4][5-1] = 0x20; →0xA0;    /* ID: x xxxx xxx0 01-- ---- ---- ---- ----          */
(4) HCAN.MC[4][6-1] = 0x7F; →0x00;    /* ID: 0 1111 111x xx-- ---- ---- ---- ----          */
  /* Settings for mailbox 5 (transmit: flash-memory reprogram response) */
(5) HCAN.MC[5][1-1] = 0x01; →0x08;    /* MB5 data length = 1 byte                          */
  HCAN.MC[5][5-1] = 0x00;             /*     Data frame, standard format                   */
(6) HCAN.MC[5][5-1] = 0x40; →0x80;    /* ID: x xxxx xxx0 10-- ---- ---- ---- ----          */
(7) HCAN.MC[5][6-1] = 0xC0; →0x00;    /* ID: 1 1000 000x xx-- ---- ---- ---- ----          */
}
```

6.3 Example of Changing Receive and Transmit Mailbox Numbers for CAN Communications

Change the receive and transmit mailbox numbers (change [4] to [14] for the receive mailbox, and [5] to [15] for the transmit mailbox).

1. Change the transmit/receive direction of the mailbox.
   In (1) HCAN.MBCR.WORD = 0x1100; change 0x1100; to 0x0140;.
   Only MBCR14 is set to 1, and only mailbox 14 is set for reception.
   See the register description for MBCR in the hardware manual.
2. Change the initialization for the mailbox.
   In (2), change for(j=4;j<=5;j++) to for(j=14;j<=15;j++){.
   Only mailboxes 14 and 15 will be initialized.
   See the register description for MC0 to MC15 and MD0 to MD15 in the hardware manual.
3. Change the settings for the mailbox.
   In (3) to (6), change HCAN.MC[4] to HCAN.MC[14].
   In (7) to (10), change HCAN.MC[5] to HCAN.MC[15].
   The receive mailbox [14] and transmit mailbox [15] are set.
   See the register description for MC0 to MC15 in the hardware manual.
4. Change the trigger receive processing.
   In (11), change if(HCAN.RXPR.BIT.RXPR4==1){ to if(HCAN.RXPR.BIT.RXPR14==1){.
   See the register description for RXPR in the hardware manual.
5. Change the receive processing.
   In (12), change while(HCAN.RXPR.BIT.RXPR4==0); to while(HCAN.RXPR.BIT.RXPR14==0);.
   In (13) HCAN.RXPR.WORD=0x1000;, change 0x1000; to 0x0040;.
   In (14), change return HCAN.MD[4][0] to return HCAN.MD[14][0];.
   See the register description for RXPR and MD0 to MD15 in the hardware manual.
6. Change the transmit processing.
   In (15), change HCAN.MD[5][0] to HCAN.MD[15][0].
   In (16), change HCAN.TXPR.BIT.TXPR5 to HCAN.TXPR.BIT.TXPR15.
   In (17), change while(HCAN.TXPR.BIT.TXPR5==0); to while(HCAN.TXPR.BIT.TXPR15==0);.
   In (18) HCAN.TXACK.WORD=0x2000;, change 0x2000; to 0x0080;.
   See the register description for MD0 to MD15, TXPR, and TXACK in the hardware manual.

Program lists (1) to (3)
Source name: HCAN_up.c
Module name: InitHCAN

```
/*****************************************************************************************/
/* HCAN initialization                                                                  */
/* • Baud rate: 1000kbps, TSEG1=4,TSEG2=3,BRP=0,SJW=0                                   */
/* • MB4: Receiving "flash-memory reprogram request" [ID(11bit):03F9]                   */
/* • MB5: Transmitting "flash-memory reprogram response" [ID(11bit):0602]               */
/* Remarks: Since communication will take place during operation in RAM, set program to */
/* prohibit any interrupt.                                                              */
/*****************************************************************************************/
void InitHCAN( void )
{
  int i, j;                            /* Loop counter */
  /* Clear HCAN module stop bit (MSTPCRC)*/
  System.MSTPCRC.BIT.HCANCKSTP = 0;
  HCAN.BCR.WORD    = 0x0034;           /* 1000-kbps (at 20 MHz) baud rate setting */
(1) HCAN.MBCR.WORD = 0x1100; →0x0140; /* Transmission and reception setting for mailboxes
MB4=Rcv */
  /* Clear MC [4-5] [0-7] and MD [4-5] [0-7] to all 0 */
(2) for ( j = 4; j <= 5; j++ ) { →for ( j = 14; j <= 15; j++ )
        for ( i = 0; i < 8; i++ ) {
             HCAN.MC[j][i] = 0x00;     /* MC[4][0] to MC[5][7] */
             HCAN.MD[j][i] = 0x00;     /* MD[4][0] to MD[5][7] */
        }
  }
  HCAN.MCR.BYTE = 0x04;                /* Transmission method: Order of mailboxes */
  while ( HCAN.GSR.BIT.GSR3 == 1 );    /* While not in HCAN normal mode */
                                       /* Enter HCAN normal mode */
  /* Settings for mailbox 4 (receive: flash-memory reprogram request) */
(3) HCAN.MC[4] → [14] [1-1] = 0x01;   /* MB4 data length = 1 byte                      */
(4) HCAN.MC[4] → [14] [5-1] = 0x00;   /*     Data frame, standard format               */
(5) HCAN.MC[4] → [14] [5-1] = 0x20;   /* ID: x xxxx xxx0 01-- ---- ---- ---- ----      */
(6) HCAN.MC[4] → [14] [6-1] = 0x7F;   /* ID: 0 1111 111x xx-- ---- ---- ---- ----      */
  /* Settings for mailbox 5 (transmit: flash-memory reprogram response) */
(7) HCAN.MC[5] → [15] [1-1] = 0x01;   /* MB5 data length = 1 byte                      */
(8) HCAN.MC[5] → [15] [5-1] = 0x00;   /*     Data frame, standard format               */
(9) HCAN.MC[5] → [15] [5-1] = 0x40;   /* ID: x xxxx xxx0 10-- ---- ---- ---- ----      */
(10)HCAN.MC[5] → [15] [6-1] = 0xC0;   /* ID: 1 1000 000x xx-- ---- ---- ---- ----      */
}
```

Program list (4)

Source name: HCAN_up.c

Module name: PowerOn_Reset

```
/*****************************************************************************************/
/* Power-on reset processing                                                             */
/*           Waiting for trigger for entering user program mode (receiving H command)    */
/*           Switching between sample program and user program mode                     */
/*****************************************************************************************/
#pragma section _BOOT                  /* Section name "P_BOOT" */
void PowerON_Reset(void){              /* Power-on reset (vector number 0) handler */
  volatile unsigned char LatchMDCR;
  unsigned char Data;
  LatchMDCR = System.MDCR.BYTE;        /* Latch MD2 to MD0 */
  /* Prohibit interrupt */
  set_imask_ccr(1);
  set_imask_exr(7);
  HCAN.IRR.WORD   = 0x0100;            /* Write 1 to IRR0 and clear to 0 */
  InitHCAN();
  /* User program */
  Data = 0x00;
  PORT.PDDDR.BYTE = 0xFF;              /* Initialize LEDs */
  PORT.PDDR.BYTE = 0xFF;
  while(Data != 'H'){                  /* H command? */
(11) if(HCAN.RXPR.BIT.RXPR4 →RXPR14 == 1){ /* Has any data been received? */
        Data = CAN_MB4_rcv1byte();
     }else{
        UserApli();                    /* Sample program */
     }
  }
  PORT.PDDR.BYTE = 0xFF;               /* Turn off LEDs */
  UserProgramMode_Main();
  sleep();                             /* (Do not come here) */
}
```

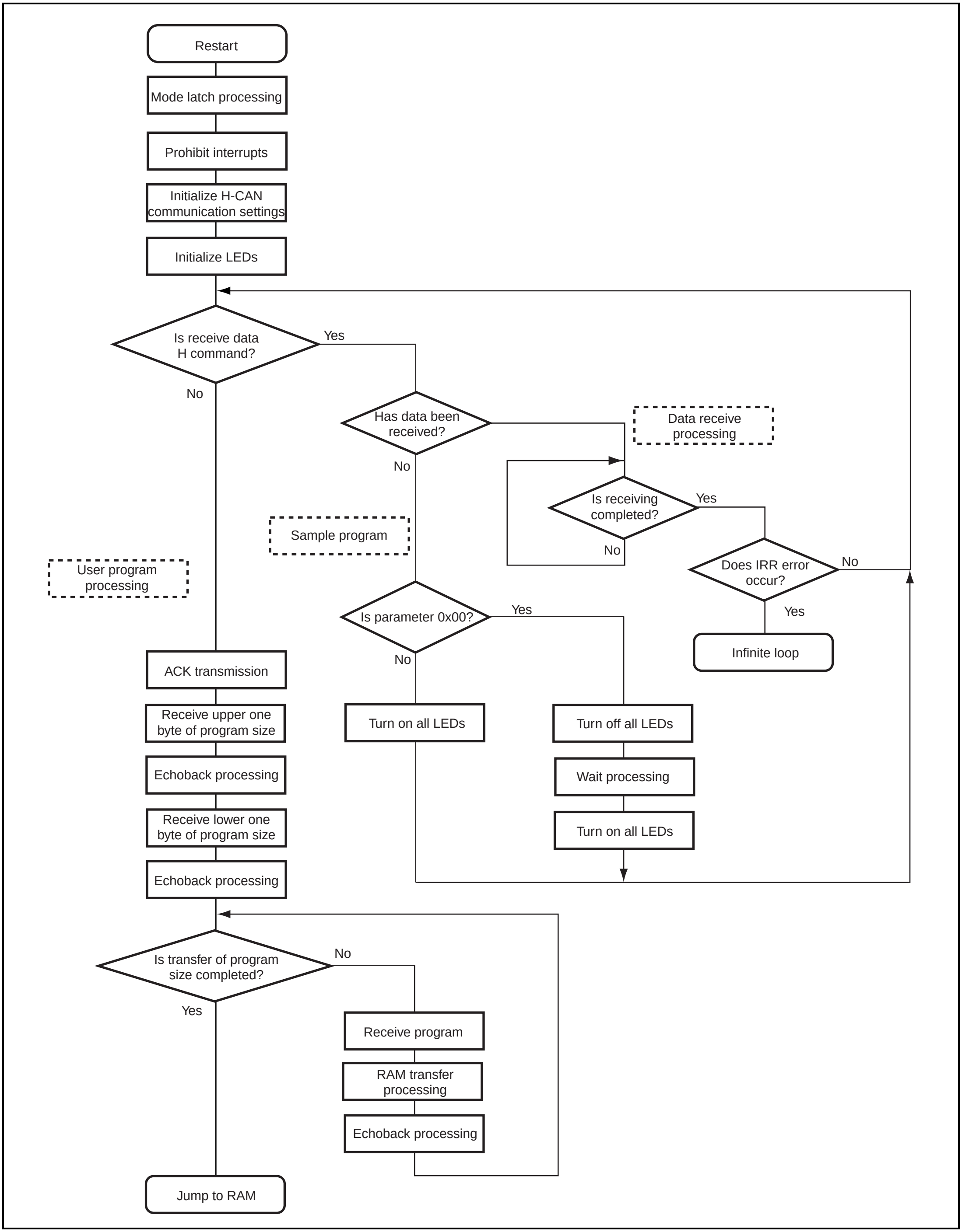
Program list (5)
Source name: HCAN_up.c
Module name: CAN_MB4_rcv1byte

```
/*****************************************************************************************/
/* Receive CAN one-byte                                                                  */
/*****************************************************************************************/
unsigned char CAN_MB4_rcv1byte( void ){
(12) while( HCAN.RXPR.BIT.RXPR4 →RXPR14 == 0 ); /* Wait for MB4 to complete receiving */
        if( (HCAN.IRR.WORD & 0x1802) != 0 ){
           while(1);                                  /* Infinite loop */
        }
(13) HCAN.RXPR.WORD = 0x1000; →0x0040;                /* Clear receive flag */
(14) return HCAN.MD[4] → [14] [0];                    /* Receive one byte from MD4 */
}
```

Program list (6)
Source name: HCAN_up.c
Module name: CAN_MB5_trs1byte

```
/*****************************************************************************************/
/* Receive CAN one-byte                                                                  */
/*****************************************************************************************/
void CAN_MB5_trs1byte( unsigned char TrsData ){
(15) HCAN.MD[5] → [15] [0] = TrsData;                 /* Set one byte of transmit data to MD5 */
(16) HCAN.TXPR.BIT.TXPR5 →TXPR15  |= 1;               /* Start transmission */
(17) while( HCAN.TXPR.BIT.TXPR5 →TXPR15  == 0 );/* Wait for completion of transmission */
    if( (HCAN.IRR.WORD & 0x1802) != 0 ){
       while(1);                                      /* Infinite loop */
    }
(18) HCAN.TXACK.WORD = 0x2000; →0x0080;               /* Transmission completed */
}
```

6.4 Application (Sample) Program Flowchart



RENESAS

Section 7 Example of Creating Program/Erase Control Program

7.1 Overview

The following describes the calculation of the application time for the erase (E) and program (P) bits, and the wait time for the SWE bit. The parenthesized value indicates the unit.

- Calculation methods
  1. Time required for one cycle = 1 (sec) ÷ target clock (MHz)
  2. Time required for one loop = Time required for one cycle (µs) × number of cycles required for one loop
  3. Wait time = Target clock (MHz) × stipulated wait time (µs)
  4. Wait loop count = Wait time ÷ time required for one loop
  5. Wait time other than after setting of the P or E bit = Wait loop count + 1

     (Since this wait time must be equal to or greater than the stipulated wait time, add 1 to the wait loop count obtained by the calculation.)
  6. Wait time after setting of the P or E bit = Wait loop count

     (Since this wait time must be equal to or smaller than the stipulated wait time, use the wait loop count obtained by the calculation.)

- Expressions

  The following shows examples of calculating the wait time:

  Prerequisites: Target clock = 20 (MHz)

  Number of cycles required for one loop = 4 (cycles)

Example 1: Obtain the wait time after the clearing of the SWE bit (100 µs or longer).

1. 20 (MHz) × 1000* = 20000 (kHz)
2. 20000 (kHz) × 100 (µs) = 2000000
3. 2000000 ÷ 4 (cycles) × 1000* = 500 (times)
4. 500 (times) + 1 (time) = 501 (times)
5. WLOOP100 = 501
6. Time required for one cycle: 1 (sec) ÷ 20 (MHz) = 0.05 (µs)
7. Time required for one loop: 0.05 (µs) × 4 (cycles) = 0.2 (µs)
8. Wait loop time: 0.2 (µs) × 501 (times) = 100.2 (µs)

Example 2: Obtain the wait time after the setting of the E bit (within 10 ms).

1. 20 (MHz) × 1000* = 20000 (kHz)
2. 10000 (µs) ÷ 1000* = 10 (ms)

3. $20000 (kHz) \times 10 (ms) = 200000$
4. $200000 \div 4 (cycles) = 50000 (times)$
5. WTIME10000 = 50000
6. Time required for one cycle: 1 (sec) ÷ 20 (MHz) = 0.05 (μs)
7. Time required for one loop: 0.05 (μs) × 4 (cycles) = 0.2 (μs)
8. Wait loop time: 0.2 (μs) × 50000 (times) = 10000 (μs) = 10 (ms)

Note: * To minimize error, the above examples convert the unit of frequency from MHz to kHz to use the thousandfold values in calculation. Then, the results are divided by 1000 to obtain the correct count.

1. Erasing Flash Memory

(1) Erase

Erase functionality erases the flash memory block by block. To use erase, set 1 to the SWE bit in the flash memory control register (FLMCR), and then use the erase block register (EBR) to set one bit of the area in the flash memory to be erased. Then, set the ESU bit in FLMCR to prepare for erase mode (erase mode setup), and set the E bit in FLMCR to shift the operating mode to erase mode. The period of time during which the E bit is set is the erase time. After the erase time has elapsed, clear the E, ESU, and SWE bits in FLMCR to cancel erase mode.

(2) Erase-Verify

Erase-verify functionality verifies whether the flash memory has been erased successfully. To use erase-verify, set the SWE bit and then the EV bit in FLMCR to shift the operating mode to erase-verify mode. Before the flash memory is read in erase-verify mode, a dummy writing of data (H′FF) is performed on the address to be read. When the flash memory is subsequently read (the verify data is read in 16-bit units), the data at the latched address is read. If the read data has been erased (that is, all data is 1), a dummy writing of the next data is performed. Then erase-verify is performed for the next data. Upon completion of the verify operation, clear the EV and SWE bits in FLMCR to cancel erase-verify mode.

The following shows the wait time for each bit in FLMCR for erasing the flash memory.

Table 7.1 FLMCR Bits and Wait Time for Erasing Flash Memory

Set or Clear Each Bit	Wait Time (Standard Value)	Wait Time at 20 MHz
SWE set	1 µs or greater	1.2 µs
ESU set	100 µs or greater	100.2 µs
E set	Up to 10 ms	10.0 ms
E cleared	10 µs or greater	10.2 µs
ESU cleared	10 µs or greater	10.2 µs
EV set	20 µs or greater	20.2 µs
Dummy write	2 µs or greater	2.2 µs
EV cleared	4 µs or greater	4.2 µs
SWE cleared	100 µs or greater	100.2 µs
Maximum count	100 times	100 times

2. Programming Flash Memory

(1) Program

Program functionality programs data to the flash memory. One program is performed in 128-byte units. To use program, set 1 to the SWE bit in the flash memory control register (FLMCR). Then, the system stores the 128-byte program data in the program data area and reprogram data area, and then sequentially programs 128 bytes of data from the program data area in RAM to the target address (that is, byte-by-byte data transfer is sequentially performed 128 times). Note that the lower eight bits of the target start address must be H′00 or H′80. The program address and program data are latched in the flash memory.

A program of data of less than 128 bytes also requires a transfer of 128-byte data, for which data H′FF must be programmed to unnecessary address. Then, set the PSU bit in FLMCR to prepare for program mode (program mode setup), and set the P bit in FLMCR to shift the operating mode to program mode.

The period of time during which the P bit is set is the program time for the flash memory. After the program time has elapsed, clear the P, PSU, and SWE bits in FLMCR to cancel program mode. (Note that an additional program is performed for the programmed bits for up to the sixth initial program.)

(2) Program-Verify

Program-verify functionality verifies whether the flash memory has been programmed successfully. Set the SWE bit and then the PV bit in FLMCR to shift the operating mode to program-verify mode.

Before the flash memory is read in program-verify mode, a dummy writing of data (H′FF) is performed on the address to be read. When the flash memory is subsequently read (the verify data is read in 16-bit units), the data at the latched address is read. Then, the system compares the programmed data with the verify data, calculates the reprogram data, and then transfers the reprogram data to the reprogram data area. Upon completion of the verification of the data of 128 bytes, clear the PV and SWE bits in FLMCR to cancel program-verify mode.

The following shows the wait time for each bit in FLMCR for programming the flash memory. (Note that additional program data is calculated and transferred to the additional program data area for up to the sixth initial program.)

Table 7.2 FLMCR Bits and Wait Time for Programming Flash Memory

Set or Clear Each Bit	Wait Time (Standard Value)	Wait Time at 20 MHz
SWE set	1 µs or greater	1.2 µs
PSU set	500 µs or greater	50.2 µs
P set (1st to 6th) (1st to 6th) additional (7th to 1000th)	Up to 30 µs Up to 10 µs Up to 200 µs	30.0 µs 10.0 µs 200.0 µs
P cleared	5 µs or greater	5.2 µs
PSU cleared	5 µs or greater	5.2 µs
PV set	20 µs or greater	20.2 µs
Dummy write	2 µs or greater	2.2 µs
PV cleared	2 µs or greater	2.2 µs
SWE cleared	100 µs or greater	100.2 µs
Maximum count	1000 times	1000 times

7.2 Functions, Variables, and Constants

1. Functions

Source name: HCAN2612f3.src

Abbreviation	Module Name	Description
MAIN	Main processing	Receives the initialization target clock for the stack, receives a check command, and performs check. (W and C commands)
WLOOP_INI	Wait loop initialization processing	Initializes the wait loop.
WAITLOOP_CAL	Wait loop calculation main processing	Calculates and sets the wait loop.
WLOOP_CAL	Wait loop calculation processing	Calculates the wait loop.
WCMD	W command processing	Performs erase processing, program processing, and checksum processing
GET_EADR	Erase address receive processing	Receives the start address of the erase block.
BLK_CHECK	Specified block check processing	Checks the specified block.
GET_WADR	Program address receive processing	Receives the program address (four bytes).
GET_BUFFER	Program data receive processing	Receives the program data (128 bytes).
RCVNBYTE	N-byte receive processing	Receives N bytes of data.
RCV1BYTE	One-byte receive processing	Receives one byte of data.
TRS1BYTE	One-byte transmit processing	Transmits one byte of data.
XON_CHECK	XON check processing	Checks a response during program processing.
FWRITE128	Flash memory 128-byte program processing	Performs: Initial program and verify (for the first to the sixth program). Initial program, and program-verify (before the program). Additional program (for the first to the sixth program). Reprogram and program-verify (for the 7th to the 1000th program).

Abbreviation	Module Name	Description
FWRITEVF	Program-verify processing	Performs program-verify, creates reprogram data, and creates additional program data.
FWRITE	Flash memory program processing	Programs data to the flash memory.
BLK1_ERASE	One-block erase processing	Checks the specified block. Performs initial erase-verify (before the erasing). Erases data, and performs erase-verify (for the first to the 100th erase).
FERASEVF	Erase-verify processing	Performs erase-verify.
FERASE	Flash memory erase processing	Erases the flash memory.
CHECKSUM	Checksum processing	Calculates and transmits the checksum value (four bytes).

2. Variables

Abbreviation	Variable Name	Description	Size
W_ADR	Program address	Stores the program address.	4 bytes
W_BUF	Program buffer	Stores the program data.	128 bytes
BUFF	Buffer	Stores the reprogram data.	128 bytes
OWBUFF	Additional program buffer	Stores the additional program data.	128 bytes
COUNT	Counter	Counter for the number of times the erase count is programmed to the counter	2 bytes
EVF_ST	Erase start address	Stores the start address of the erase block.	4 bytes
EVF_ED	Erase end address	Stores the end address of the erase block.	4 bytes
BLK_NO	Erase specification block No.	Stores the erase specification block number.	1 byte
VF_RET	Verification result flag	Result of erase-verify and program-verify	1 byte
RESTSIZE	Program data size	Stores the program data size.	4 bytes
E_ADR	Erase block address	Stores the erase block address.	64 bytes
E_ADR_PTR	Erase block address pointer	Pointer to the erase block address.	4 bytes
WORKCLK	Clock	Stores the target clock.	4 bytes
ERASEBLOCK	Erase block count	Stores the number of erase blocks.	1 byte
WLOOP1	Wait 1 µ	Stores the loop count for the 1 µs wait.	2 bytes
WLOOP2	Wait 2 µ	Stores the loop count for the 2 µs wait.	2 bytes

Abbreviation	Variable Name	Description	Size
WLOOP4	Wait 4 µ	Stores the loop count for the 4 µs wait.	2 bytes
WLOOP5	Wait 5 µ	Stores the loop count for the 5 µs wait.	2 bytes
WLOOP10	Wait 10 µ	Stores the loop count for the 10 µs wait.	2 bytes
WLOOP20	Wait 20 µ	Stores the loop count for the 20 µs wait.	2 bytes
WLOOP50	Wait 50 µ	Stores the loop count for the 50 µs wait.	2 bytes
WLOOP100	Wait 100 µ	Stores the loop count for the 100 µs wait.	2 bytes
WTIME10	Program wait 10 µ	Stores the loop count for the 10 µs wait during an additional program.	4 bytes
WTIME30	Program wait 30 µ	Stores the loop count for the 30 µs wait during an initial program.	4 bytes
WTIME200	Program wait 200 µ	Stores the loop count for the 200 µs wait during a reprogram.	4 bytes
WTIME1000	Erase wait loop 10 m	Stores the loop count for the 10 ms wait during erasing.	4 bytes

7.3 Example of Changing Receive and Transmit Mailbox Numbers for CAN Communications

Change the receive and transmit mailbox numbers (change [4] to [14] for the receive mailbox, and [5] to [15] for the transmit mailbox).

1. Change the registers and bits.

   Add the following definitions for the registers and bits in (1) to (9):

   (1) RXPR14_MOV .EQU H'0040

   (2) RXPR_L .EQU H'FFF80F

   (3) RXPR14 .EQU 6

   (4) MD14_0 .EQU H'FFF920

   (5) MD15_0 .EQU H'FFF928

   (6) TXPR_L .EQU H'FFF807

   (7) TXPR15 .EQU 7

   (8) TXACK15 .EQU 7

   (9) TXACK15_MOV .EQU H'0080

   See the register description for RXPR, TXPR, TXACK, and MD0 to MD15 in the hardware manual.

2. Change the settings of receive processing.

   In (10) and (11), change RXPR4 to RXPR14.

   In (10), change RXPR to RXPR_L.

   In (12), change MD4_0 to MD14_0.

   In (13), change RXPR4_MOV to RXPR14_MOV.

   See the register description for RXPR and MD0 to MD15 in the hardware manual.

3. Change the settings of transmit processing.

   In (14), change MD5_0 to MD15_0.

   In (15) and (17), change TXPR to TXPR_L.

   In (16), (18), and (19), change TXPR5 to TXPR15.

   In (20), change TXACK5_MOV to TXACK15_MOV.

   See the register description for TXPR, TXACK, and MD0 to MD15 in the hardware manual.

Program list (1)

Source name: HCAN2612f3.src

Module name: Data

```
; HCAN
RXPR                .EQU    H'FFF80E            ; Receive complete register (16 bits)
 RXPR4:             .EQU    4
 RXPR4_MOV:         .EQU    H'1000
(1) RXPR14_MOV      .EQU    H'0040
(2) RXPR_L          .EQU    H'FFF80F            ; Receive complete register (lower 8 bits)
(3) RXPR14          .EQU    6
MD4_0               .EQU    H'FFF8D0
MD5_0               .EQU    H'FFF8D8
(4) MD14_0          .EQU    H'FFF920
(5) MD15_0          .EQU    H'FFF928
TXPR                .EQU    H'FFF806
 TXPR5:             .EQU    5
(6) TXPR_L          .EQU    H'FFF807            ; Transmit wait register (lower 8 bits)
(7) TXPR15          .EQU    7
TXACK               .EQU    H'FFF80A            ; Transmit acknowledge register
 TXACK5:            .EQU    5
 TXACK5_MOV:        .EQU    H'2000
(8) TXACK15         .EQU    7
(9) TXACK15_MOV     .EQU    H'0080
IRR                 .EQU    H'FFF812
 IRR_ERR:           .EQU    H'1802
```

Program list (2)

Source name: HCAN2612f3.src

Module name: RCV1BYTE

```
;*****************************************************************************************
; * TITLE    / H-CAN 1 BYTE DATA RECEPTION                                              *
; * FUNCTION / RECEIVE 1 BYTE DATA                                                      *
; * INPUT    / -                                                                        *
; * OUTPUT   / R2L = RECEIVED DATA                                                      *
;*****************************************************************************************
RCV1BYTE     .EQU     $
                      SUB.W    R0,R0
(10)                  BLD.B    #RXPR4,@RXPR →#RXPR14,@RXPR_L
(11)                  BST.B    #RXPR4 →#RXPR14,R0L
                      MOV.W    R0,R0
                      BEQ      RCV1BYTE
;
                      MOV.W    @IRR,R0                       ; ERROR CHECK
                      AND.W    #IRR_ERR,R0
                      BNE      RCV_ERR
;
(12)                  MOV.B    @MD4_0 →@MD14_0,R2L           ; RECEIVE DATA TO R0H
;
(13)                  MOV.W    #RXPR4_MOV →#RXPR14_MOV,R0
                      MOV.W    R0,@RXPR                      ; RXPR4 CLEAR
                      RTS
;
RCV_ERR      BRA      RCV_ERR                                ; INFINITE LOOP
;
```

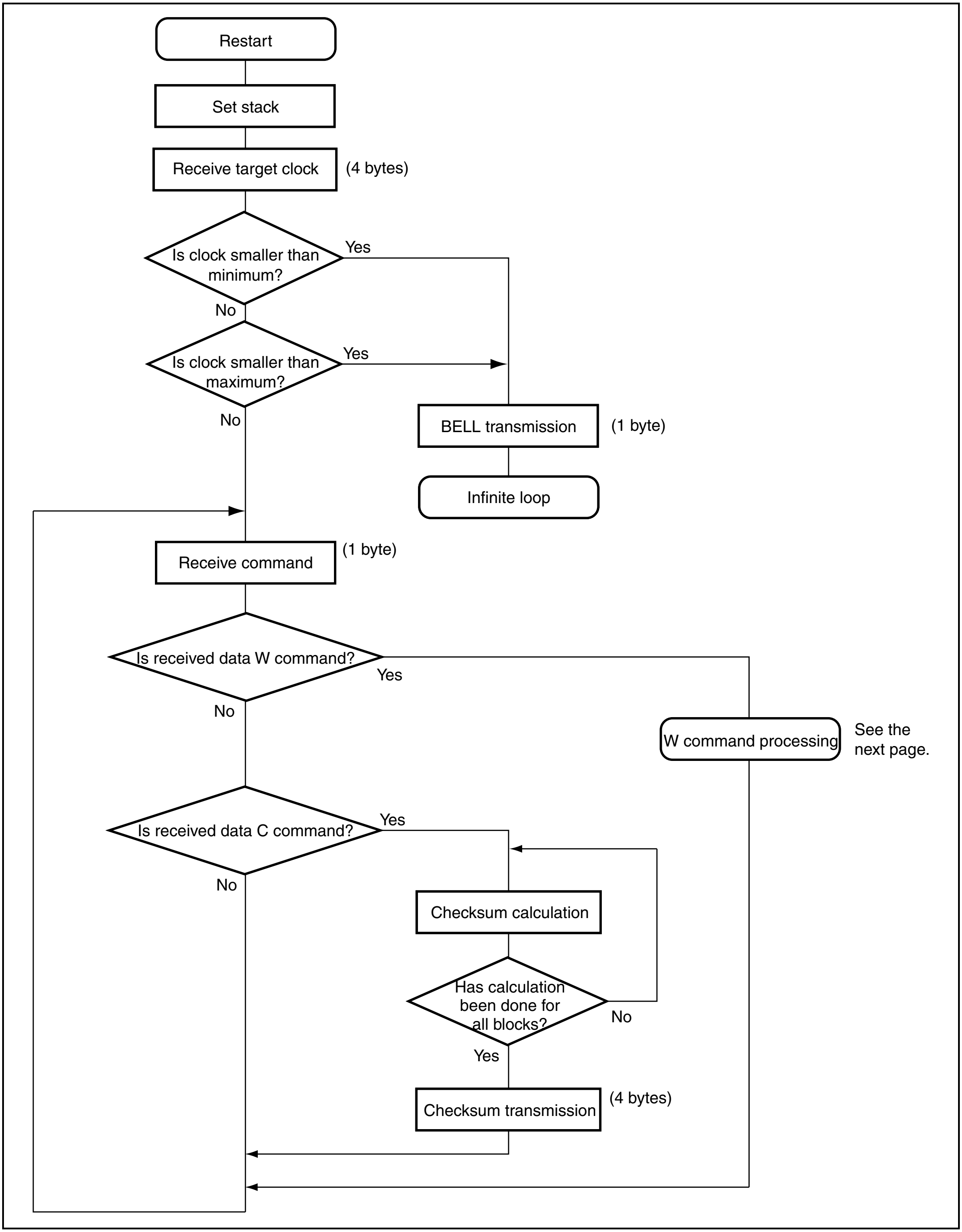
Program list (3)

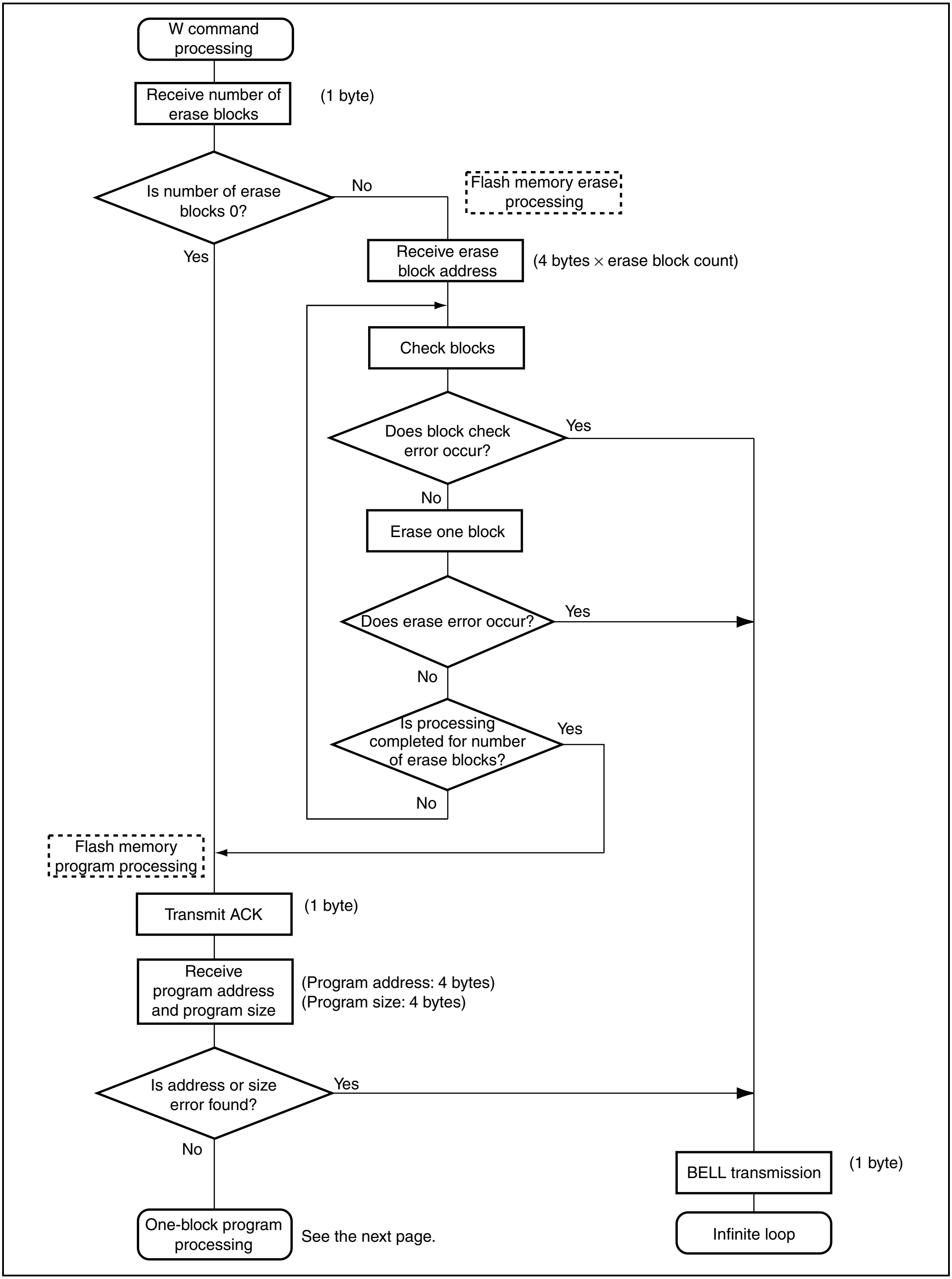
Source name: HCAN2612f3.src

Module name: TRS1BYTE

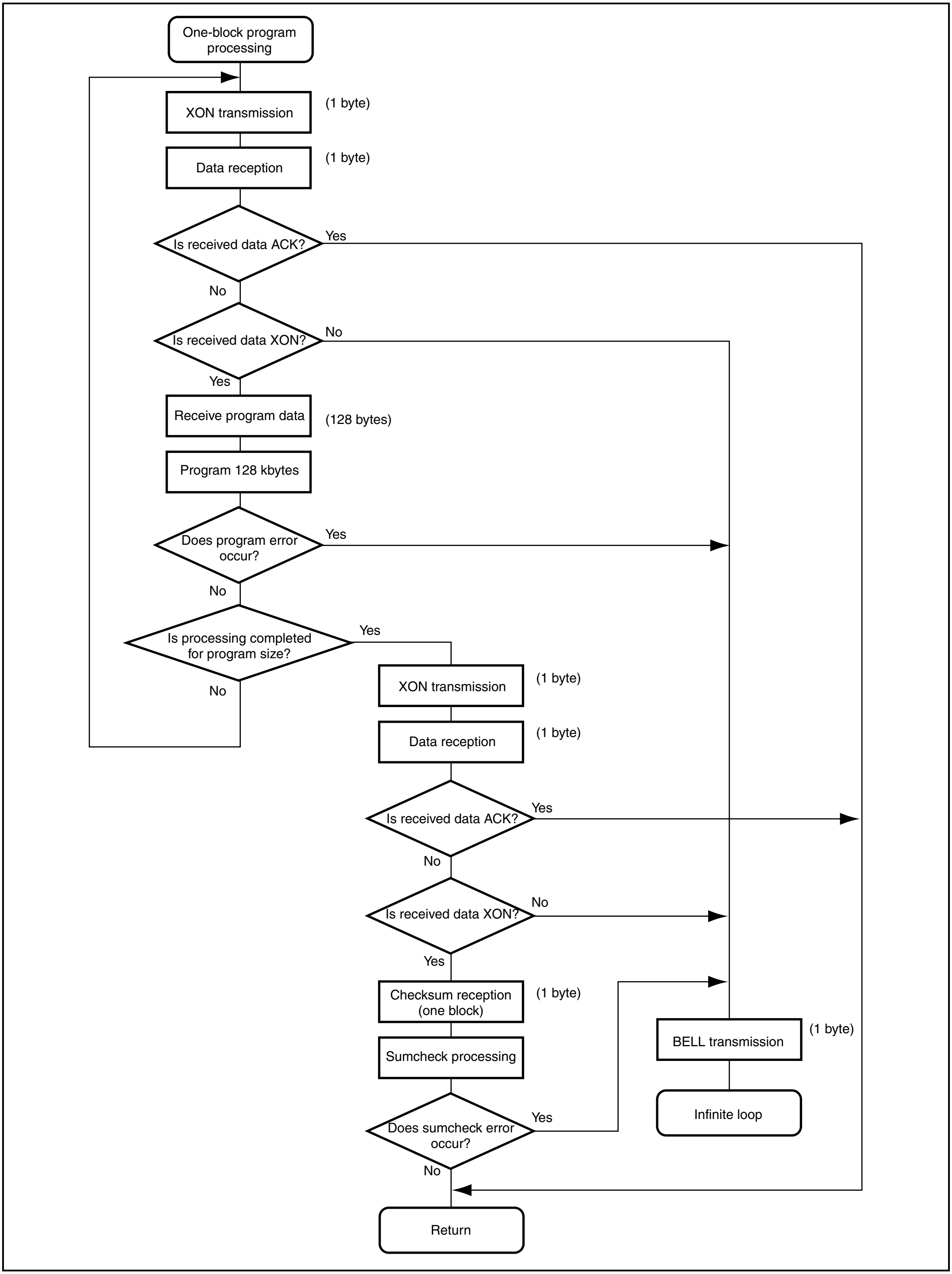
```
;*********************************************************************************************
; * TITLE    / H-CAN 1 BYTE DATA TRANSMISSION                                                *
; * FUNCTION / SEND 1 BYTE DATA                                                              *
; * INPUT    / R2L = SEND DATA                                                               *
; * OUTPUT   / -                                                                             *
;*********************************************************************************************
TRS1BYTE.    EQU      $
(14)         MOV.B    R2L,@MD5_0 →@MD15_0        ; TRANSMIT R2L DATA TO MD5_0
;
(15)         MOV.W    @TXPR →@TXPR_L,R0
(16)         BSET.B   #TXPR5 →#TXPR15,R0H
(17)         MOV.W    R0,@TXPR →@TXPR_L          ; SET TXPR5
;
TRS_WAIT     SUB.W    R0,R0
(18)         BLD.B    #TXPR5, →#TXPR15 @TXPR
(19)         BST.B    #TXPR5, →#TXPR15 R0H
             MOV.W    R0,R0
             BNE      TRS_WAIT
;
             MOV.W    @IRR,R0                    ; ERROR CHECK
             AND.W    #IRR_ERR,R0
             BNE      TRS_ERR
;
(20)         MOV.W    #TXACK5_MOV →#TXACK15_MOV,R0
             MOV.W    R0,@TXACK                  ; CLEAR TXACK5
             RTS
;
TRS_ERR      BRA  TRS_ERR                        ; INFINITE LOOP
;
```

7.4 Program/Erase Program Flowchart





W command processing
Receive number of erase blocks
(1 byte)
Is number of erase blocks 0?
No
Yes
Flash memory erase processing
Receive erase block address
(4 bytes × erase block count)
Check blocks
Does block check error occur?
Yes
No
Erase one block
Does erase error occur?
Yes
No
Is processing completed for number of erase blocks?
Yes
No
Flash memory program processing
Transmit ACK
(1 byte)
Receive program address and program size
(Program address: 4 bytes)
(Program size: 4 bytes)
Is address or size error found?
Yes
No
One-block program processing
See the next page.
BELL transmission
(1 byte)
Infinite loop

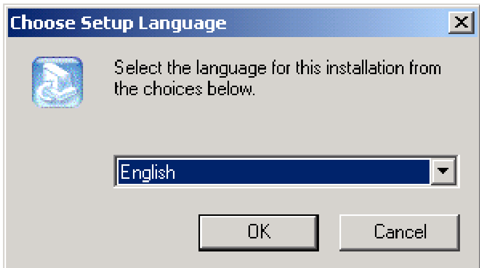

One-block program processing
XON transmission
(1 byte)
Data reception
(1 byte)
Is received data ACK?
Yes
No
Is received data XON?
No
Yes
Receive program data
(128 bytes)
Program 128 kbytes
Does program error occur?
Yes
No
Is processing completed for program size?
Yes
No
XON transmission
(1 byte)
Data reception
(1 byte)
Is received data ACK?
Yes
No
Is received data XON?
No
Yes
Checksum reception (one block)
(1 byte)
Sumcheck processing
Does sumcheck error occur?
Yes
No
BELL transmission
(1 byte)
Infinite loop
Return

Section 8 Functions and Operations of On-Board Programming Tool and SCI-HCAN Communication Conversion Program

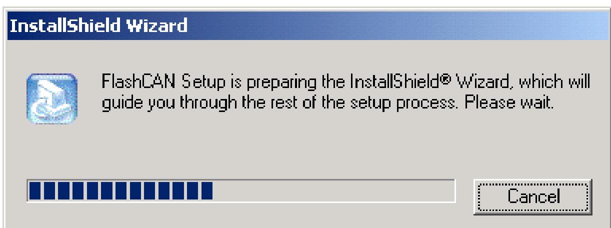
For details on the on-board programming tool, see the F-ZTAT Microcomputer On-Board Writing Program Manual.

8.1 Installing On-Board Programming Tool

1. Start Setup.exe.

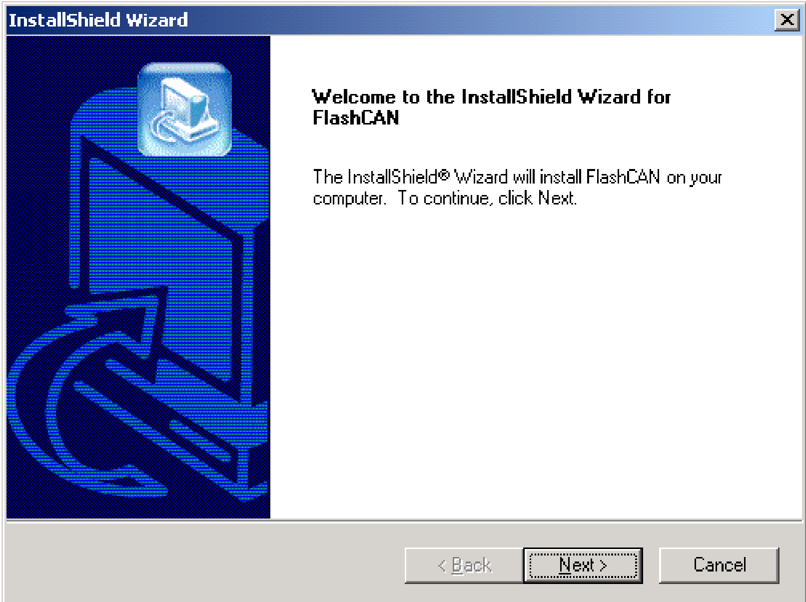


2. Select the language you want to use and click the OK button.
   (Click the Cancel button to cancel the installation of the tool.)



   (Click the Cancel button to cancel the installation of the tool.)

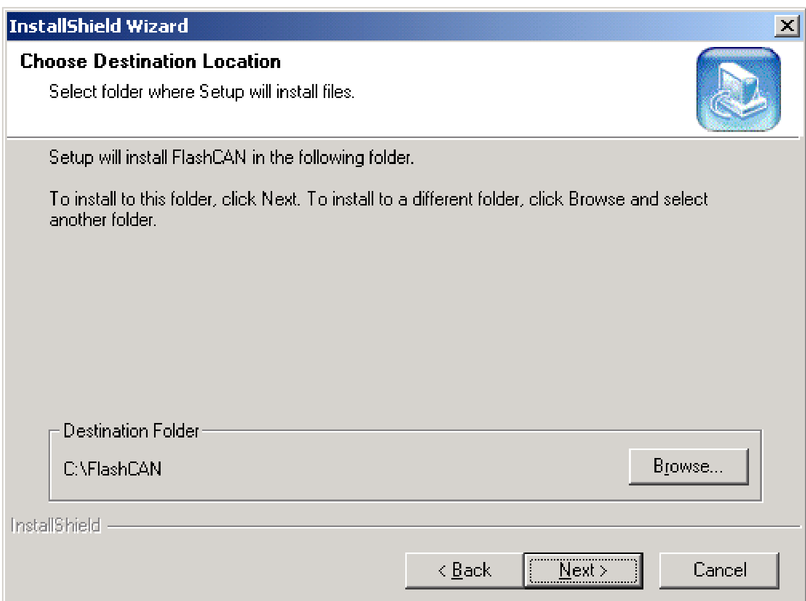
3. Click the Next button to continue the installation of the tool.



(Click the Cancel button to display the Confirm dialog box to cancel the installation of the tool.)

4. Select the destination folder where you want to install the tool.

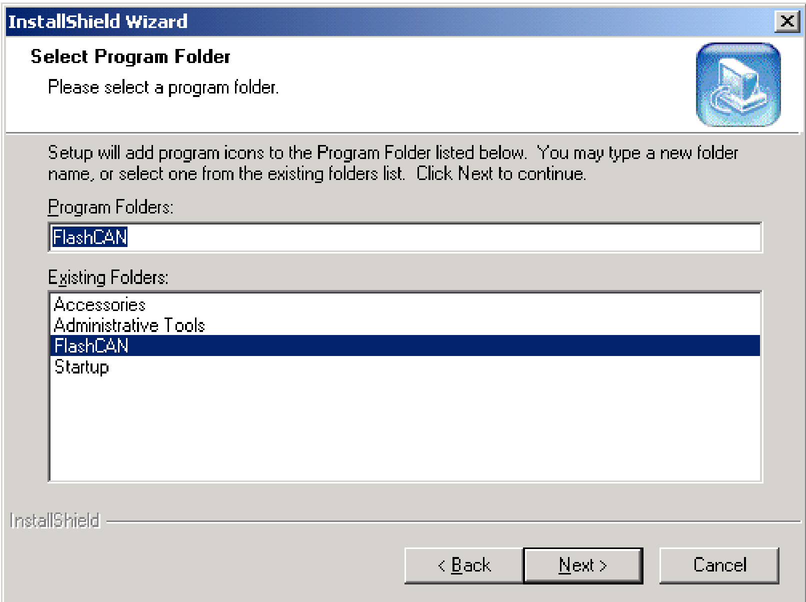
(To change the destination folder, click the Browse button to select another folder.)



5. Click the Next button.

   (Click the Back button to return to step 3.)

   (Click the Cancel button to display the Confirm dialog box to cancel the installation of the tool.)



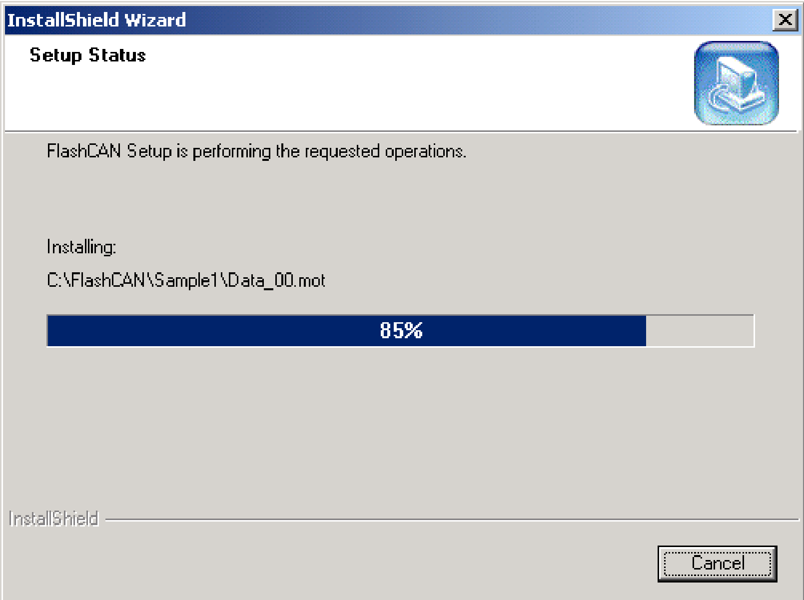
6. Select a program folder and click the Next button.

   (Click the Back button to return to step 3.)

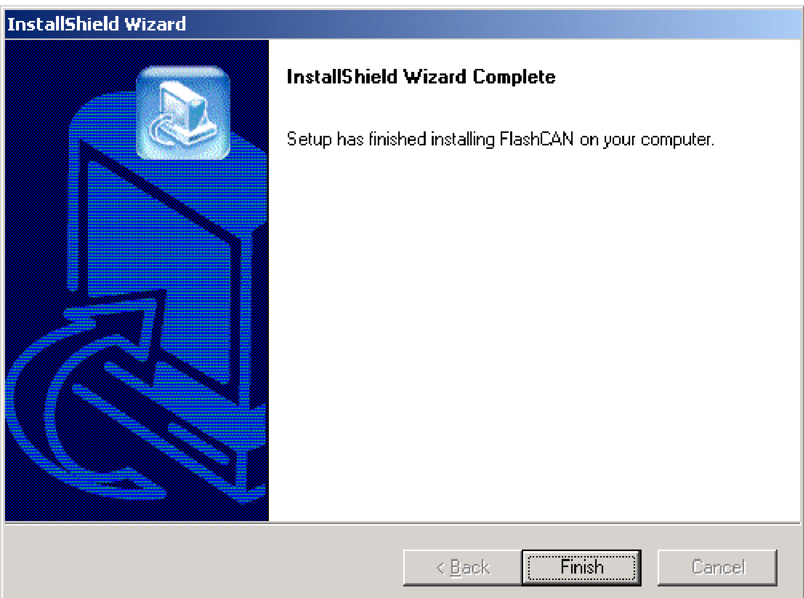
   (Click the Cancel button to display the Confirm dialog box to cancel the installation of the tool.)

7. The setup status of the tool appears.

   (Click the Cancel button to display the Confirm dialog box to cancel the installation of the tool.)



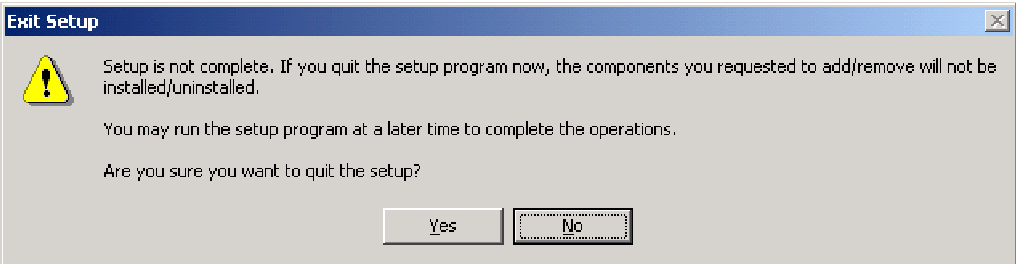
8. The wizard appears to indicate the completion of the installation. Click the Finish button.



Now, the on-board programming tool is installed completely.

Select the Start, Programs, and FlashCAN menu, and then the FlashCAN shortcut menu to start the on-board programming tool.

- Confirm dialog box to cancel the installation of the tool

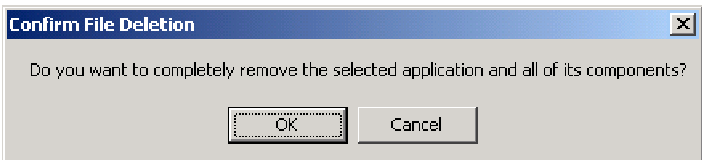


Click the Yes button to cancel the installation of the tool.

Click the No button to continue the installation.

- Confirm dialog box to uninstall the tool

Start Setup.exe to uninstall the installed tool.



Click the OK button and follow the wizard to uninstall the tool.

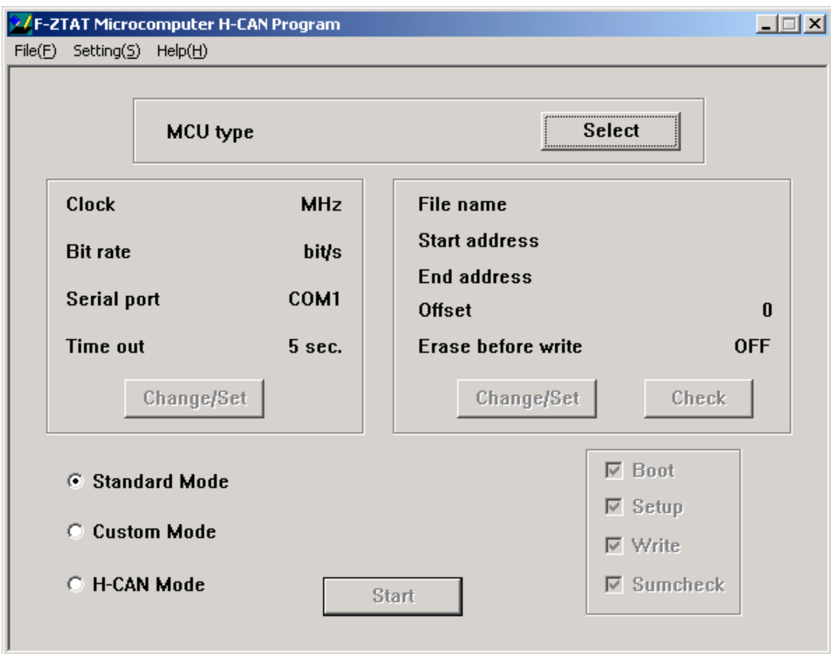
Click the Cancel button to cancel the uninstallation of the tool.

8.2 First Programming of Application (Sample) Program to Target Board

1. Use the serial cable to connect the personal computer to the target board. (In this application note, the serial cable is connected to COM1.)
2. Turn the target board on.
3. Place the target board in boot mode.
4. Start FlashCAN.exe.

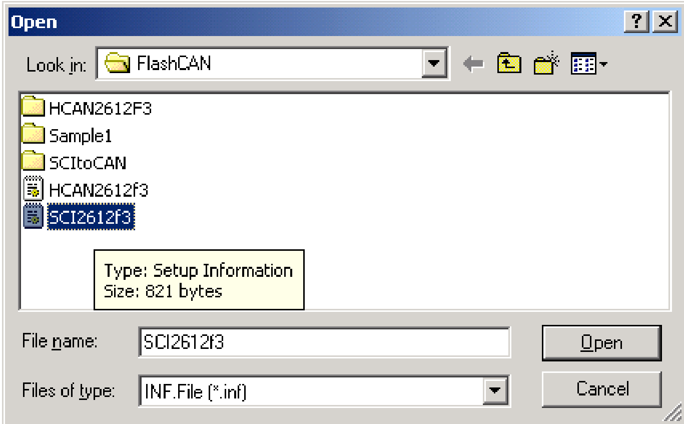
   The version information (for a working sample) appears.

5. Click the OK button to display the main window.

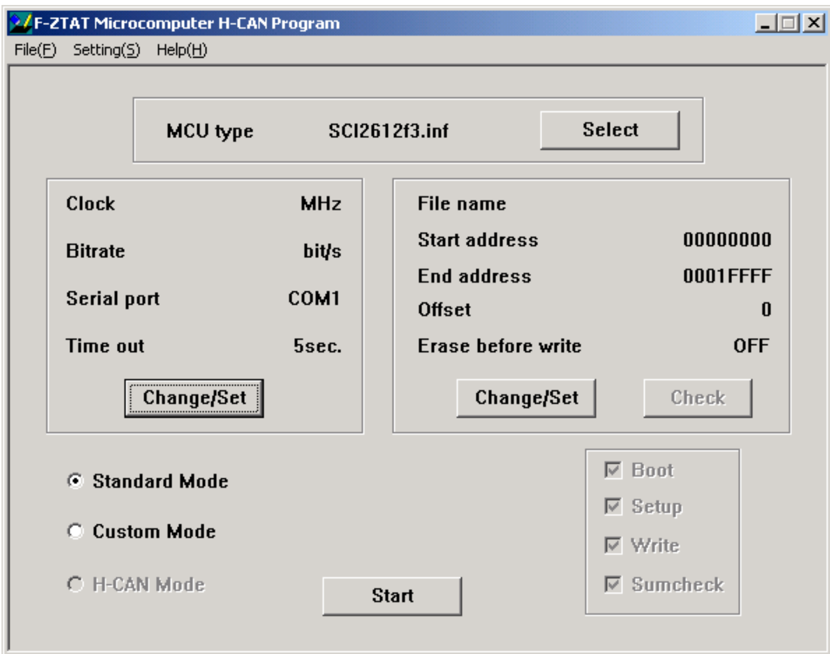


6. Click the Select button to select a microcomputer in the Select File dialog box.

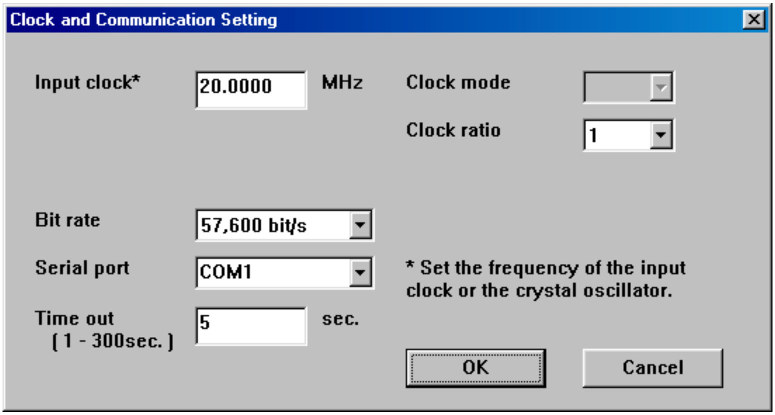
7. Select the SCI2612f3.inf file.



8. Click the Open button to return to the main window.
(Click the Cancel button to return to the main window without selecting a file.)



9. Click the Change/Set button for the clock and bit rate to make settings.



10. Set the input clock to 20.0000 MHz.

    The input clock can be set between 4 MHz and 20 MHz as specified in the SCI2612F3.inf file.

11. Select 1 for the clock ratio.

    The clock ratio can be selected from among 1, 2, and 4 as specified in the SCI2612F3.inf file.

12. Select 57600 bit/s (bit rate) for the serial communication.

    The bit rate can be selected from among 2400, 4800, 9600, 19200, 38400, 57600, 115200, and None.

13. Select COM1 for the serial port.

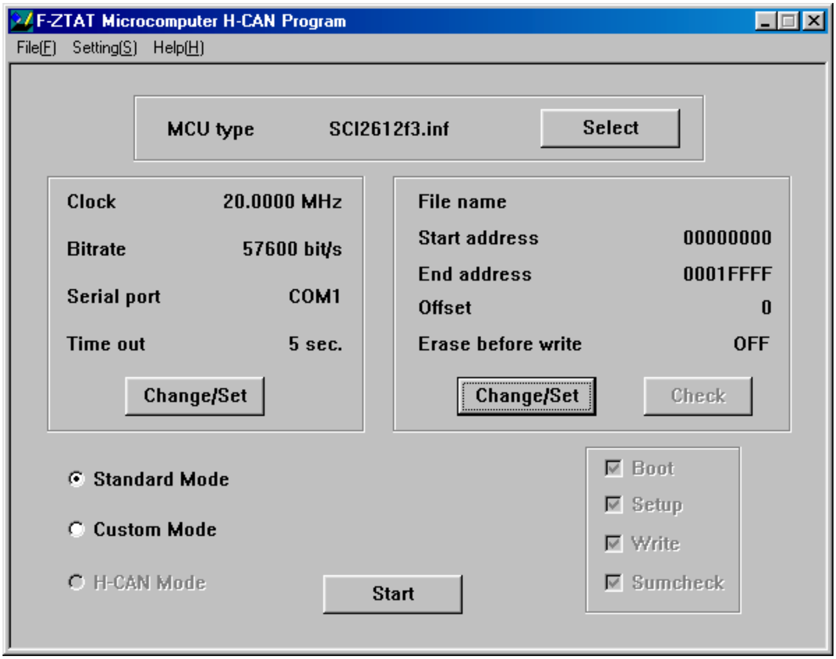
    Either COM1 or COM2 can be selected.

    Alternatively, you can directly enter any serial port name.

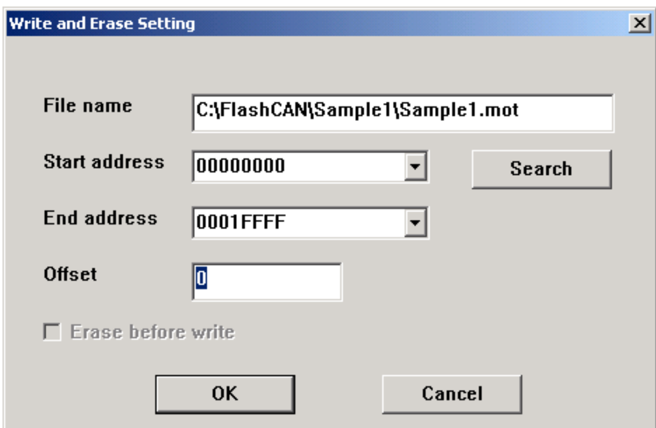
14. Set the timeout value to 5.

    The timeout value can be set between 1 and 300.

15. Click the OK button to return to the main window.
(Click the Cancel button to return to the main window without changing the settings.)



16. Click the Change/Set button for Erase before write to make settings.



RENESAS

17. Select the write data file Sample1.mot.

    Click the Search button to select a file in the Select File dialog box.

18. Select 0x00000000 for the start address.

    The start address can be selected from among 000000, 000400, 000800, 000C00, 001000, 008000, 00C000, 00E000, 010000, and 018000 as specified in the SCI2612F3.inf file.

    Alternatively, you can directly enter the start address between 000000 and 01FFFE.

19. Select 0x0001FFFF for the end address.

    The end address can be selected from among 0003FF, 0007FF, 000BFF, 000FFF, 007FFF, 00BFFF, 00DFFF, 00FFFF, 017FFF, and 018FFF as specified in the SCI2612F3.inf file.

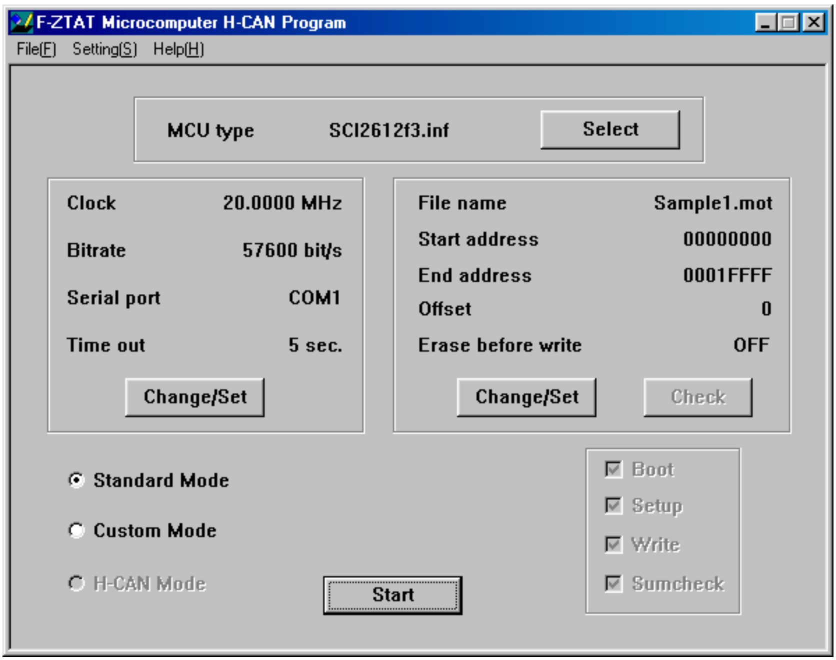
    Alternatively, you can directly enter the end address between 000001 and 01FFFF.

20. Set the offset to 0x00000000.

    The offset can be set between 000000 and 01FFFF as specified in the SCI2612F3.inf file.

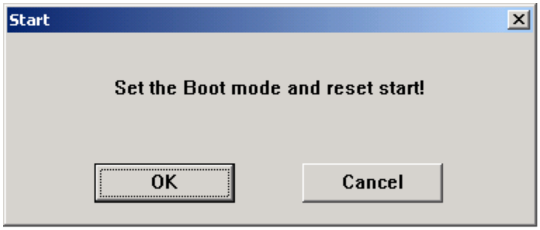
21. Click the OK button to return to the main window.

    (Click the Cancel button to return to the main window without changing the settings.)

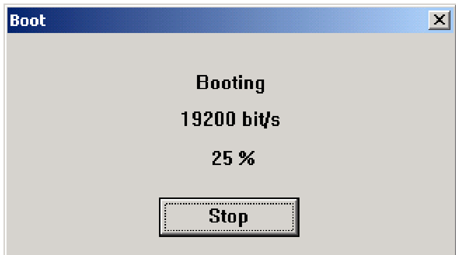


22. Now, all the settings are made completely. Click the Start button.

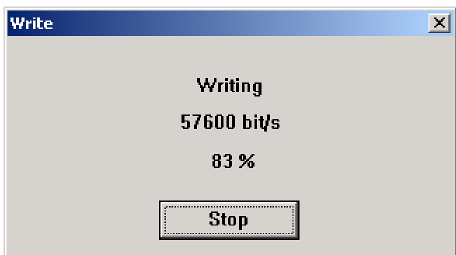
23. The Start dialog box appears.
Restart the target board.



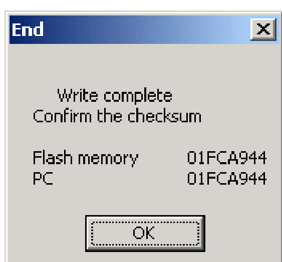
24. Click the OK button to start processing.
(Click the Cancel button to return to the main window.)



25. The status of the boot processing appears.
(Click the Stop button to stop the boot processing and return to the main window.)



26. The status of the write processing appears.
(Click the Stop button to stop the write processing and return to the main window.)
27. The processing completes and the checksum appears.



RENESAS

28. Confirm the checksum and click the OK button to return to the main window.

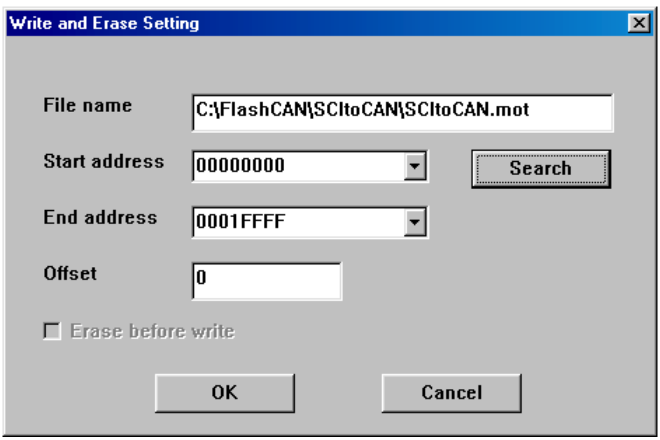
Now, the application (sample) program is completely programmed into the target board for the first time.

Restarting the target board initiates the application (sample) program.

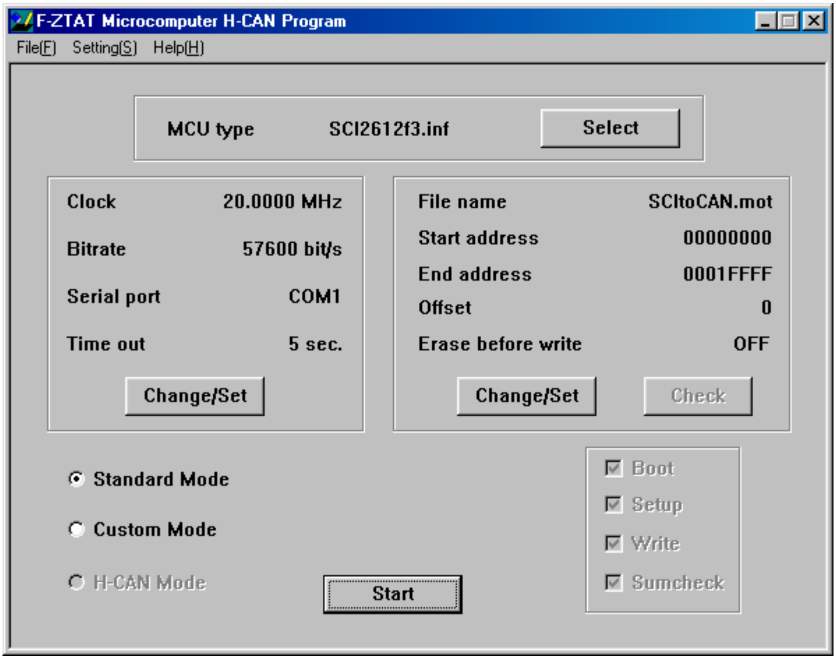
Since the parameter is set to 0xFF, all the LEDs light up.

8.3 Programming SCI-HCAN Communication Conversion Program to SCI-HCAN Communication Conversion Board

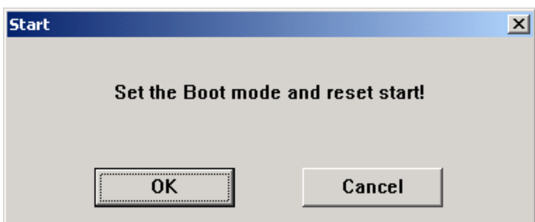
1. Use the serial cable to connect the personal computer to the SCI-HCAN communication conversion board.
   (In this application note, the serial cable is connected to COM1.)
2. Turn the SCI-HCAN communication conversion board on.
3. Place the SCI-HCAN communication conversion board in boot mode.
4. Follow steps 4 to 16 in section 8.2, First Programming of Application (Sample) Program to Target Board.
5. Select the write data file SCItoCAN.mot in step 17 in section 8.2.



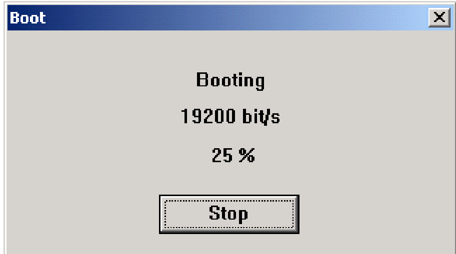
6. Follow steps 18 to 21 in section 8.2.



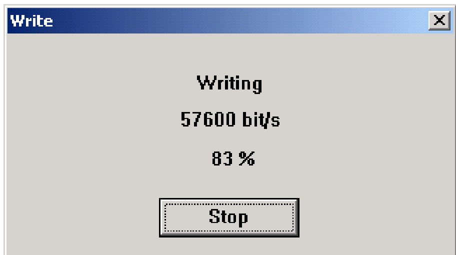
7. Now, all the settings are made completely. Click the Start button.
8. The Start dialog box appears.
   Restart the SCI-HCAN communication conversion board.



9. Click the OK button to start processing.
   (Click the Cancel button to return to the main window.)



10. The status of the boot processing appears.
    (Click the Stop button to stop the boot processing and return to the main window.)



11. The status of the write processing appears.
    (Click the Stop button to stop the write processing and return to the main window.)
12. The processing completes and the checksum appears.

13. Confirm the checksum and click the OK button to return to the main window.

Now, the SCI-HCAN communication conversion program is completely programmed into the SCI-HCAN communication conversion board.

Restarting the SCI-HCAN communication conversion board initiates the SCI-HCAN communication conversion program.

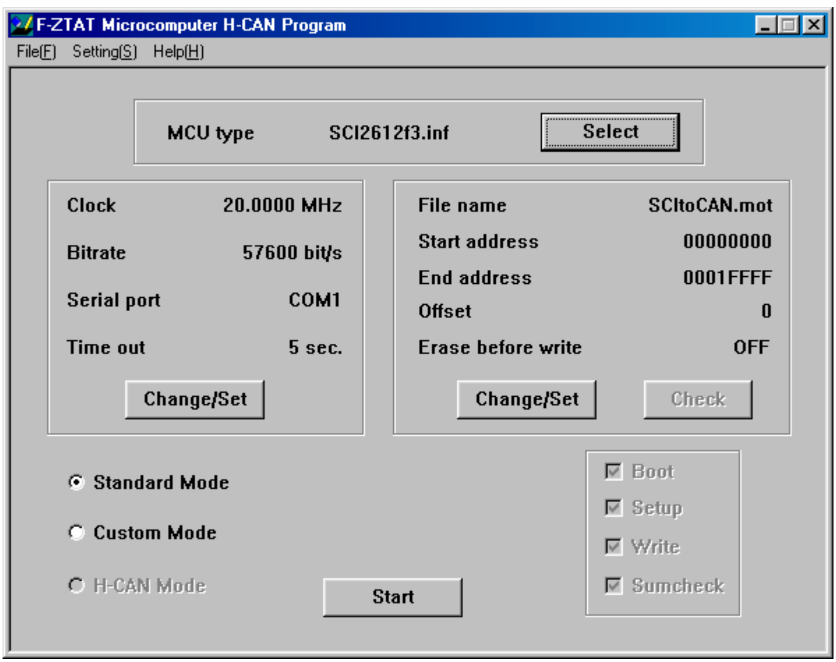
8.4 Reprogramming Flash Memory in User Program Mode

1. Use the serial cable to connect the personal computer to the SCI-HCAN communication conversion board.

   (In this application note, the serial cable is connected to COM1.)
2. Use the CAN cable to connect the target board to the SCI-HCAN communication conversion board.
3. Turn the SCI-HCAN communication conversion board on.
4. Turn the target board on.
5. Place the SCI-HCAN communication conversion board in user mode.
6. Place the target board in user mode (by resetting the target board).
7. Start FlashCAN.exe.

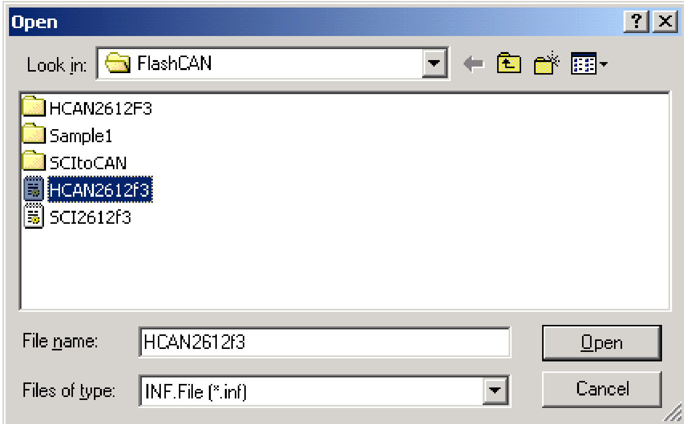
   The version information (for a working sample) appears.

8. Click the OK button to display the main window.

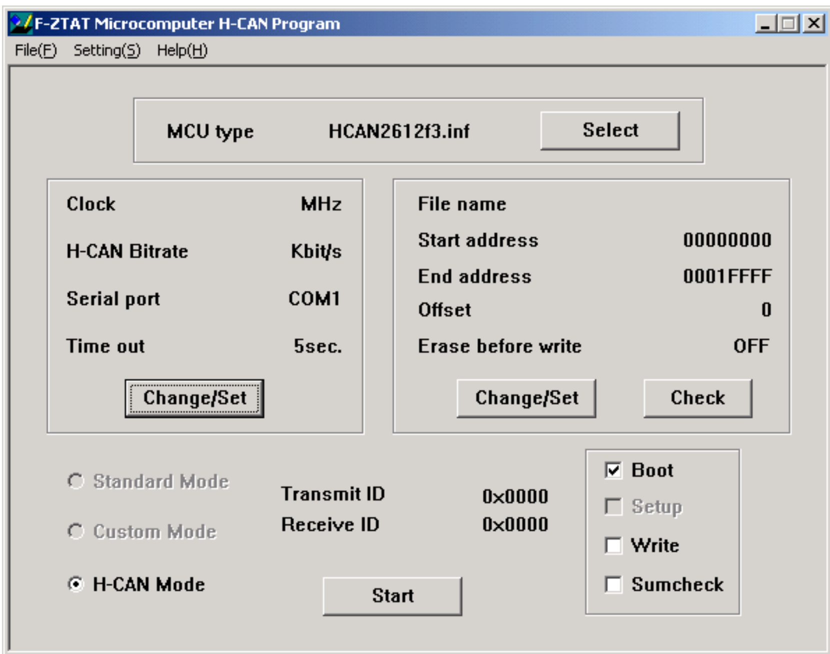


The preset contents appear in the main window.

9. Click the Select button to select a microcomputer in the Select File dialog box.
10. Select the HCAN2612f3.inf file.



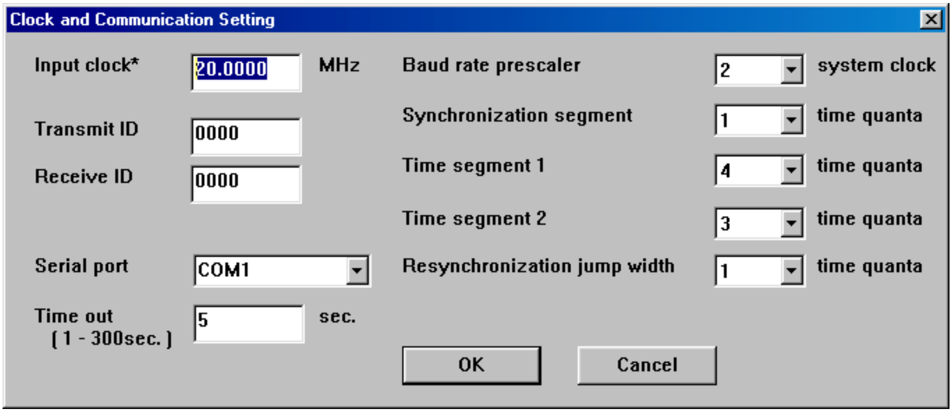
11. Click the Open button to return to the main window.
(Click the Cancel button to return to the main window without selecting a file.)



12. Click the Change/Set button for the clock and bit rate to make settings.

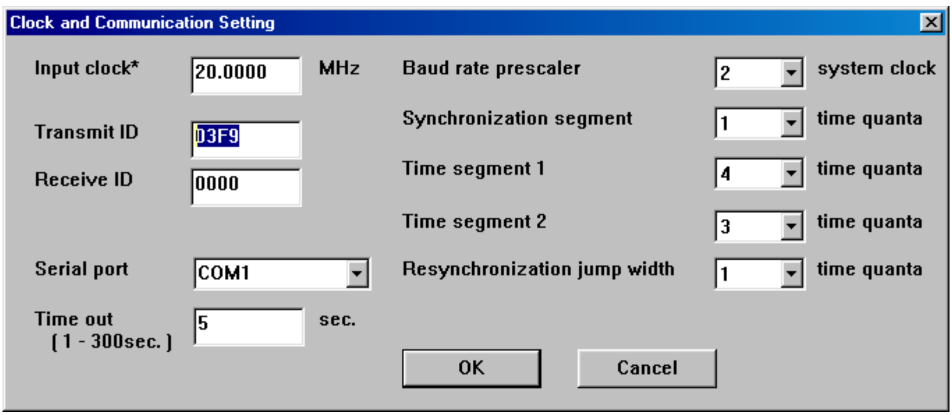
13. Set the input clock of the target board to 20.0000 MHz.

The input clock can be set between 4 MHz and 20 MHz as specified in the HCAN2612F3.inf file.



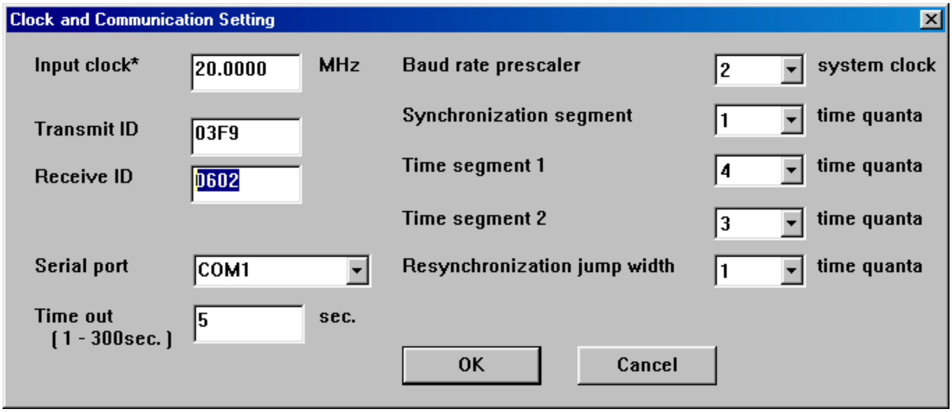
14. Set the transmit mailbox ID to 0x03F9.

The transmit mailbox ID can be set between 0000 and 07EF.



15. Set the receive mailbox ID to 0x0602.

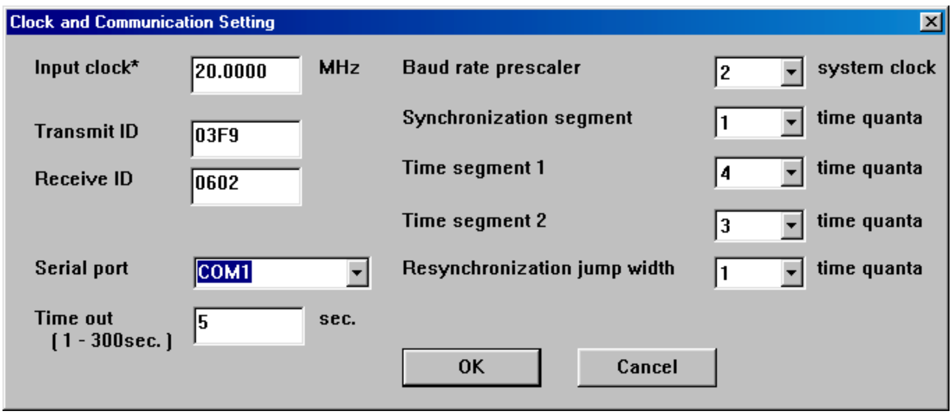
The receive mailbox ID can be set between 0000 and 07EF.



16. Select COM1 for the serial port.

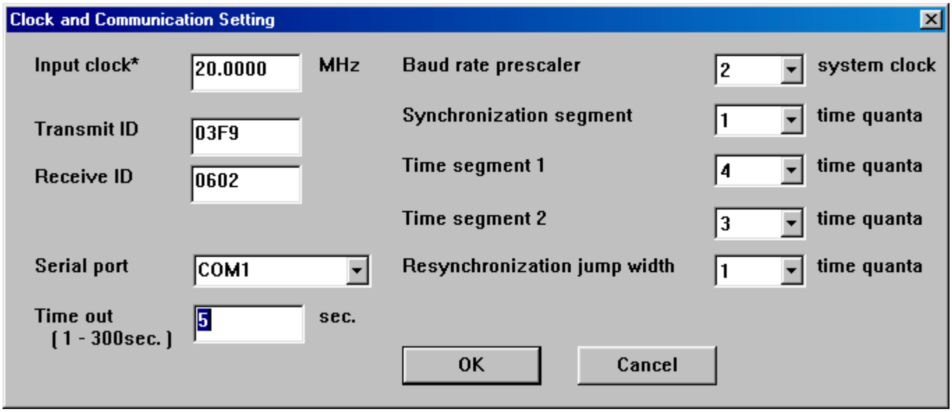
Either COM1 or COM2 can be selected.

Alternatively, you can directly enter any serial port name.



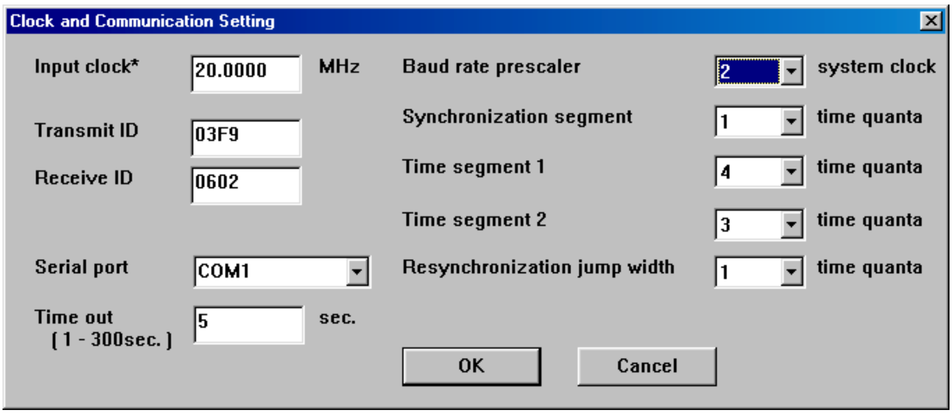
17. Set the timeout value to 5.

The timeout value can be set between 1 and 300.



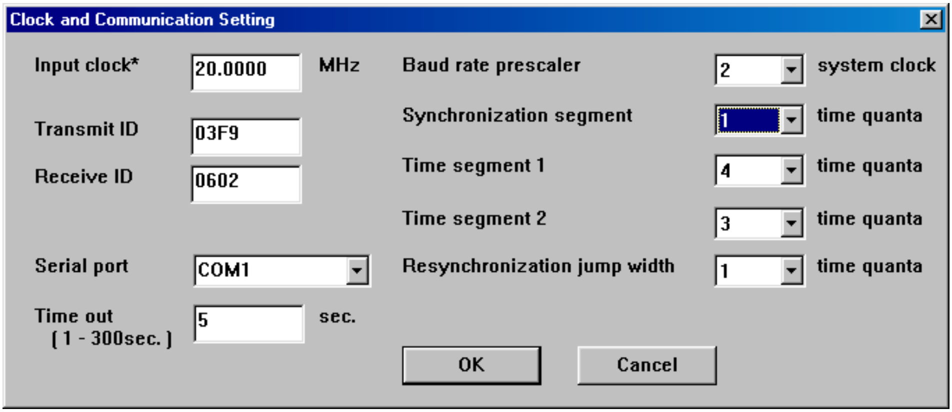
18. Select 2 (system clock) for the baud rate prescalor.

The value can be set between 2 and 128.



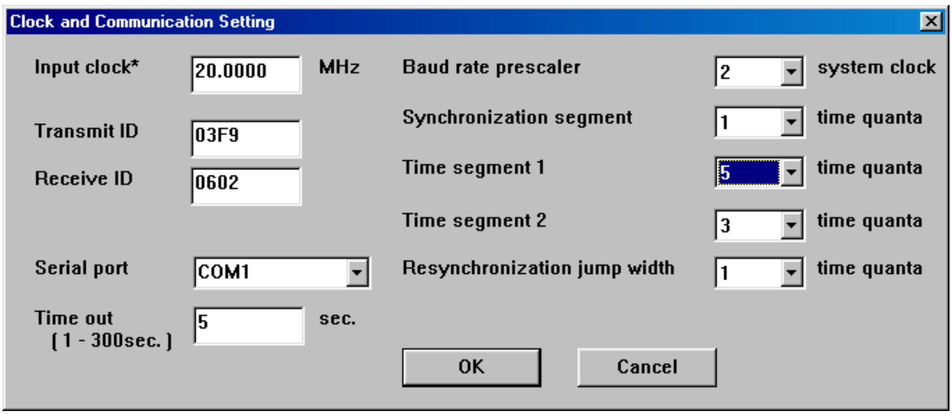
19. Select 1 (time quanta) for the synchronization segment.

Only 1 can be selected for this value.



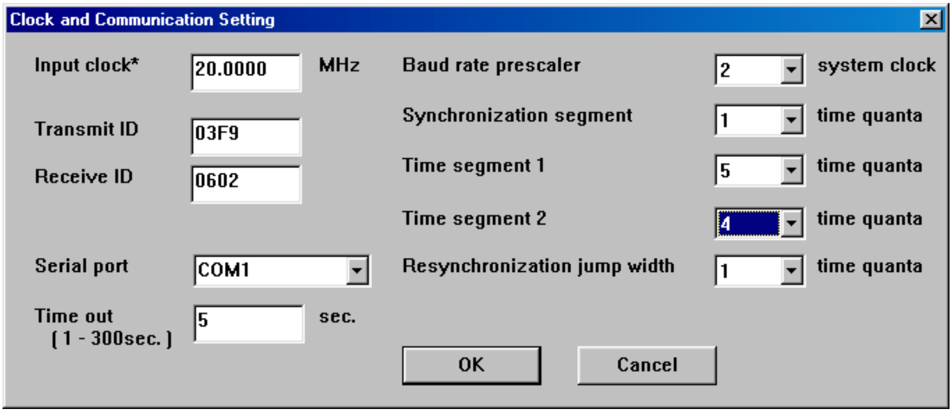
20. Select 5 (time quanta) for time segment 1.

The value can be set between 4 and 16.



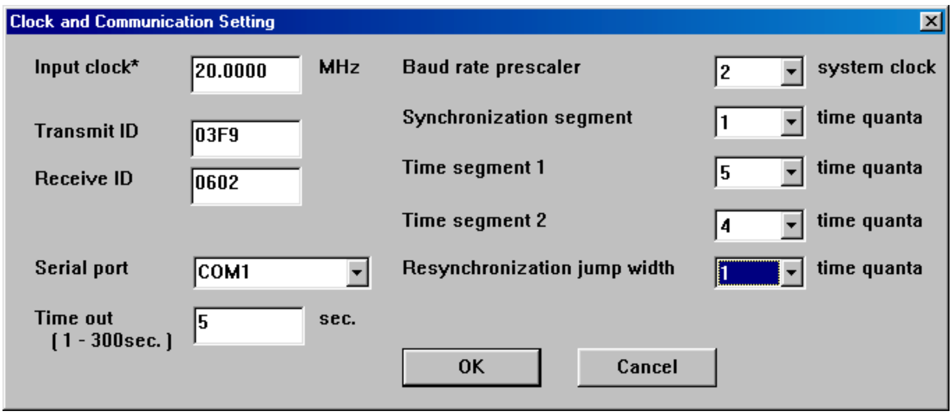
21. Select 4 (time quanta) for time segment 2.

The value can be set between 3 and 8.

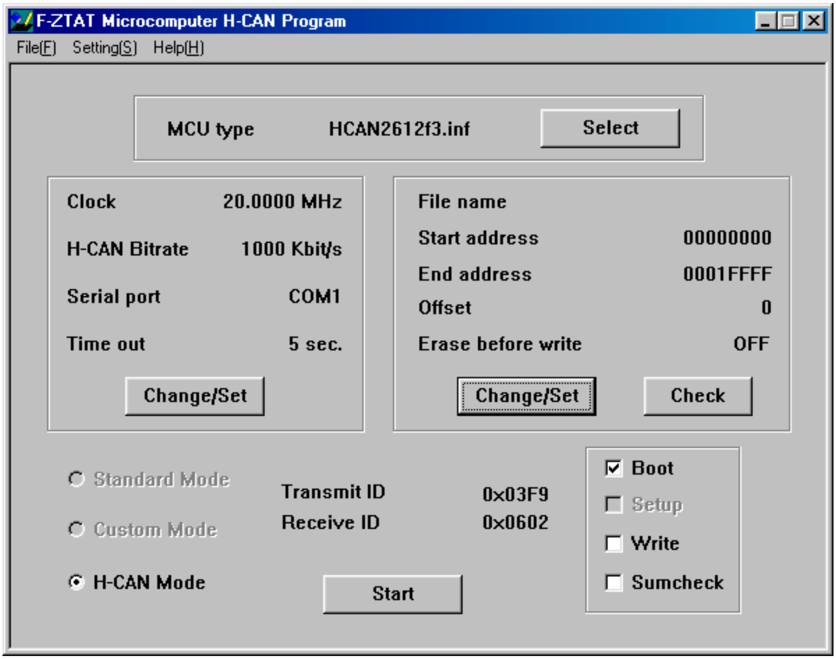


22. Select 1 (time quanta) for the resynchronization jump width.

The value can be set between 1 and 4.

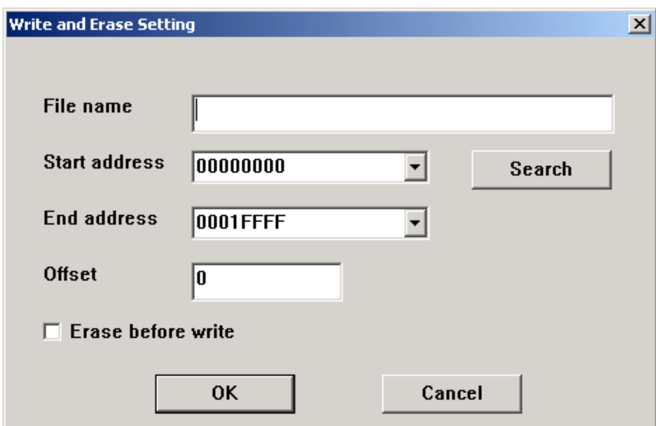


23. Click the OK button to return to the main window.
(Click the Cancel button to return to the main window without changing the settings.)

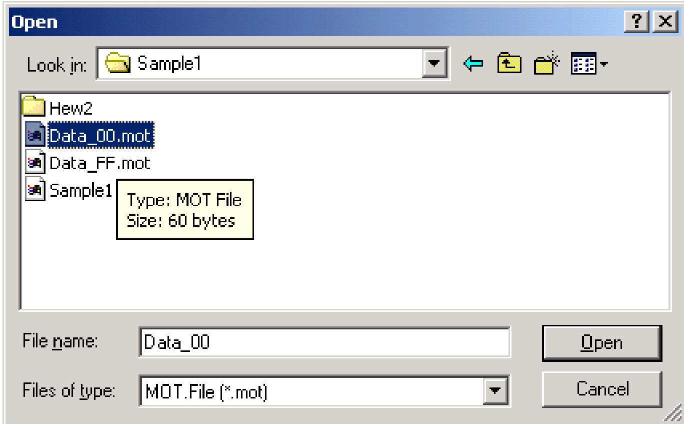


The bit rate for the HCAN communication appears. This value is based on the settings of the input clock, serial port, timeout, transmit mailbox ID, and receive mailbox ID.

24. Click the Change/Set button for erase before write to make settings.
25. Click the Search button to select a write data file.

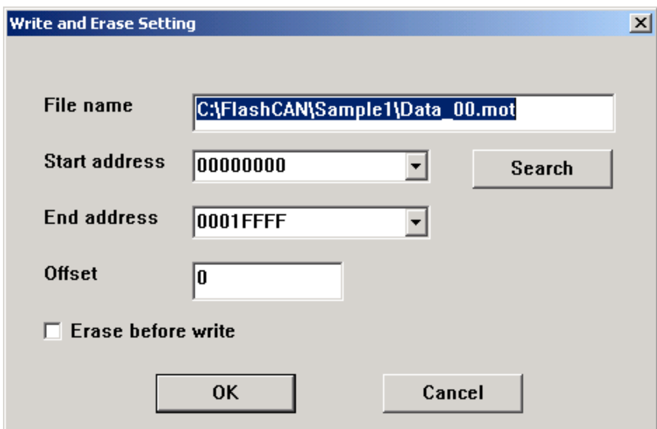


26. Select the write data file Data_00.mot.



27. Click the Open button to return to the Write and Erase Setting dialog box.

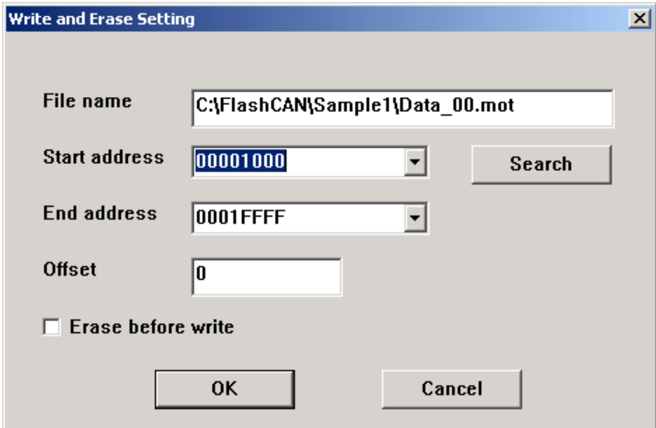
(Click the Cancel button to return to the Write and Erase Setting dialog box without selecting a file.)



28. Select 0x00001000 for the start address.

The start address can be selected from among 000000, 000400, 000800, 000C00, 001000, 008000, 00C000, 00E000, 010000, and 018000 as specified in the HCAN2612F3.inf file.

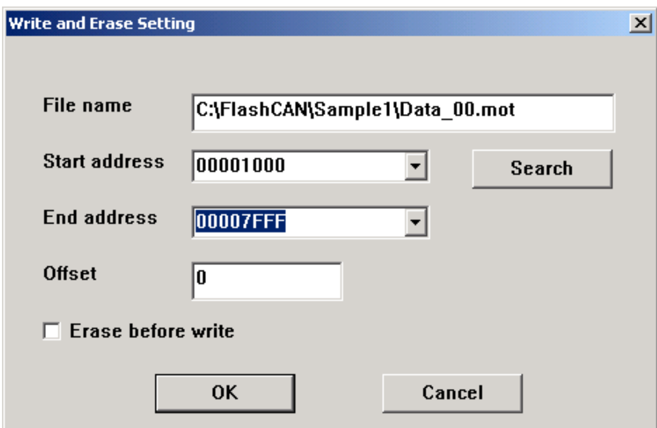
Alternatively, you can directly enter the start address between 000000 and 01FFFE.



29. Select 0x00007FFF for the end address.

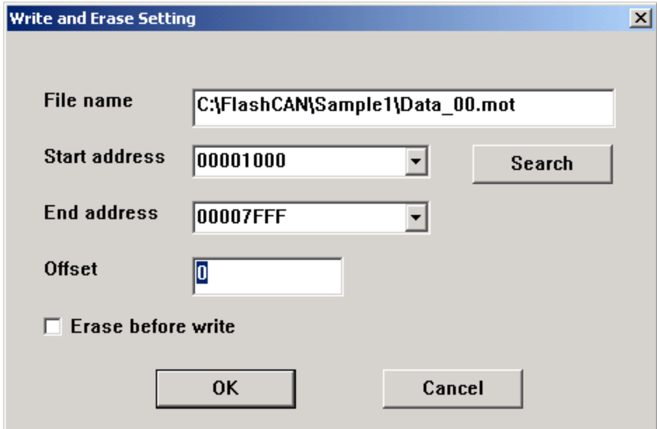
The end address can be selected from among 0003FF, 0007FF, 000BFF, 000FFF, 007FFF, 00BFFF, 00DFFF, 00FFFF, 017FFF, and 018FFF as specified in the HCAN2612F3.inf file.

Alternatively, you can directly enter the end address between 000001 and 01FFFF.



30. Set the offset to 0x00000000.

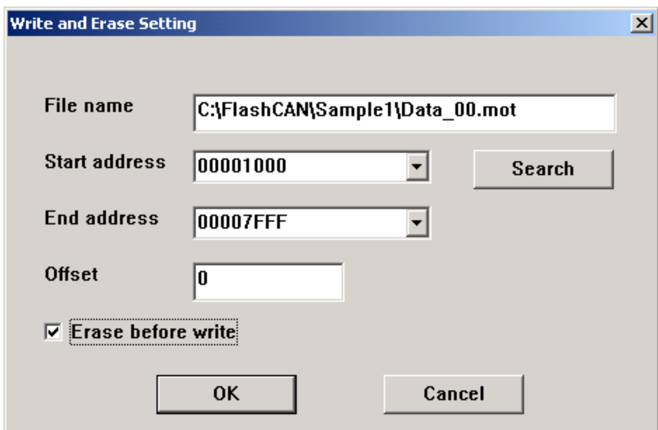
The offset can be set between 000000 and 01FFFF as specified in the HCAN2612F3.inf file.



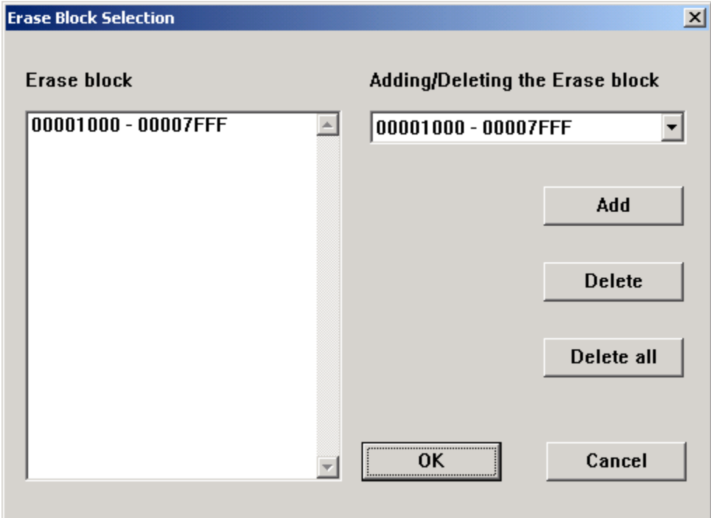
31. Place a checkmark (on) in the Erase before write checkbox.

Place a checkmark (on) in the checkbox to erase blocks before the write processing.

Uncheck the checkbox (off) not to erase blocks before the write processing.



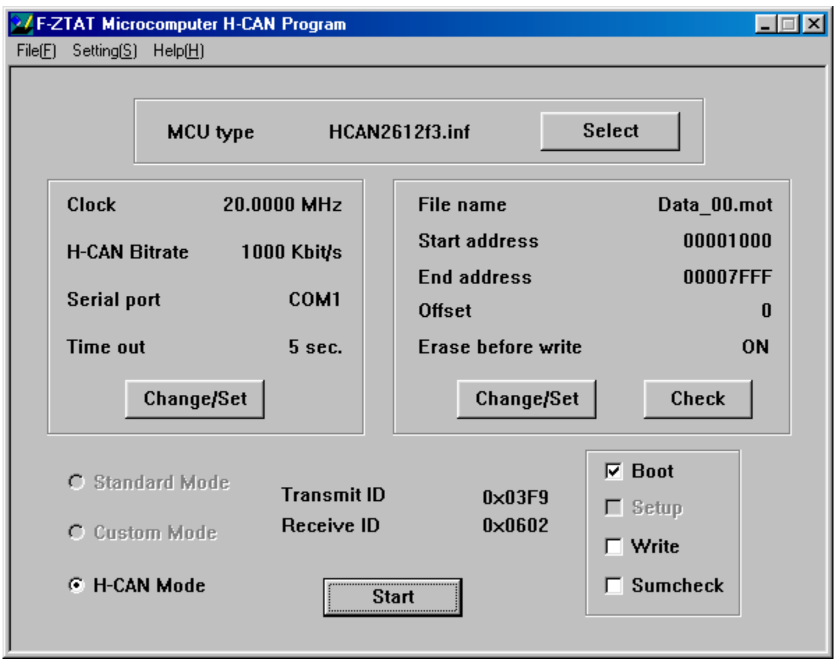
32. Click the OK button to select the blocks to be erased in the Erase Block Selection dialog box.



The blocks between the selected start address and the end address are set for erasure.

33. Click the OK button to return to the main window.

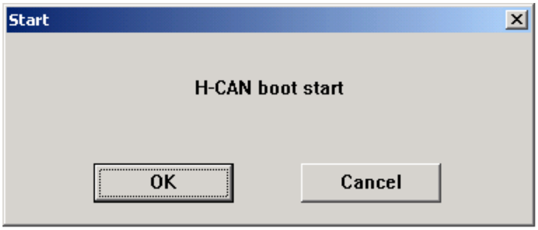
(Click the Cancel button to return to the Write and Erase Setting dialog box.)



34. Now, all the settings are made completely. Click the Start button.

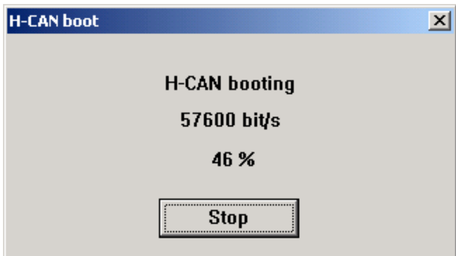
35. The Start dialog box appears.

Restart the SCI-HCAN communication conversion board.



36. Click the OK button to start processing.

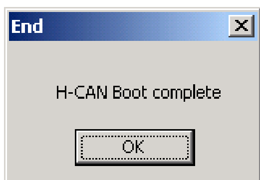
(Click the Cancel button to return to the main window.)



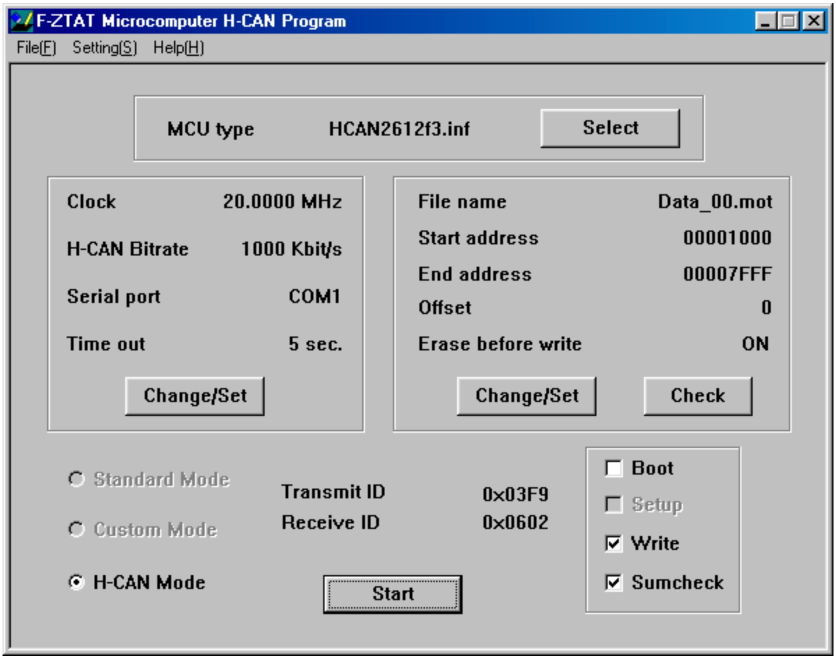
37. The status of the boot processing appears.

(Click the Stop button to stop the boot processing and return to the main window.)

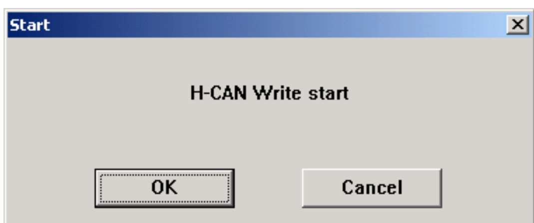
38. The boot processing completes and then the completion message appears.



39. Click the OK button to return to the main window.

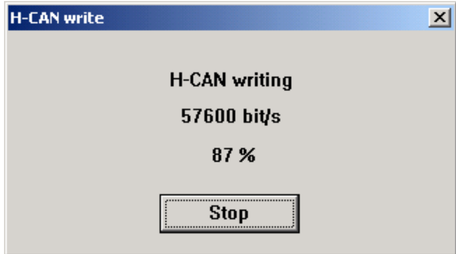


40. Uncheck the Boot checkbox (off).
    Place a checkmark in the Write checkbox (on).
    Place a checkmark in the Sumcheck checkbox (on).
41. Place the target board in user program mode.
42. Now, all the settings are made completely for writing. Click the Start button.
43. The Start dialog box appears.



RENESAS

44. Click the OK button to start processing.
(Click the Cancel button to return to the main window.)



45. The status of the write processing appears.
(Click the Stop button to stop the write processing and return to the main window.)
46. The processing completes and the checksum appears.

47. Confirm the checksum and click the OK button to return to the main window.

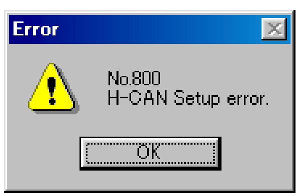
Now, the flash memory is completely reprogrammed in user program mode.

Restarting the target board initiates the application (sample) program.

Since the parameter is set to 0x00, all the LEDs blink.

8.5 Error Messages for FlashCAN.exe (Additional Messages for HCAN)

1. Error dialog box
The Error dialog box appears if an error occurs.
Check an error message, and then click the OK button.



2. List of error messages

No. 319	Information file: Incorrect specification of H-CAN.
Description	An incorrect HCAN is specified in the information (INF) file.
No. 650	Transmit-ID was illegally specified.
Description	An illegal transmit ID (transmit mailbox ID) is specified.
No. 651	Receive-ID was illegally specified.
Description	An illegal receive ID (receive mailbox ID) is specified.
No. 652	Transmit-ID/Receive-ID was illegally specified.
Description	The transmit ID (transmit mailbox ID) and receive ID (receive mailbox ID) are identical.
No. 653	TSEG2 was illegally specified.
Description	An illegal TSEG2 (time segment 2) is specified.
No. 654	TSEG1 was illegally specified.
Description	An illegal TSEG1 (time segment 1) is specified.
No. 800	H-CAN Setup error.
Description	An error is detected during communication of the H command. The host side received NAK (0x07) and this caused the HCAN setup error.
No. 801	H-CAN Setup error. (confirmation error)
Description	An error is detected during communication of the H command. The host side received a signal other than ACK (0x06) and NAK (0x07) and this caused the HCAN setup error.
No. 802	H-CAN Setup error. (time out)
Description	An error is detected during communication of the H command. The host side could not receive any signal and this caused the timeout. The HCAN setup failed.
No. 803	H-CAN Send clock error.
Description	An error is detected during transmission with the HCAN frequency. The host side received NAK (0x07) and this caused the transmission error.
No. 804	H-CAN Send clock error. (confirmation error)
Description	An error is detected during transmission with the HCAN frequency. The host side received a signal other than ACK (0x06) and NAK (0x07) and this caused the transmission error.
No. 805	H-CAN Send clock error. (time out)
Description	An error is detected during transmission with the HCAN frequency. The host side could not receive any signal and this caused the timeout. The transmission failed.

Section 9 Supplementary Description

9.1 Required Items for Reprogramming Flash Memory in User Program Mode

To reprogram the flash memory in user program mode, the user must provide the following methods. The shaded section in the table indicates the method used in this application note.

1. Items required for user board

No.	User Must Provide:	Method Examples
1	Method for programming in boot mode: This method switches user mode to boot mode, and vice versa. It also provides program data from the SCI_2.	Mode switch and SCI_2 connector
2	Method for switching the FWE pin by hardware: This method switches user mode to user program mode, and vice versa.	Mode switch

Note: Do not always apply a high level voltage to the FWE pin. Be sure to switch the FWE pin when the CPU is not accessing the flash memory.

2. Items that application must include

No.	User Must Provide:	Method Examples
1	Method for transition to the flash-memory reprogram processing: This method accepts a trigger for starting the flash-memory reprogram processing, and then jumps to the reprogram processing.	FWE pin level sense, external interrupt, command reception by SCI, or command reception by HCAN, etc.
2	Method for transferring the program/erase control program to RAM: A program in RAM must control the programming and erasing of the flash memory. To satisfy this condition, this method transfers the program/erase control program to RAM, and then jumps to the transferred program.	Transfer from ROM to RAM, transfer from outside using SCI, or transfer from outside using HCAN, etc.

3. Items required by host

No.	User Must Provide:	Method Examples
1	Program/erase control program: This program is a Motorola-type load module. Use the program/erase algorithm that is recommended by the hardware manual. The user must also install the transfer function that matches the host to receive instruction or response for erase blocks, or the program data.	The program added to FDT.exe, added to FlashCAN.exe, or created by the user, etc.
2	Method for providing the program data (host): A host that controls the sequences shown in the above table and transfers the program data is necessary.	FDT.exe, FlashCAN.exe, or user-created tool, etc.

9.2 Differences between User Program Mode and Boot Mode

Two types of modes are available for on-board reprogramming of the flash memory: the user program mode and boot mode. The following shows the differences between these modes.

Item	User Program Mode	Boot Mode
Execution of application program	The application program downloaded in the flash memory is executed. Reprogramming the flash memory in user program mode requires the reprogram processing to be included in this application program in advance.	The application program downloaded in the flash memory is not executed. Instead, the on-chip boot program in the F-ZTAT microcomputer is executed.
Interface used for reprogramming of flash memory	The user can select the interface according to the user system. Examples of available interfaces are the SCI and HCAN.	Use the SCI. The F-ZTAT microcomputer automatically adjusts the communication rate. The protocol is fixed.
Transfer destination of program/erase control program	The whole RAM area from H′FFE000 to H′FFEFBF (4032 bytes) are available.	Transfer the program to RAM address from H′FFE800 to H′FFEFBF (1984 bytes).
Erase block	The user can select any block to erase. The user must provide the program that controls the erasing of the flash memory (erase control program) and transfer it to RAM.	The F-ZTAT microcomputer automatically erases all the blocks.
Programming	Reprogramming is possible in units of erase blocks. Data can be programmed only for the erased blocks. The user must provide the program that controls the programming of the flash memory (program control program) and transfer it to RAM.	Since all the blocks are erased, the user must program data to the entire flash memory. The user must provide the program that controls the programming of the flash memory (program control program) and transfer it to RAM.

Item	User Program Mode	Boot Mode
Method of mode transition	Two methods are available: Reset the system with the settings MD0 = 1, MD1 = 1, MD2 = 1, and FWE = 1; or During execution with MD0 = 1, MD1 = 1, MD2 = 1, and FWE = 0 in user mode, switch the setting to FWE = 1. This method enables reprogramming the flash memory without resetting the user system.	Reset the system with the settings MD0 = 1, MD1 = 1, MD2 = 0, and FWE = 1.

9.3 How to Measure Application Time for E and P Bits

Erasing and programming the flash memory is implemented by setting the E and P bits in FLMCR to apply a voltage. The hardware manual describes the voltage application time.

Setting too short application time for the E or P bit disables the erasing or programming. Setting too long application time causes excessive erasing or excessive programming, which leads to the permanent damage on the device. When setting the E and P bits, set up the on-chip watchdog timer to prevent a program runaway.

The sample program in this application note controls the application time for the E and P bits by adjusting the software loop count. Therefore, you need to increase or reduce the loop count according to the operating frequency. You can obtain the time required for the software loop by using calculation and simulator debugger. However, the application time for the E and P bits is so important that we recommend you actually measure these bits from outside.

To measure the E and P bits, output high/low voltage for the E and P bits to the on-chip I/O port. Output a signal to the on-chip I/O port at the same timing as setting the E and P bits to high and low. Use an oscilloscope or logic analyzer to measure the output signal of the on-chip I/O port from outside.

Source example: Output on/off of the P bit to the on-chip I/O port P00.

```
;======= WRITE pulse application =======================
       BSET.B #0,@PORT0 ; Set bit 0 in port 0 to measure P bit
       BSET.B #P,@ER6   ; Set P bit (programming)
FWRT40 DEC.L  #1,ER3    ; Program time: 10 µS, 30 µS, or 200 µS
       BNE    FWRT40:16
;=======================================================
    MOV.W     @WLOOP5,E0
    BCLR.B    #P,@ER6   ; Clear P bit
    BCLR.B    #0,@PORT0 ; Clear bit 0 in port 0 to measure P bit
```

F-ZTAT Reprogramming by On-Chip CAN
Application Note

Publication Date: 1st Edition, March 2003
Published by: Business Operation Division
Semiconductor & Integrated Circuits
Hitachi, Ltd.
Edited by: Technical Documentation Group
Hitachi Kodaira Semiconductor Co., Ltd.