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.

H8S Family

On-Board Reprogramming Example Using IEBus

Introduction

The programming data is received from the transfer source and written to flash memory. An IEBus interface is used for data transfer from the transfer source.

Target Device

H8S/2258F

Contents

1. Specifications

(1) On-Board Programming Mode

The user program mode is used.

(2) Reprogramming Method

The programming data is received from the transfer source and written to flash memory. An IEBus interface is used for data transfer from the transfer source. The data transfer format is shown below. The master is the transfer source and the slave is the receiving side.

Data transfer format: Communication mode 2 (128 bytes/frame), normal communication

(3) Control of the FWE Pin

This sample task illustrates an FWE pin circuit that can be built by the user without using an on-board programming tool from Renesas Technology Corp. The FWE pin is controlled by an I/O port (PF1) of the H8S/2258F on the receiving side.

(4) Configuration of On-Board Reprogramming

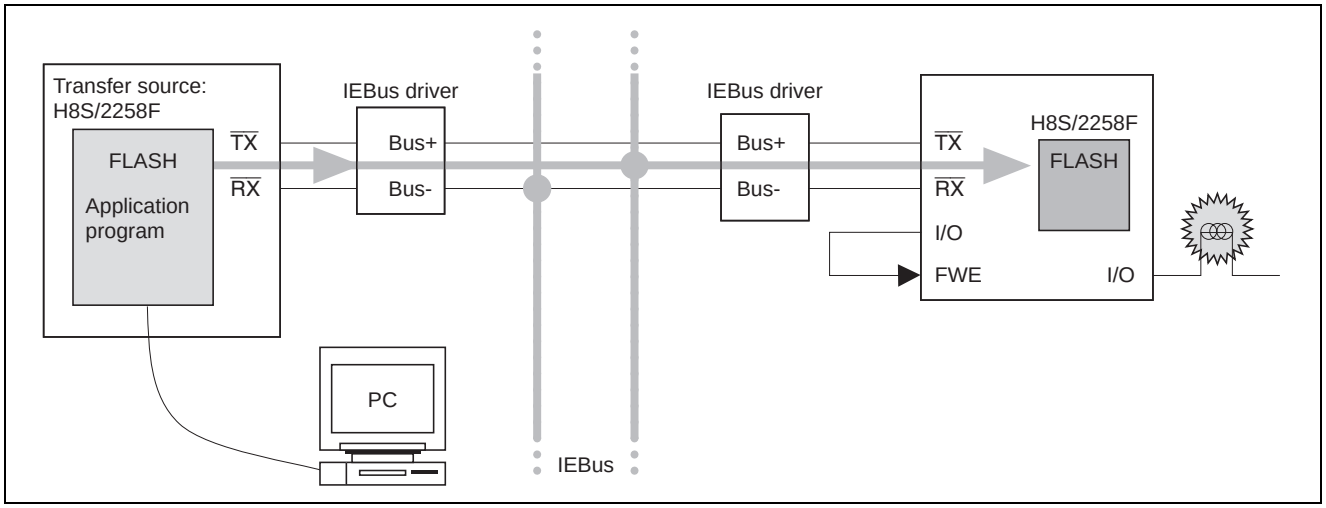


Figure 1 Configuration Diagram for On-Board Reprogramming Using IEBus Interface

(5) Execution Procedure for On-Board Reprogramming of H8S/2258F

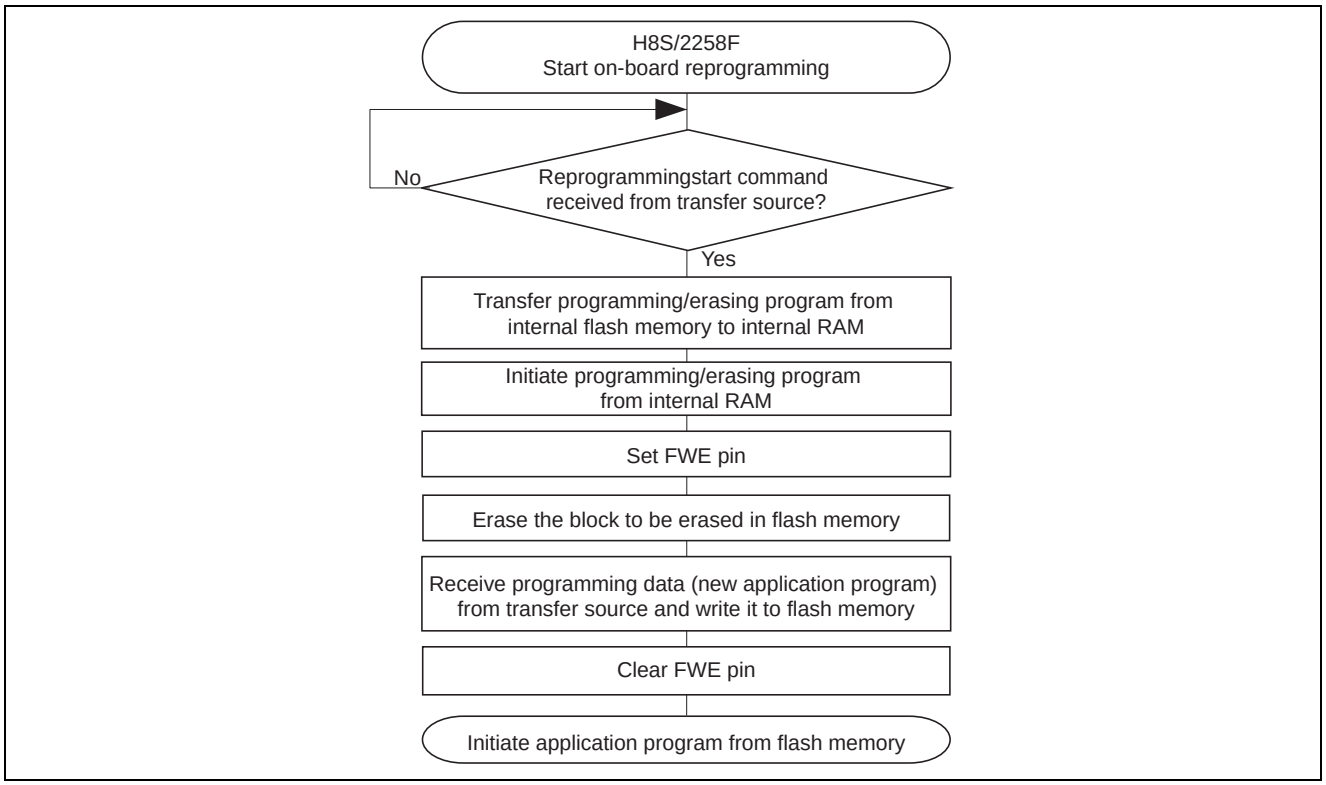


Figure 2 Procedure for On-Board Reprogramming Using IEBus Interface

2. Overview of Operation

The principles of operation for on-board reprogramming using an IEBus interface are described below.

2.1 Normal Operation

(1) The programming/erasing program is written to a portion of the application program area of flash memory. Furthermore, the FWE pin setting procedure and application transfer procedure are prepared ahead of time.

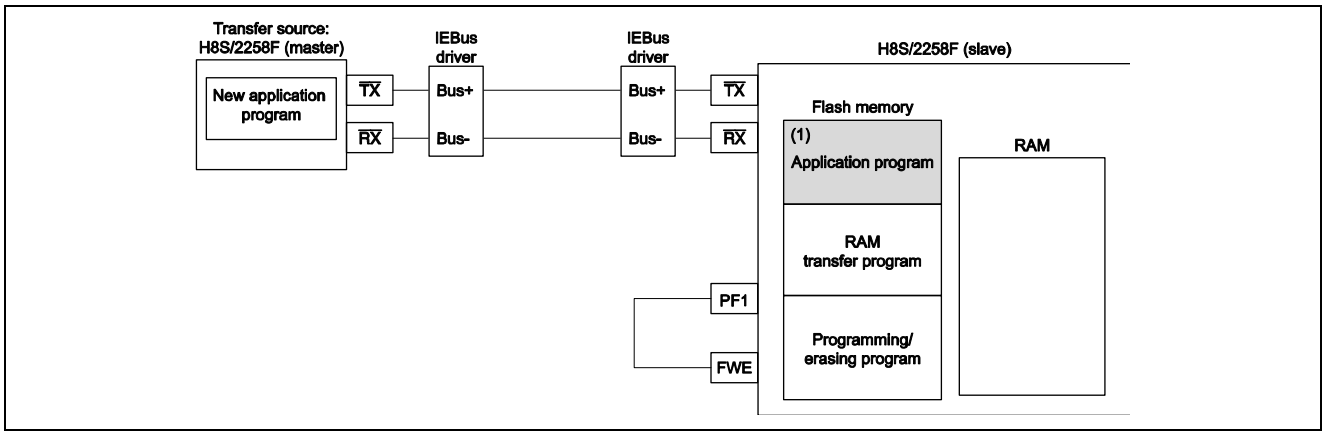


Figure 3 Normal Operation

2.2 Starting On-Board Reprogramming

(1) The command to start reprogramming is received from the transfer source.

(2) The application program runs the RAM transfer program to transfer the programming/erasing program from flash memory to internal RAM.

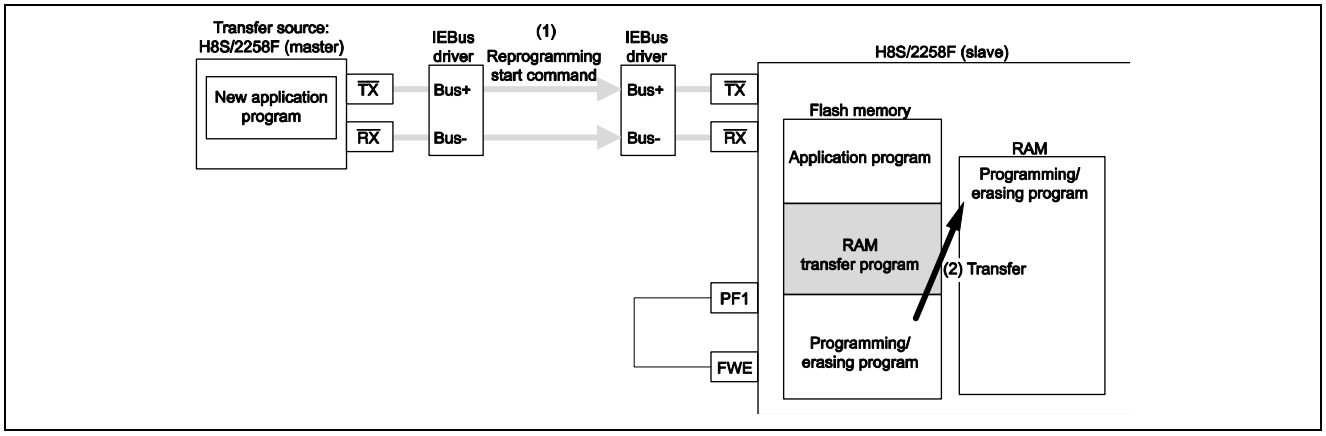


Figure 4 Starting On-Board Reprogramming

2.3 Running Programming/Erasing Program

(1) After the RAM transfer program completes the transfer, execution branches to the programming/erasing program.

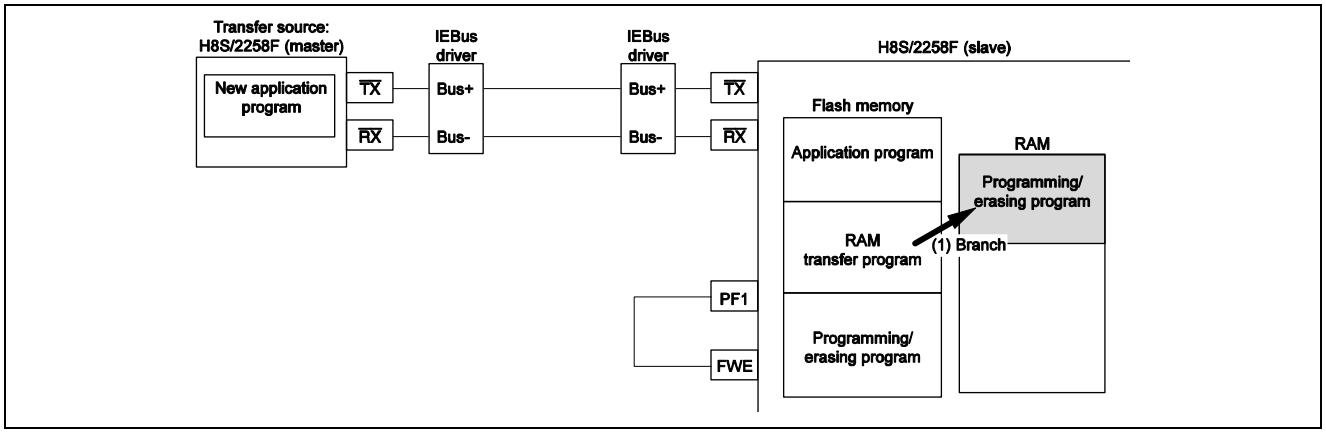


Figure 5 Running Programming/Erasing Program

2.4 Setting the FWE Pin

(1) The command to set the FWE pin is received from the transfer source.
(2) The programming/erasing program controls PF1 and sets the FWE pin to 1.

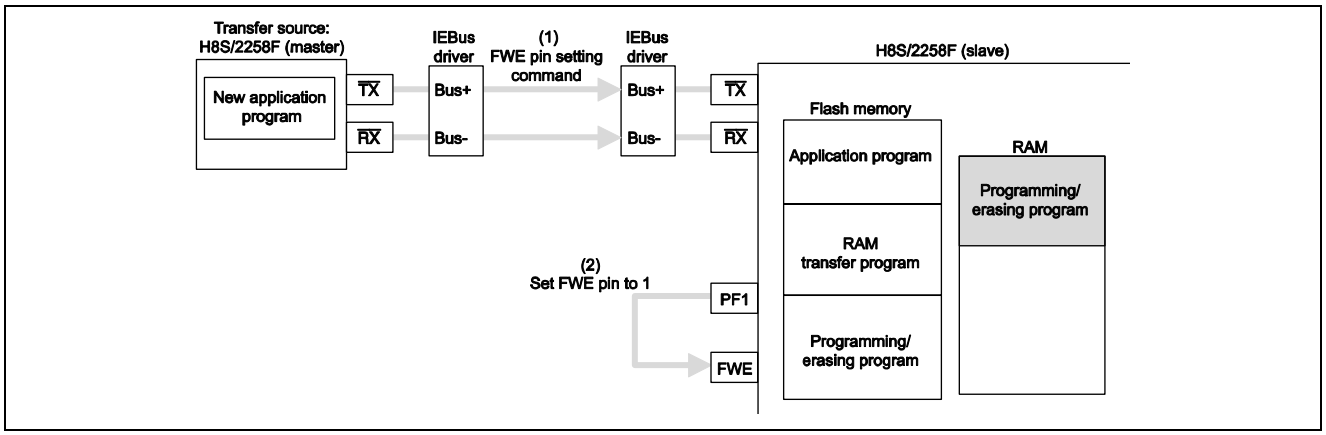


Figure 6 Setting the FWE Pin

2.5 Erasing Flash Memory Contents

(1) The erase command is received from the transfer source.
(2) The programming/erasing program erases the erase target block of the flash memory.

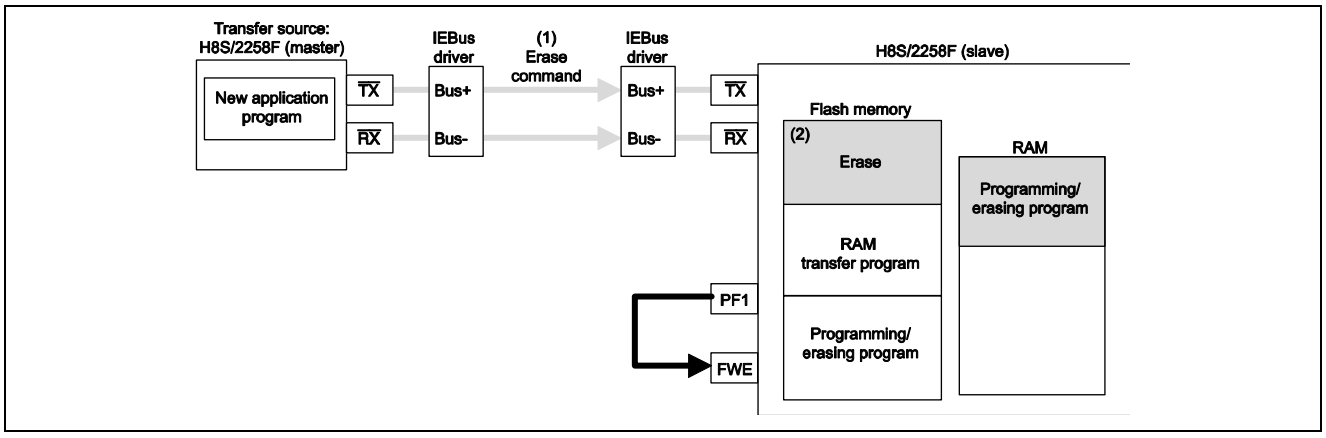


Figure 7 Erasing Flash Memory Contents

2.6 Programming the Flash Memory

(1) The programming command is received from the transfer source.
(2) The programming/erasing program receives the new application program from the transfer source and writes it to the flash memory.

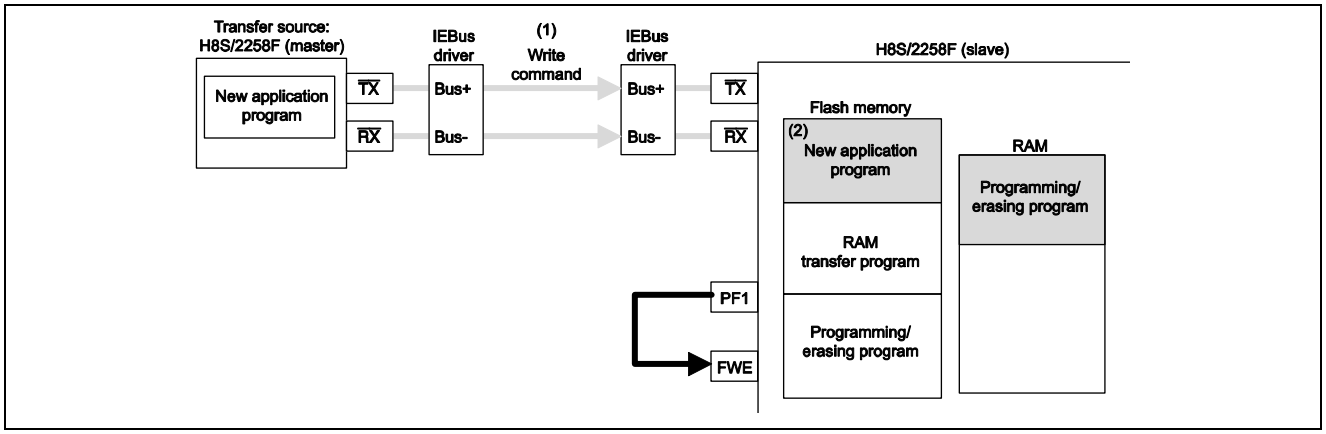


Figure 8 Programming the Flash Memory

2.7 Clearing the FWE Pin

(1) The command to clear the FWE pin is received from the transfer source.
(2) The programming/erasing program controls PF1 and clears the FWE pin to 0.

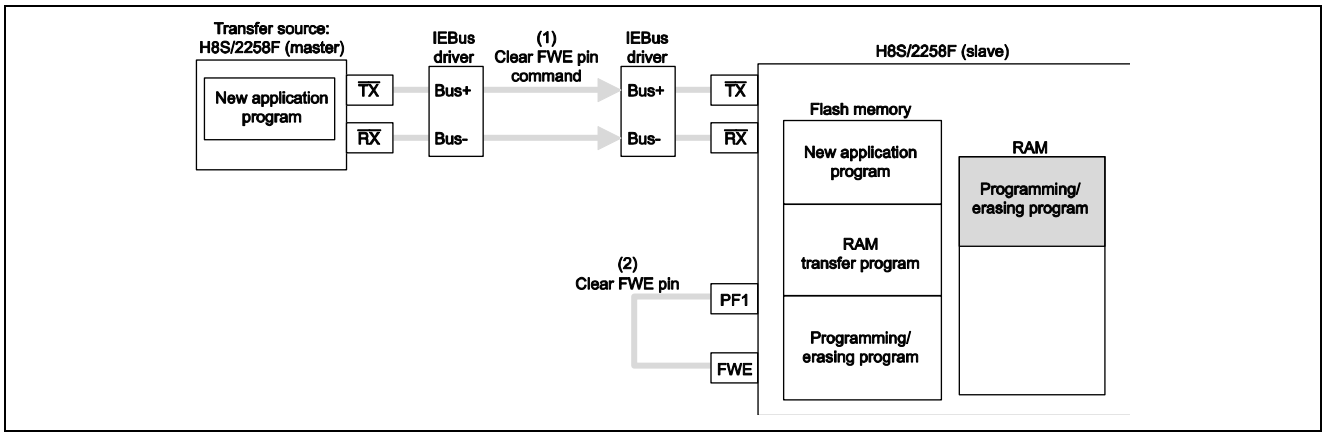


Figure 9 Clearing the FWE Pin to 0

2.8 Running New Application Program

(1) The programming/erasing program branches to the new application program in the flash memory.

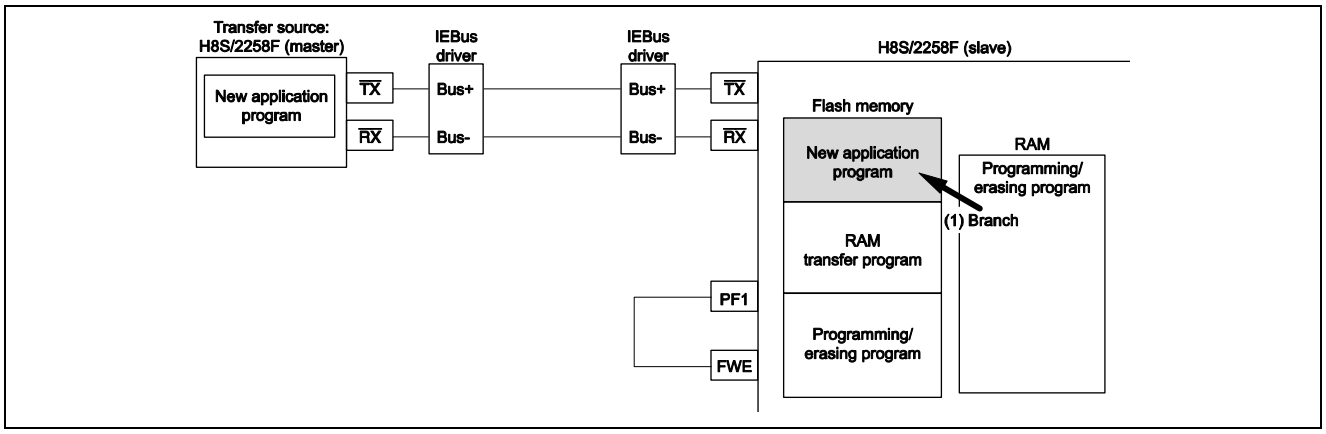


Figure 10 Running the New Application Program

3. Principles of Operation

Details of the on-board reprogramming communication operations involved in 1-to-1 communication and the IEBus interface are illustrated in figures 11 to 15. Note that parity bits transmitted by the transfer source and acknowledge bits transmitted by the receiving side are omitted from the figures.

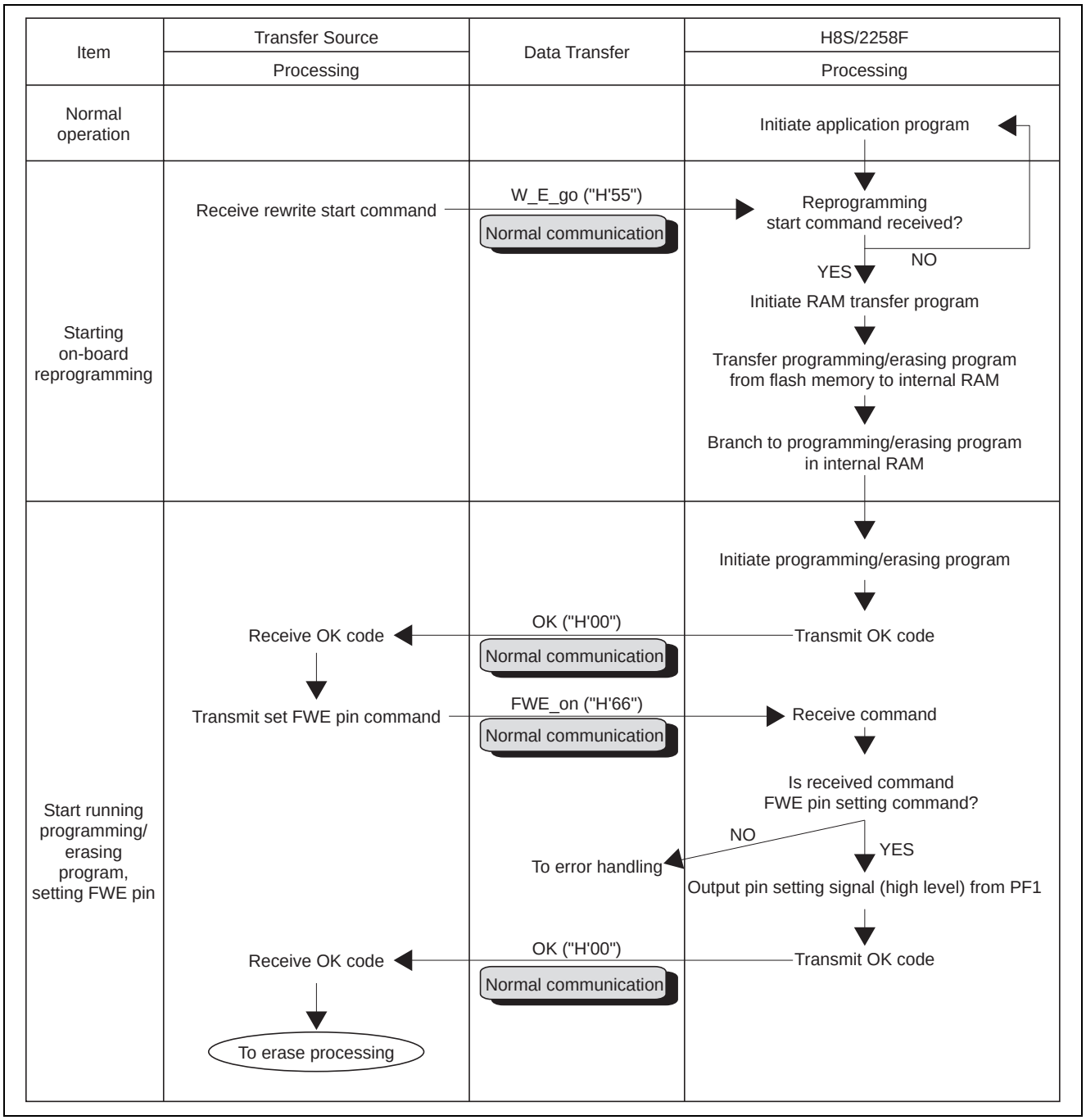
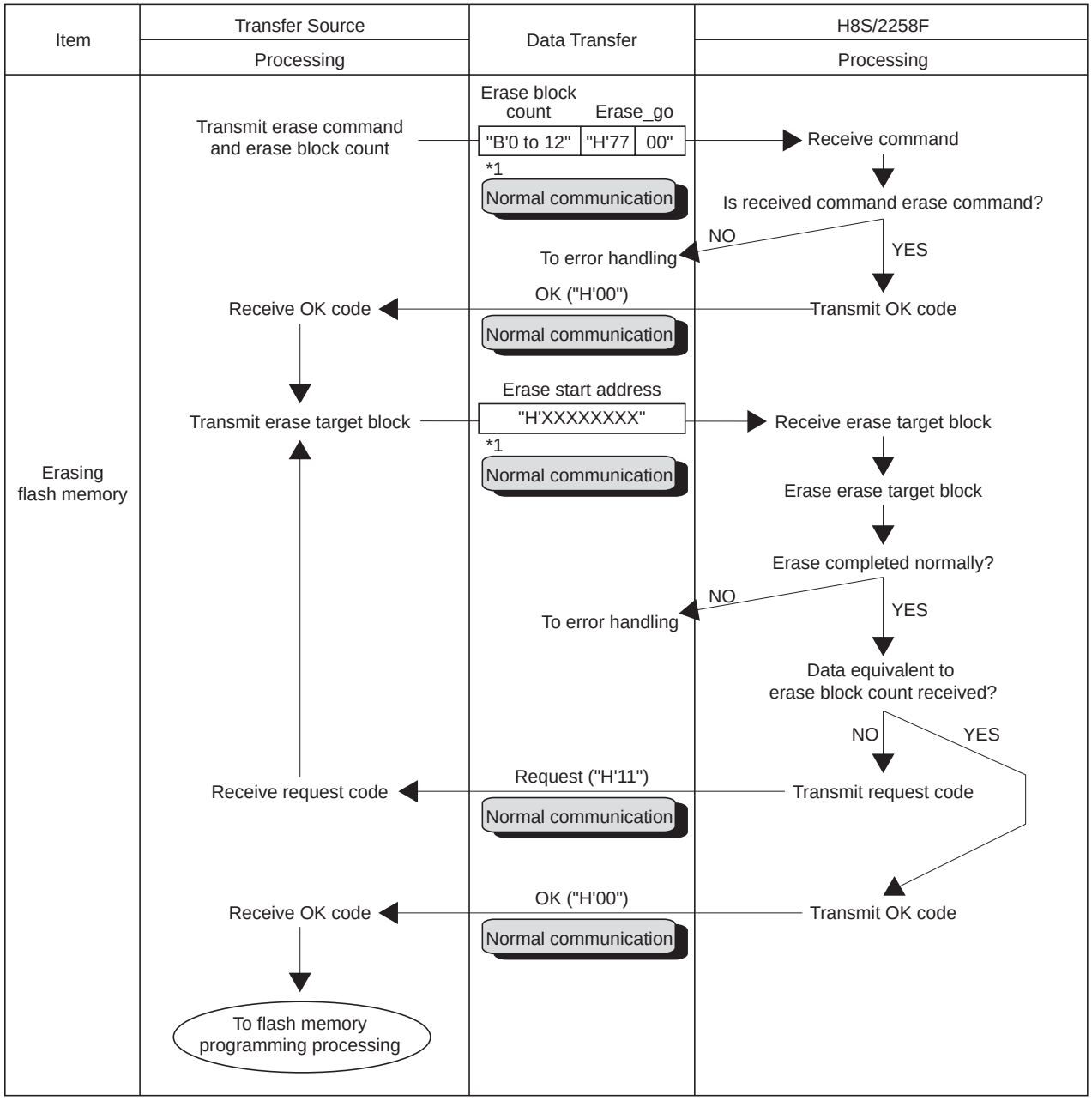


Figure 11 On-Board Reprogramming Communication Operations (1)

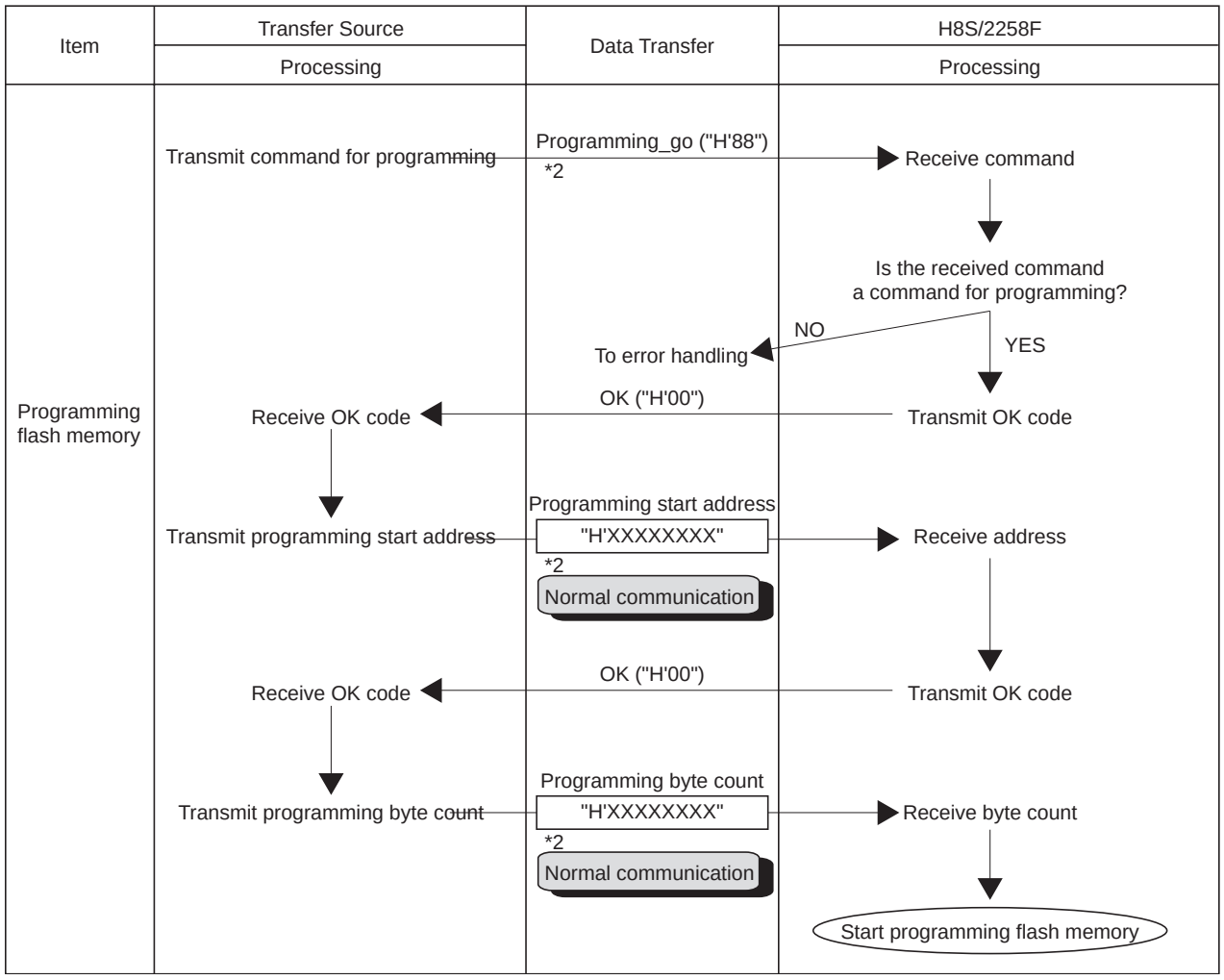
(Continues on next page.)



Note: 1. The transfer source transmits the erase block count and erase command in the first byte and the erase start address in the second and subsequent bytes.

Figure 12 On-Board Reprogramming Communication Operations (2)

(Continues on next page.)



Note: 2. The transfer source transmits in the first byte the write command for the application program to be transmitted, the write start address in the second byte, and the write byte count in the third byte.

Figure 13 On-Board Reprogramming Communication Operations (3)

(Continues on next page.)

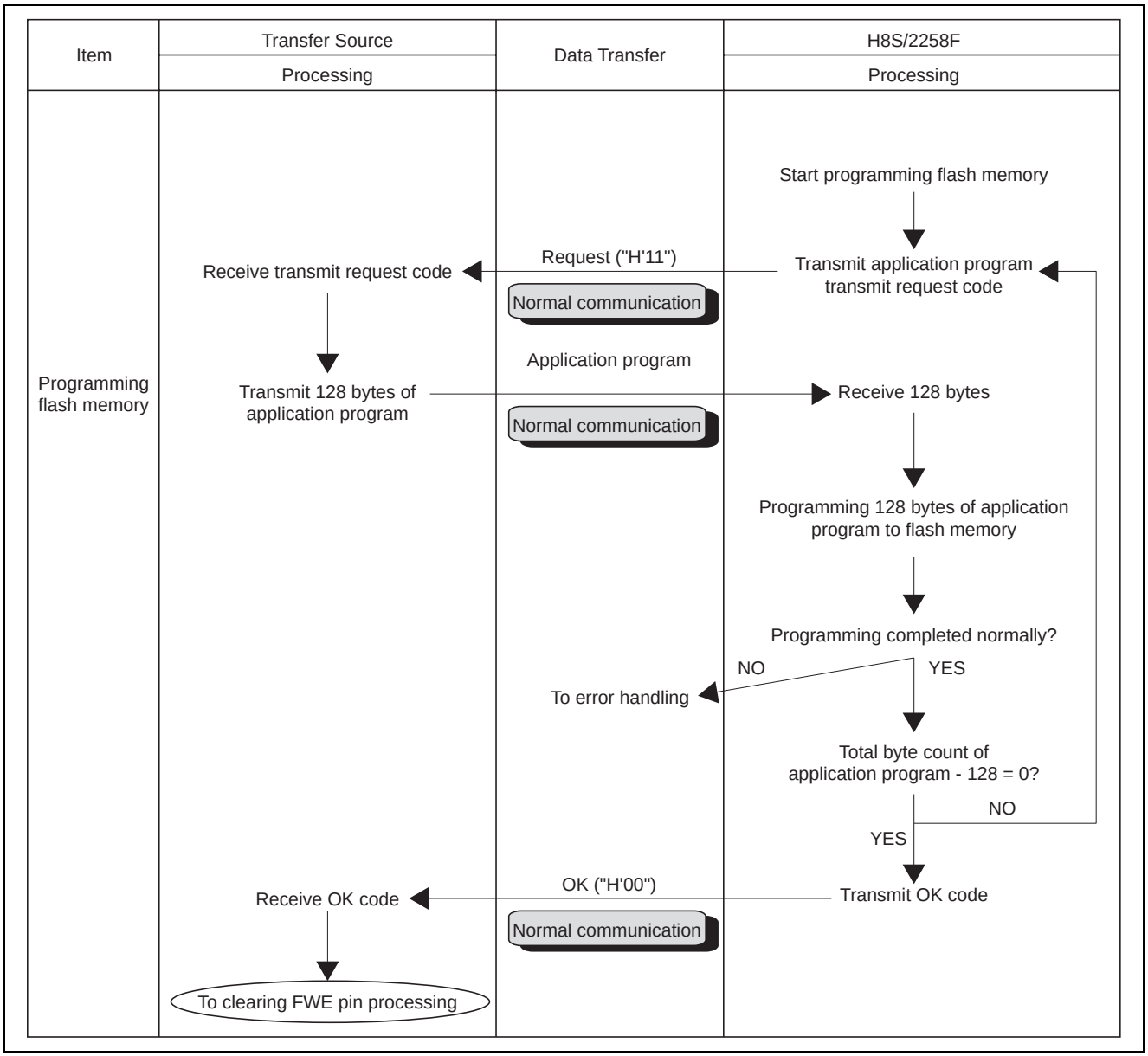


Figure 14 On-Board Reprogramming Communication Operations (4)

(Continues on next page.)

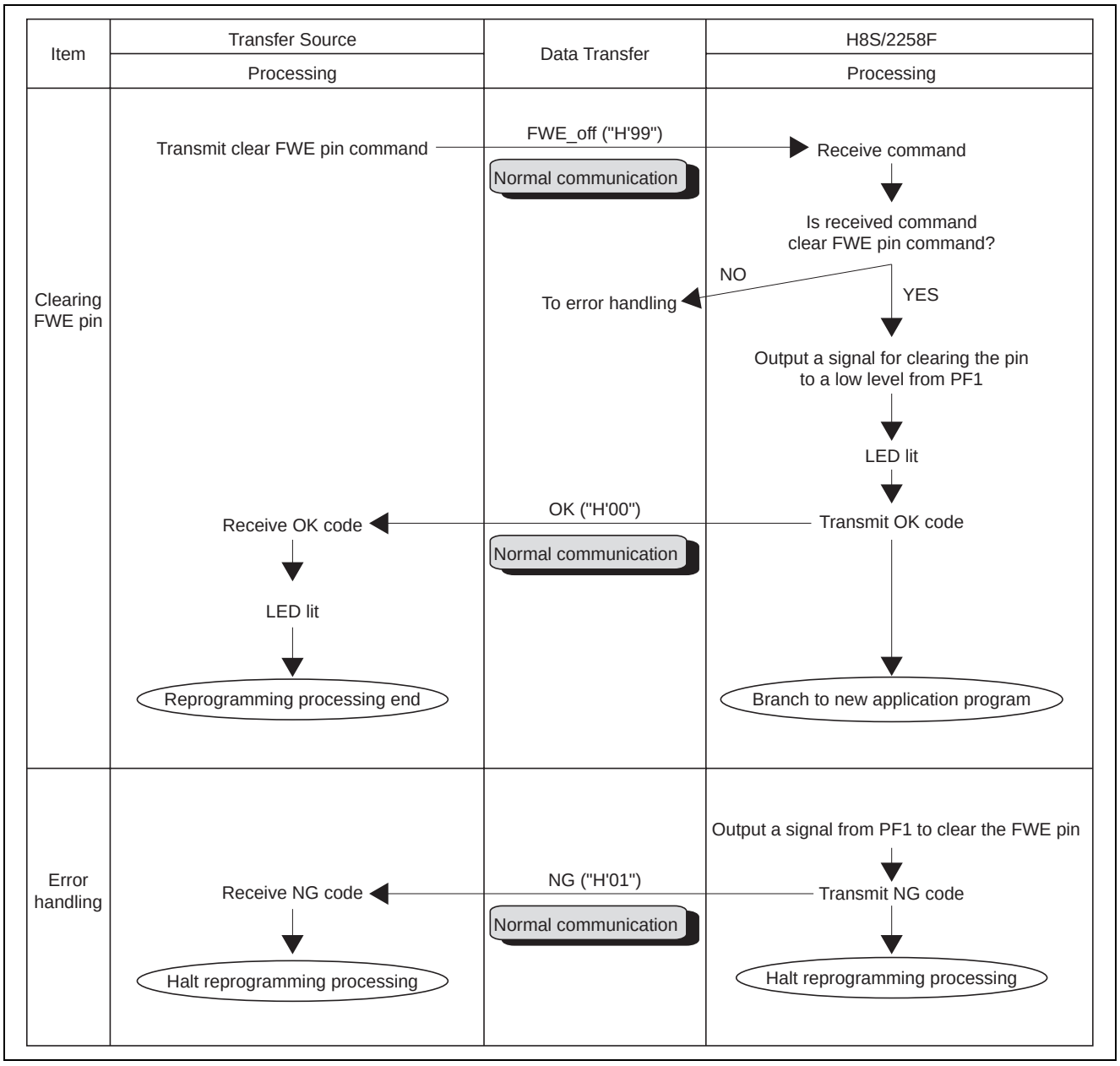
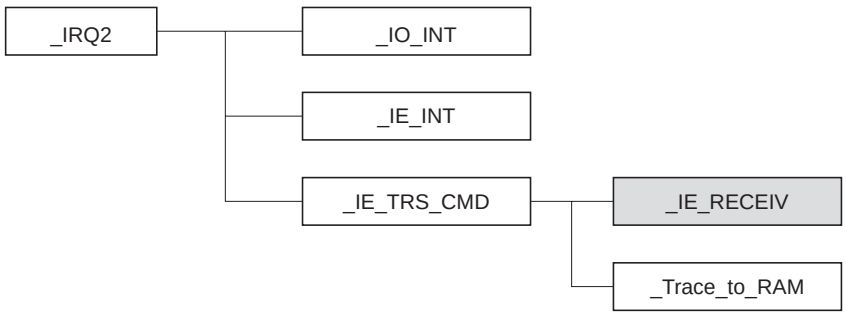


Figure 15 On-Board Reprogramming Communication Operations (5)

4. Description of Software

4.1 Hierarchy of User Program Mode Startup Program

The hierarchy of the user program mode startup program (processing for receiving the command to start reprogramming and transferring the programming/erasing program from the flash memory to the internal RAM), which is run from flash memory, is shown in figure 16.



Note: Dark shading (▓) indicates module described in section 4.4, Modules of Programming/Erasing Program.

Figure 16 Hierarchy of User Program Mode Startup Program

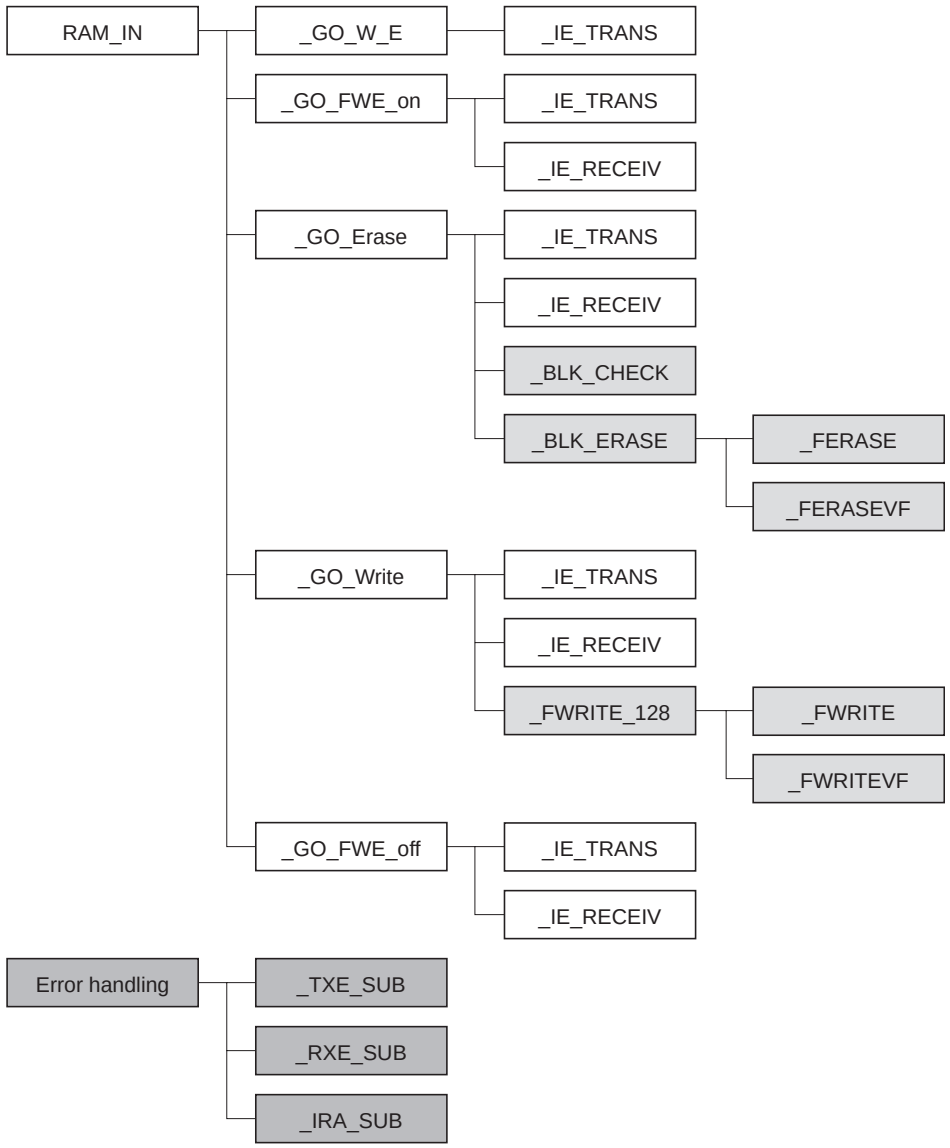
4.2 Modules of User Program Mode Startup Program

Table 1 Modules of User Program Mode Startup Program

	Arguments		Return Values		
Module	**Description**	**Register (Data Length)**	**Description**	**Register (Data Length)**	**Assignment of Functions**
_IRQ2	—	—	—	—	Starts reprogramming of flash memory using IRQ2
_IO_INT	—	—	—	—	Initializes I/O register
_IE_INT	—	—	—	—	Initializes IEBus register
_IE_TRS_CMD	Start rewrite command	ROH (byte)	—	—	Waits for start rewrite command
_Trace_to_RAM	—	—	—	—	Transfers programming/erasing program to internal RAM

4.3 Hierarchy of Programming/Erasing Program

The hierarchy of the programming/erasing program, which is run from flash memory, is shown in figure 17.



Notes: 1. Light shading (▒) indicates modules for processing of flash memory erasing/programming operations.
2. Dark shading (▓) indicates modules for handling of _IE_TRANS and _IE_RECEIV errors during IEBus communication operations.

Figure 17 Hierarchy of Programming/Erasing Program

4.4 Modules of Programming/Erasing Program

Table 2 Modules of Programming/Erasing Program

	Arguments		Return Values		
Module	Description	Register (Data Length)	Description	Register (Data Length)	Assignment of Functions
_RAM_IN	—	—	—	—	Main routine of programming/erasing program
_GO_W_E	—	—	OK command	R4H (byte)	Checks command to start programming and transmits OK command
_GO_FWE_on	Set FWE command	R0H (byte)	OK command	R4H (byte)	Sets FWE pin to 1
_GO_Erase	Erase start command	R0H (byte)	OK command or Request command or E_ERR command	R4H (byte)	Controls erasing of flash memory
	Erase block count	E0 (word)			
	Erase start address	ER3 (longword)			
_GO_Write	Write command	R0H (byte)	OK command or Request command	R4H (byte)	Controls programming of flash memory
	Write start address	ER0 (longword)			
	Write byte count	ER0 (longword)			
_GO_FWE_off	Clear FWE command	R0H (byte)	OK command	R4H (byte)	Clears FWE pin to 0
_IE_TRANS	Broadcast bit #DH_BIT	R0L (byte)	Transmit data using the IEBus	@IETBR (n × byte)	Transmits data using the IEBus
	Slave address #SADR_H	R0L (byte)			
	Slave address #SADR_L	R0L (byte)			
	Transmission message length	R3L (byte)			
	Transmission data	R0L (byte)			
_IE_RECEIV	Receive data using the IEBus	R0H (byte)	Received data	#RDATA_BUFF (n × byte)	Receives data using the IEBus
_BLK_CHECK	Erase start address	ER3 (longword)	1 for normal, 0 for error	R0L (longword)	Checks which block the erase start address corresponds to and loads the appropriate FLMCR and EBR register addresses into general registers. Furthermore, stores the erase target block number, erase verify start address, and erase verify end address in work RAM
			Address in EBR register	ER5 (longword)	
			Address in FLMCR register	ER6 (longword)	
			Erase target block no.	@BLK_NO (longword)	
			Erase verify start address	@EVF_ST (longword)	
			Erase verify end address	@EVF_ED (longword)	
_BLK_ERASE	—	—	1 for normal, 0 for error	R0L (byte)	Performs flash memory erase processing
_FWRITE_128	Write start address	@WT_ADR (longword)	1 for normal, 0 for error	R0L (byte)	Performs flash memory programming processing in 128-byte units
	Programming data	@RDATA_BUFF (128 × byte)	Overwrite data	@RW_BUFF (128 × byte)	

Note: Light shading (▒) indicates modules for processing of flash memory erasing/programming operations.

Table 2 Modules of Programming/Erasing Program (cont)

	Arguments		Return Values		
Module	**Description**	**Register (Data Length)**	**Description**	**Register (Data Length)**	**Assignment of Functions**
_FERASE	Address in EBR register	ER5 (longword)	1 for normal, 0 for error	R0L (byte)	Erases area of erase target block number
	Address in FLMCR register	ER6 (longword)			
	Erase target block No.	@BLK_NO (byte)			
_FERASEVF	Address in FLMCR register	ER6 (longword)	1 for normal, 0 for error	R0L (byte)	Reads data from erase target start address to erase target end address and verifies that the erase operation completed normally
	Erase verify start address	@EVF_ST (longword)			
	Erase verify end address	@EVF_ED (longword)			
_FWRITE	P bit setting time	ER3 (longword)	—	—	Programs flash memory (applying flash voltage) in 128-byte units
	Address in FLMCR register	ER6 (longword)			
	Programming start address	@WT_ADR (longword)			
_FWRITEVF	Address in FLMCR register	ER6 (longword)	1 for normal, 0 for error	R0L (byte)	Performs overwrite and additional write operations and verifies that 128 bytes of programming data was programmed correctly
	Reprogramming data	@RW_BUFF (128 × byte)	Reprogramming data	@RW_BUFF (128 × byte)	
	Programming data	@RDATA_BUFF (128 × byte)	Additional programming data	@OW_BUFF (128 × byte)	
	Additional programming data	@OW_BUFF (128 × byte)			
_TXE_SUB	Contents of transmission error	@IETEF (byte)	Error code	R4H (byte)	Performs transmission error handling and returns an error code
_RXE_SUB	Contents of receive error	@IEREF (byte)	Error code	R4H (byte)	Performs reception error handling and returns an error code
_IRA_SUB	—	—	Error code	R4H (byte)	Performs error handling in runaway state and returns an error code

Notes: Light shading (▒) indicates modules for processing of flash memory erasing/programming operations.
Dark shading (▓) indicates modules for handling of _IE_TRANS and _IE_RECEIV errors during IEBus communication operations.

4.5 RAM Usage

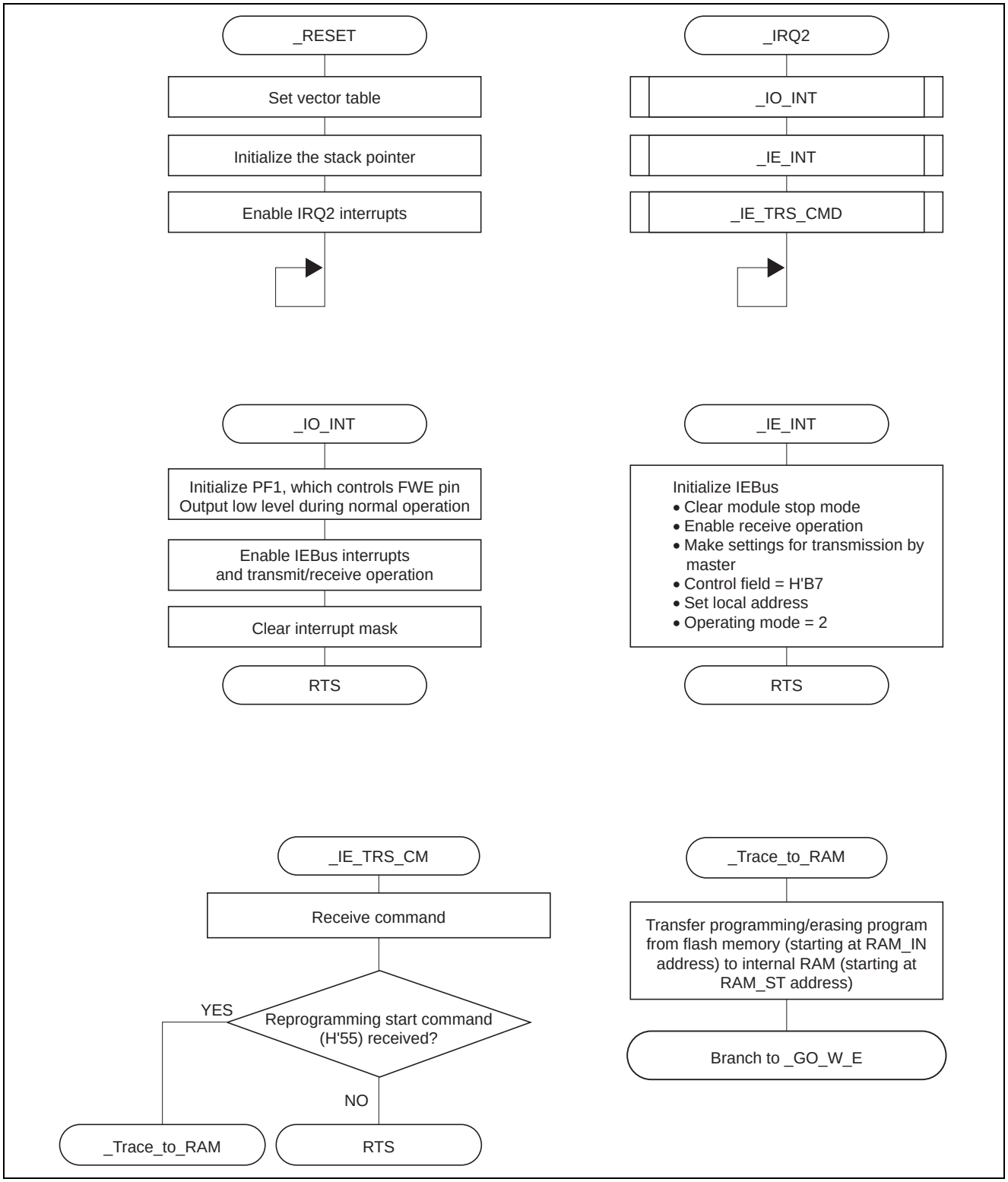
The internal RAM used as the storage and work areas for the programming control program is listed in table 3.

Table 3 RAM Usage

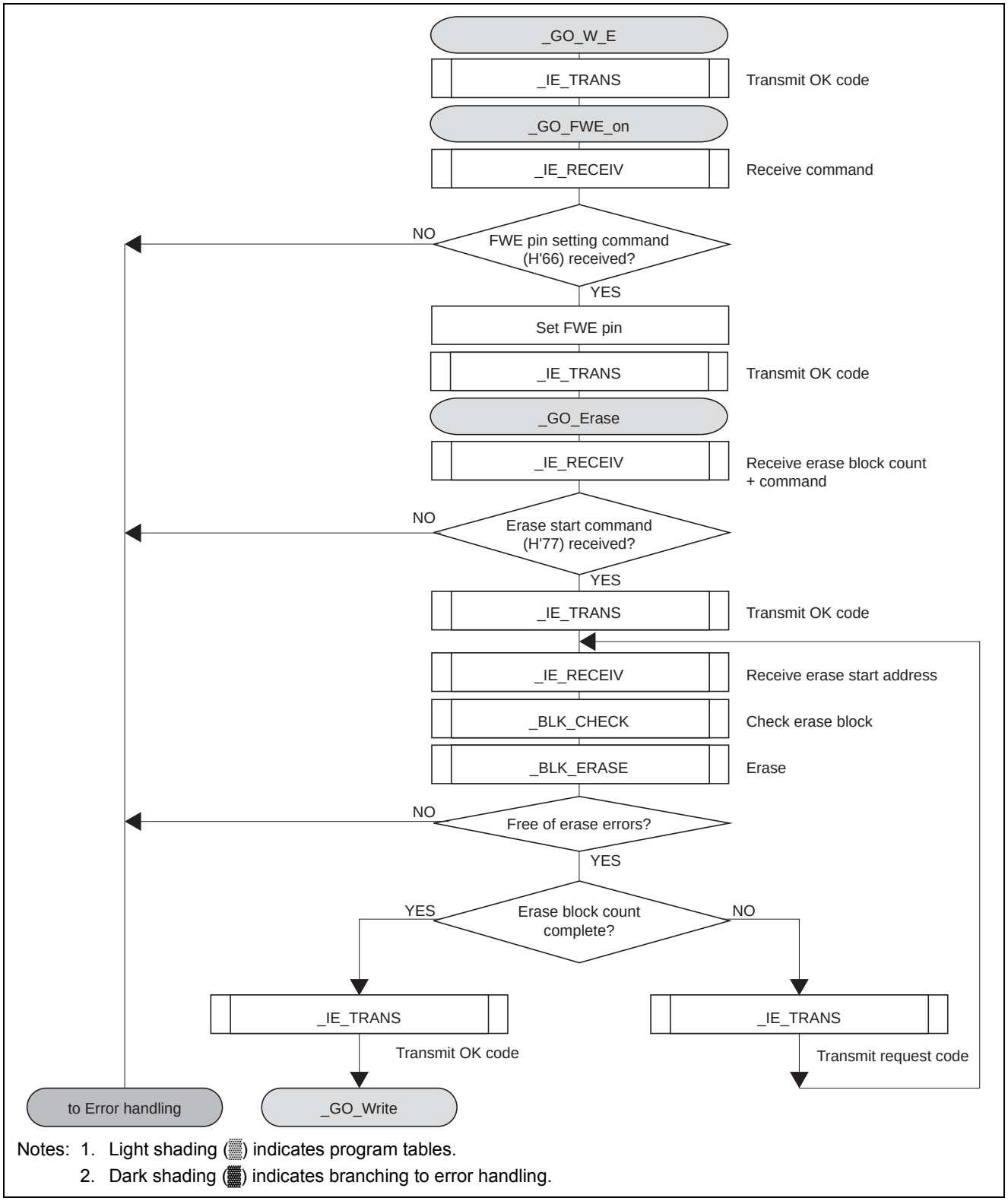
Label	Function		Data Length	Modules
TRANS_BUFF*1	THEDDA_BUFF	Stores broadcast/normal bit	1 (byte)	_IE_TRS_CMD _GO_W_E _GO_FWE_on _GO_Erase _GO_Write _GO_FWE_off _IE_TRANS
	SADR_BUFF_H	Stores slave address	1 (byte)	
	SADR_BUFF_L		1 (byte)	
	TDENBUN_BUFF	Stores message length bit for transmission	1 (byte)	
	TDUMMY	Dummy	1 (byte)	
	TDATA_BUFF	Stores transmission data	128 (bytes)	
RECEIV_BUFF*2	RDENBUN_BUFF	Stores message length bit for reception	1 (byte)	_IE_TRS_CMD _GO_W_E _GO_FWE_on _GO_Erase _GO_Write _GO_FWE_off _IE_RECEIV
	IEBUS_ERROR	Stores communication error code	1 (byte)	
	RDATA_BUFF	Stores received data	128 (bytes)	
WT_ADR	Stores programming address		4 (bytes)	_GO_Write _FWRITEVF _FWRITE
RW_BUFF	Stores reprogramming address		128 (bytes)	_FWRITE_128 _FWRITEVF
OW_BUFF	Stores additional write address		128 (bytes)	_FWRITE_128 _FWRITEVF
COUNT	Stores erase and programming count		2 (bytes)	_BLK_ERASE _FWRITE_128
ET_COUNT	Stores maximum erase count		2 (bytes)	_GO_ERASE _BLK_ERASE
WT_COUNT	Stores maximum programming count		2 (bytes)	_GO_WRITE _FWRITE_128
REST_SIZE	Stores programming data size		4 (bytes)	_GO_WRITE
EVF_ST	Stores erase verify start address		4 (bytes)	_BLK_CHECK _FERASEVF
EVF_ED	Stores erase verify end address		4 (bytes)	_BLK_CHECK _FERASEVF
BLK_NO	Stores erase block number		1 (byte)	_BLK_CHECK _FERASE
VF_RET	Stores programming verify result		1 (byte)	_FWRITE_128
ER_COUNT	Stores erase count		1 (byte)	_GO_ERASE

Notes: 1. An area sufficient for 6 bytes of transmission information plus 128 bytes of transmission data is set aside for transmission data use.
2. An area sufficient for 2 bytes of receive information plus 128 bytes of received data is set aside for received data use.

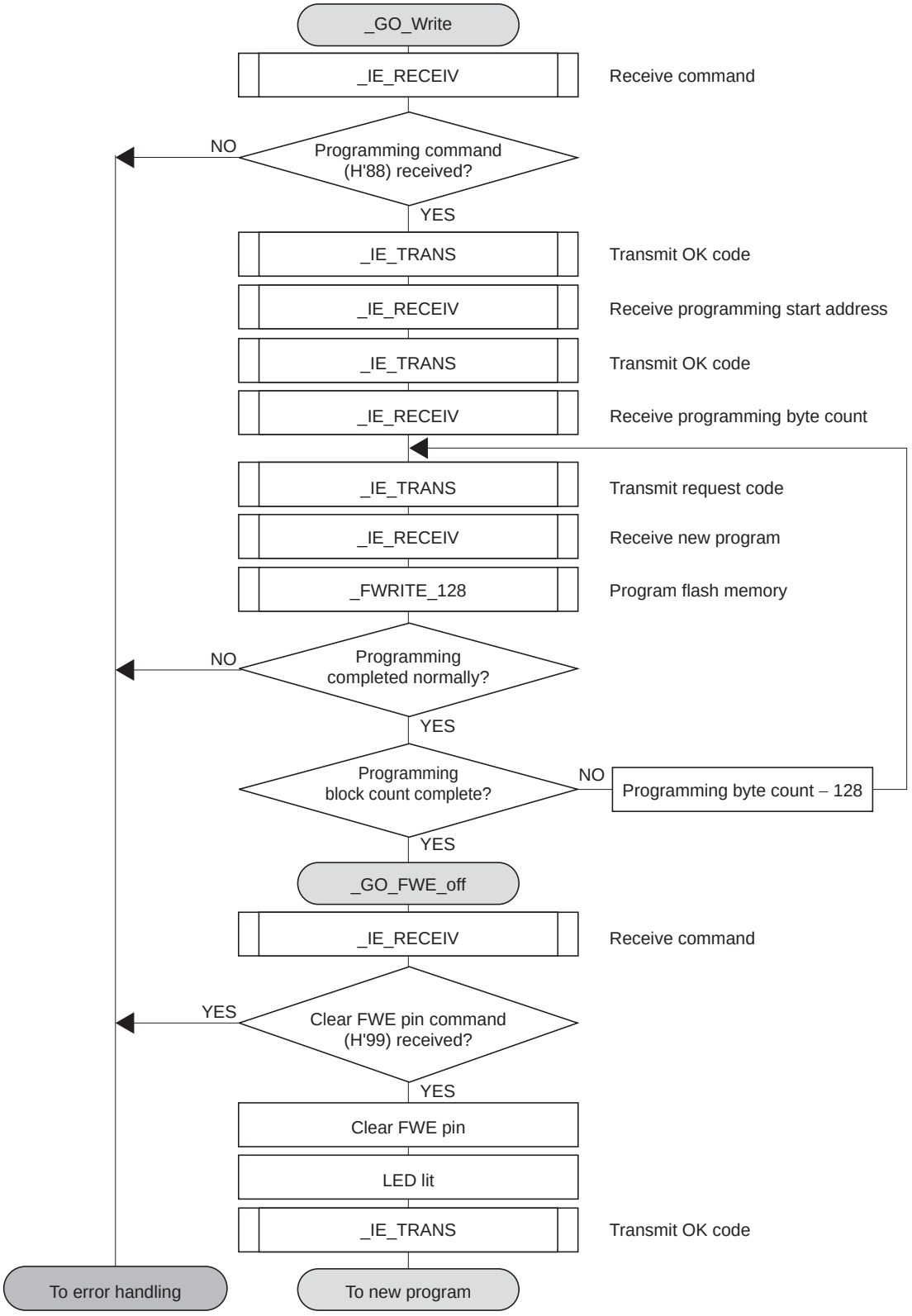
5. Flowcharts of User Program Mode Startup Program



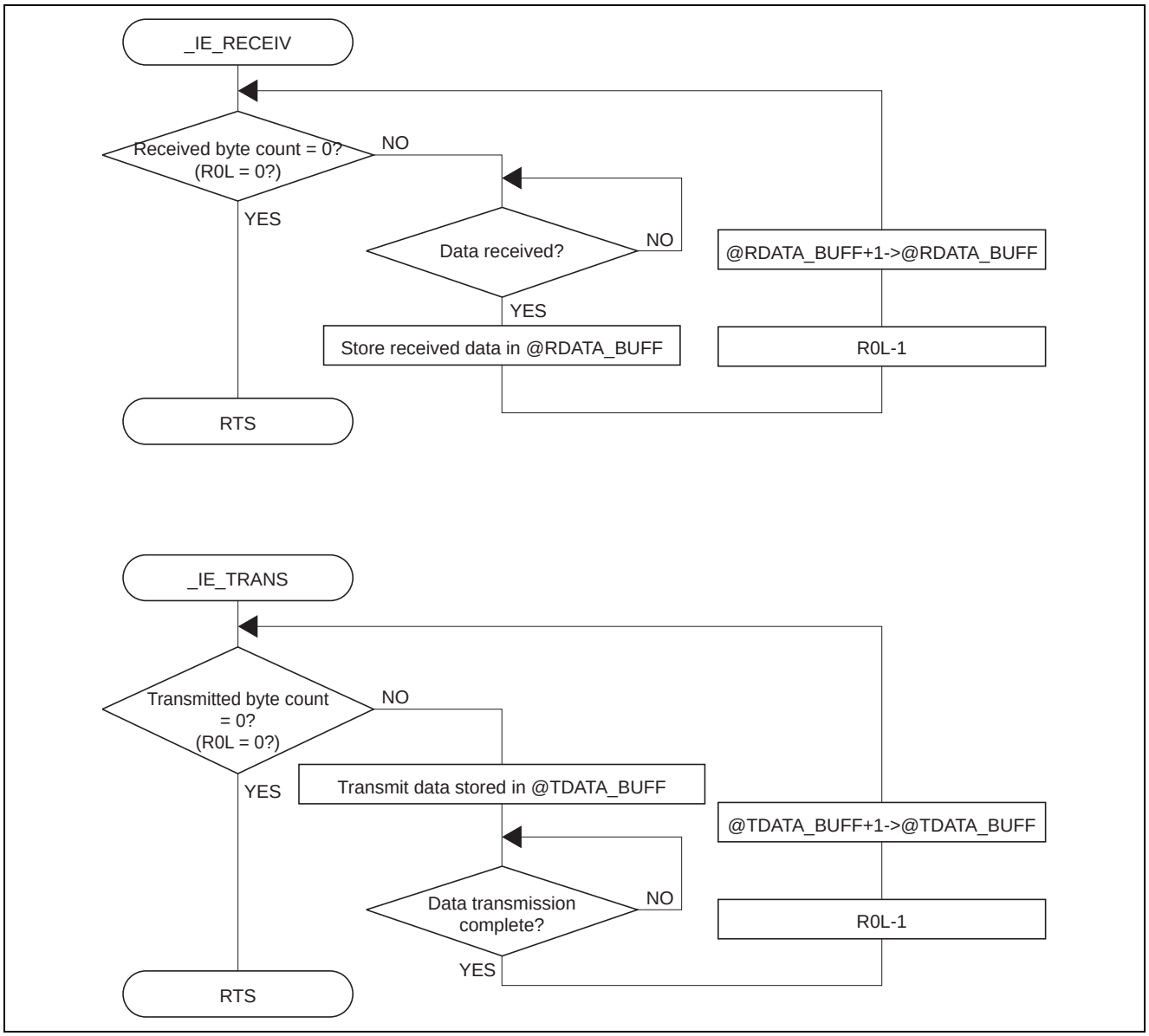
6. Flowcharts of Programming/Erasing Program

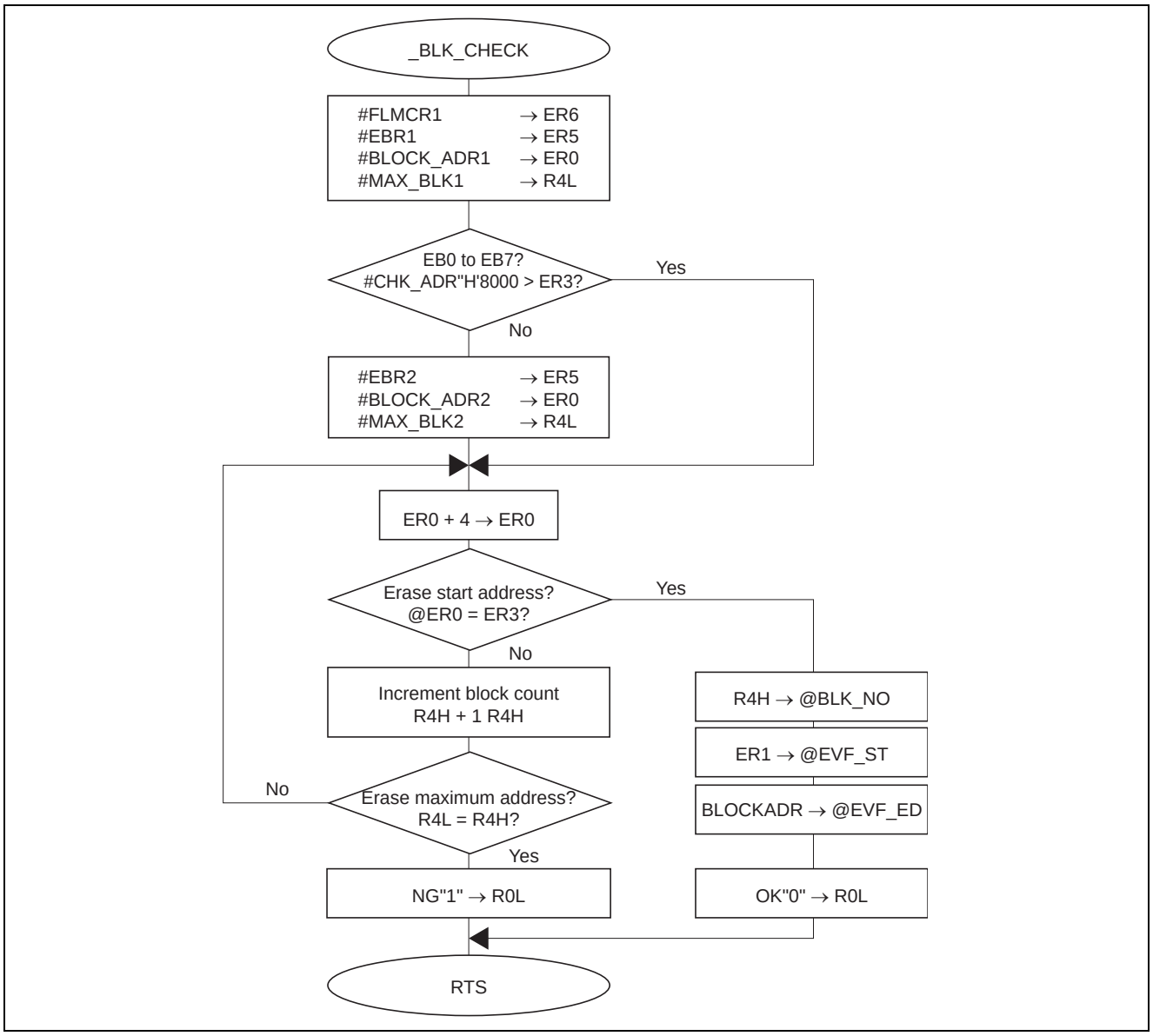


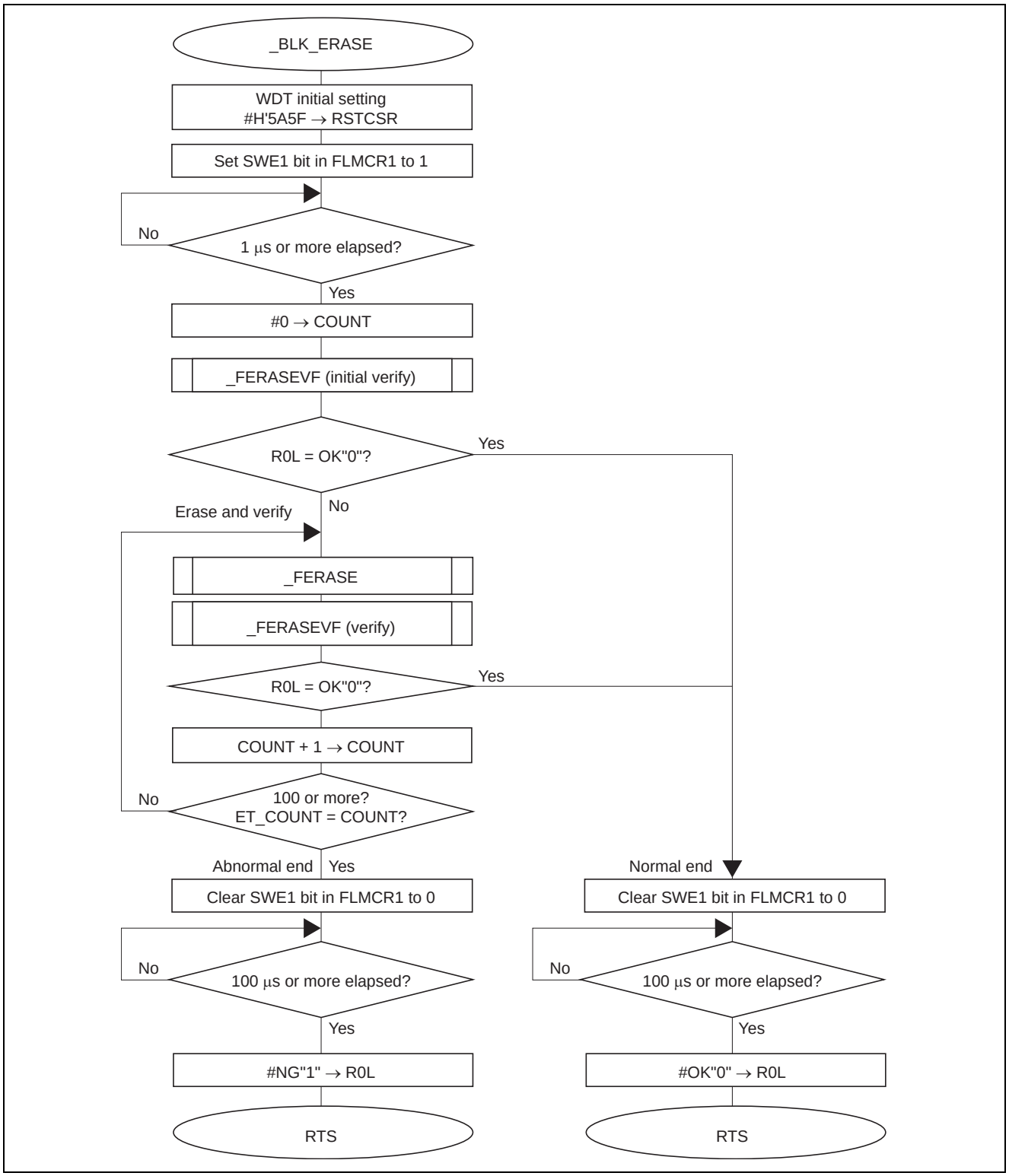
Notes: 1. Light shading (▒) indicates program tables.
2. Dark shading (▓) indicates branching to error handling.

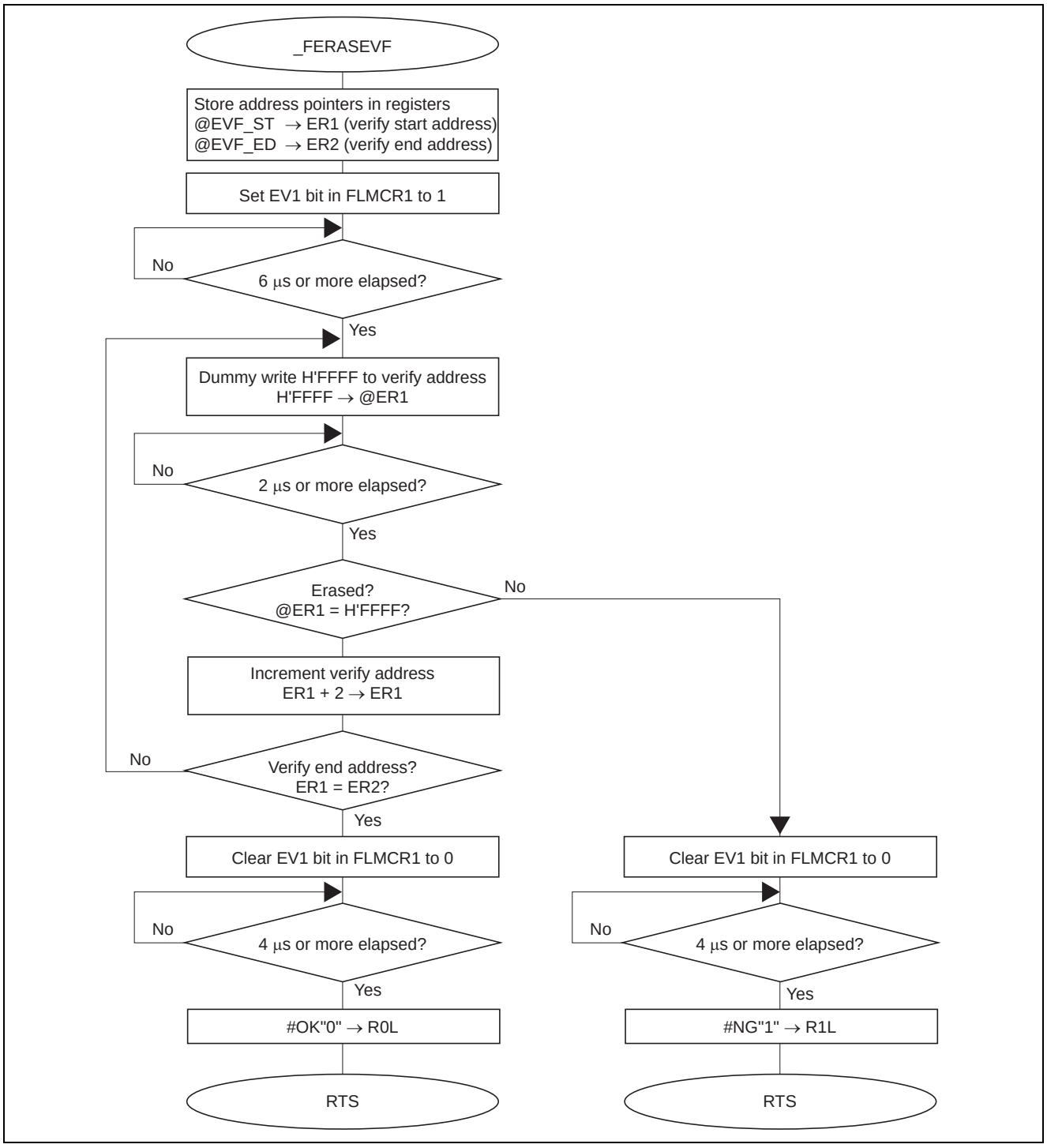


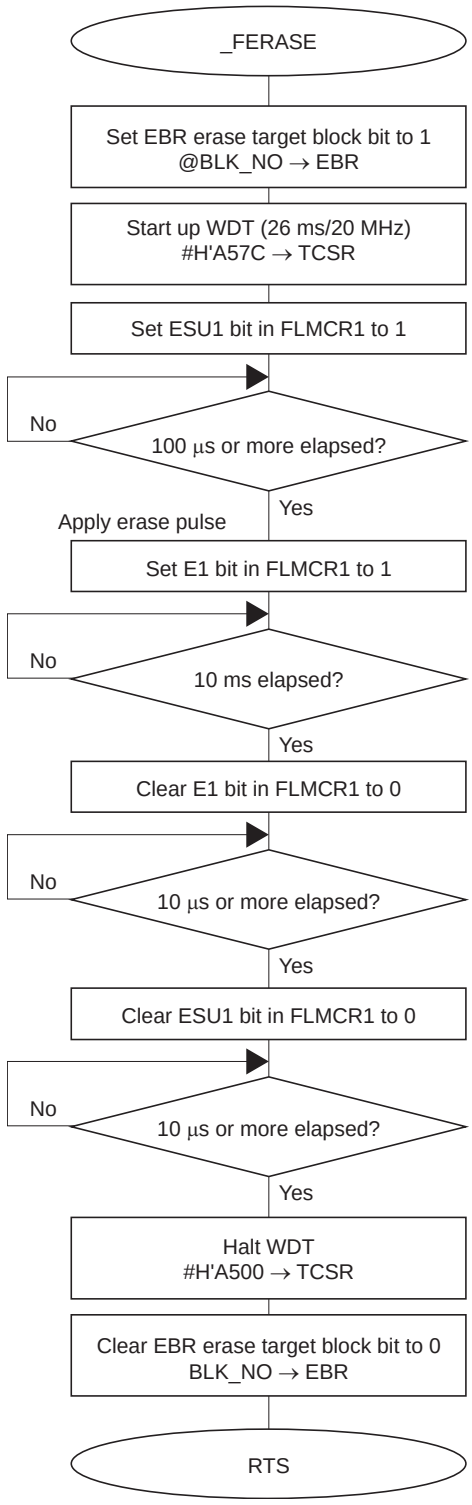
Notes: 1. Light shading (▒) indicates program tables.
2. Dark shading (▓) indicates branching to error handling.

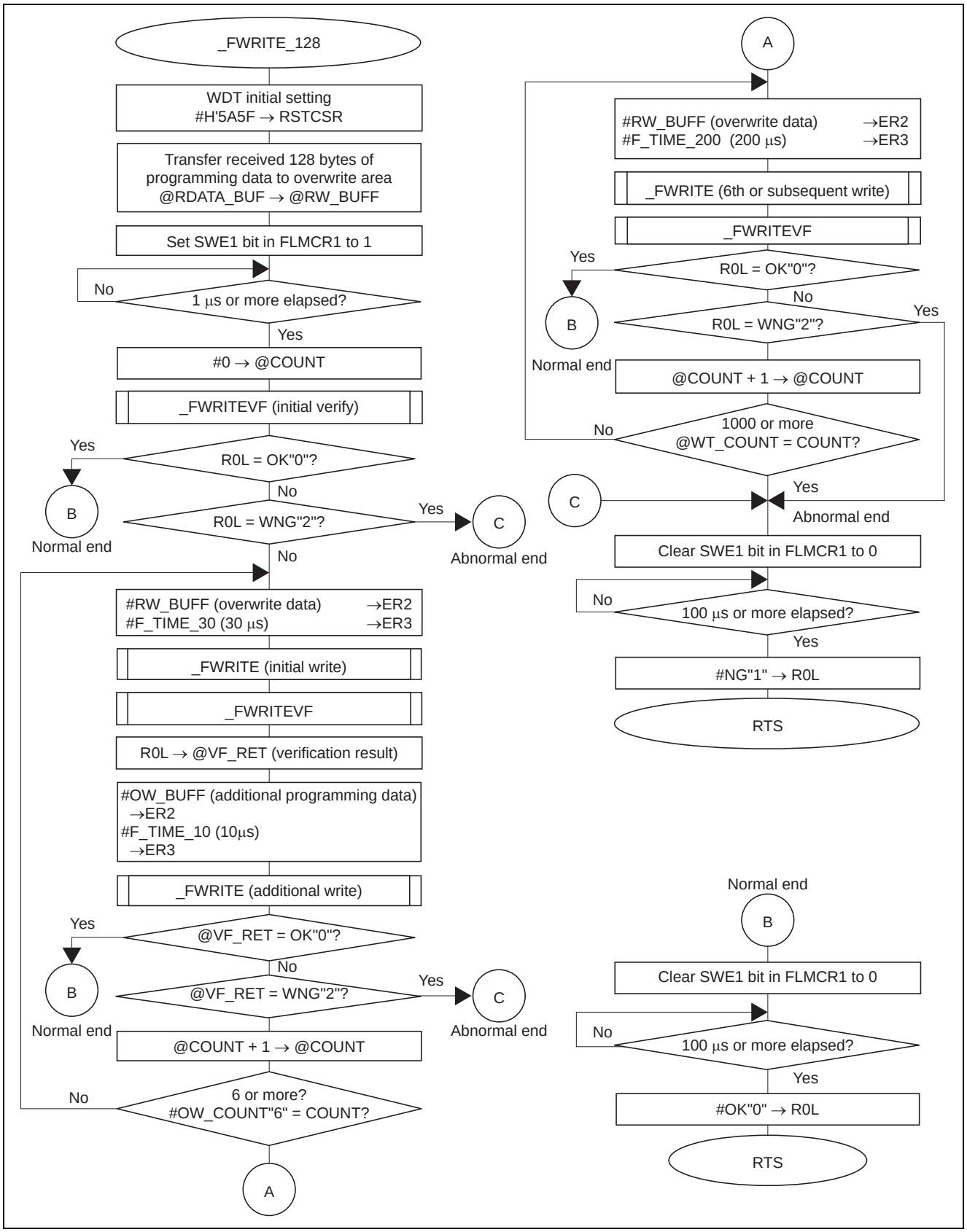

_IE_RECEIV
Received byte count = 0?
(R0L = 0?)
NO
YES
Data received?
NO
YES
@RDATA_BUFF+1->@RDATA_BUFF
Store received data in @RDATA_BUFF
R0L-1
RTS
_IE_TRANS
Transmitted byte count
= 0?
(R0L = 0?)
NO
YES
Transmit data stored in @TDATA_BUFF
@TDATA_BUFF+1->@TDATA_BUFF
Data transmission
complete?
NO
YES
R0L-1
RTS

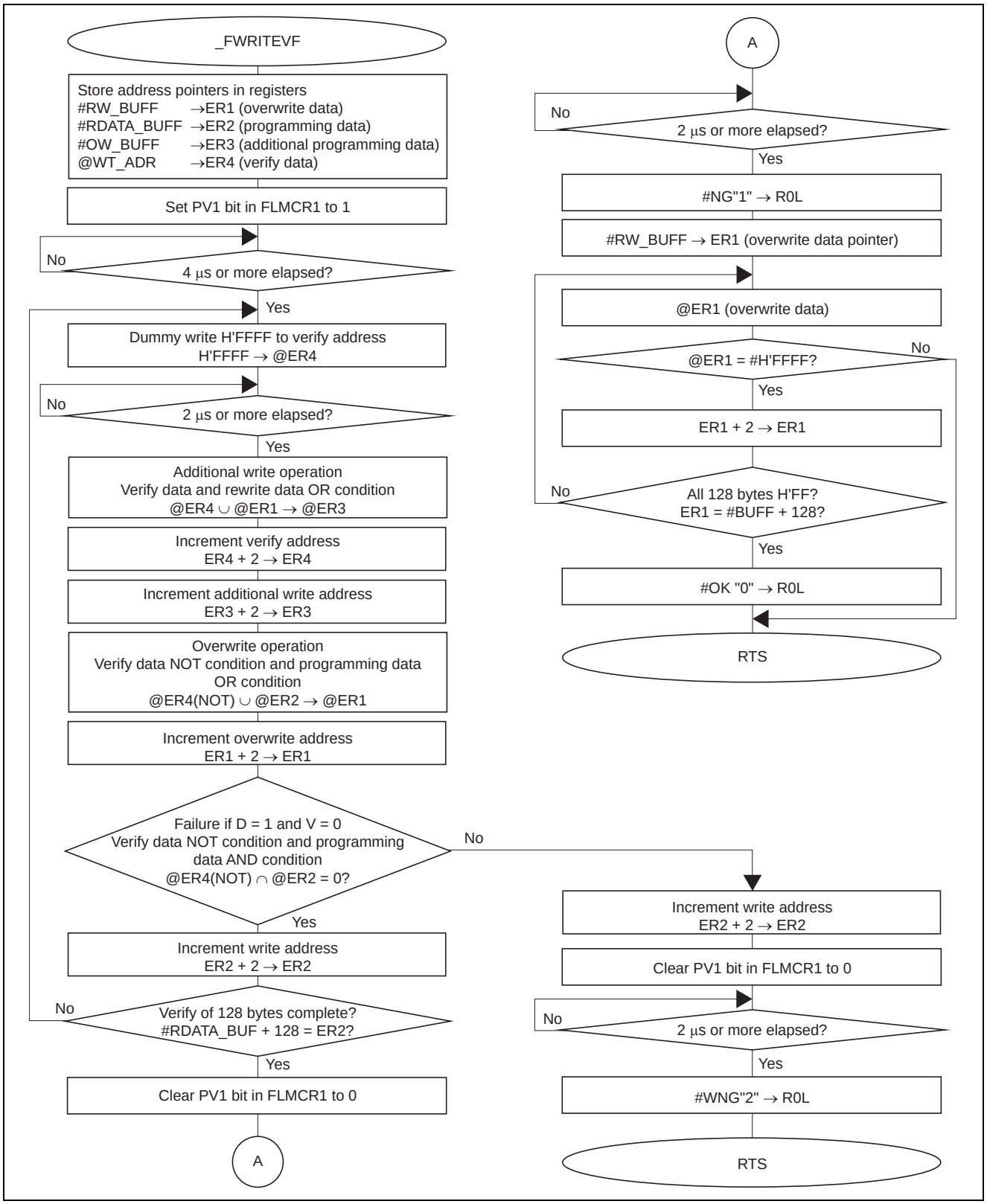

_BLK_CHECK
#FLMCR1 → ER6
#EBR1 → ER5
#BLOCK_ADR1 → ER0
#MAX_BLK1 → R4L
EB0 to EB7?
#CHK_ADR"H'8000 > ER3?
Yes
No
#EBR2 → ER5
#BLOCK_ADR2 → ER0
#MAX_BLK2 → R4L
ER0 + 4 → ER0
Erase start address?
@ER0 = ER3?
Yes
No
Increment block count
R4H + 1 R4H
R4H → @BLK_NO
ER1 → @EVF_ST
BLOCKADR → @EVF_ED
No
Erase maximum address?
R4L = R4H?
Yes
NG"1" → R0L
OK"0" → R0L
RTS

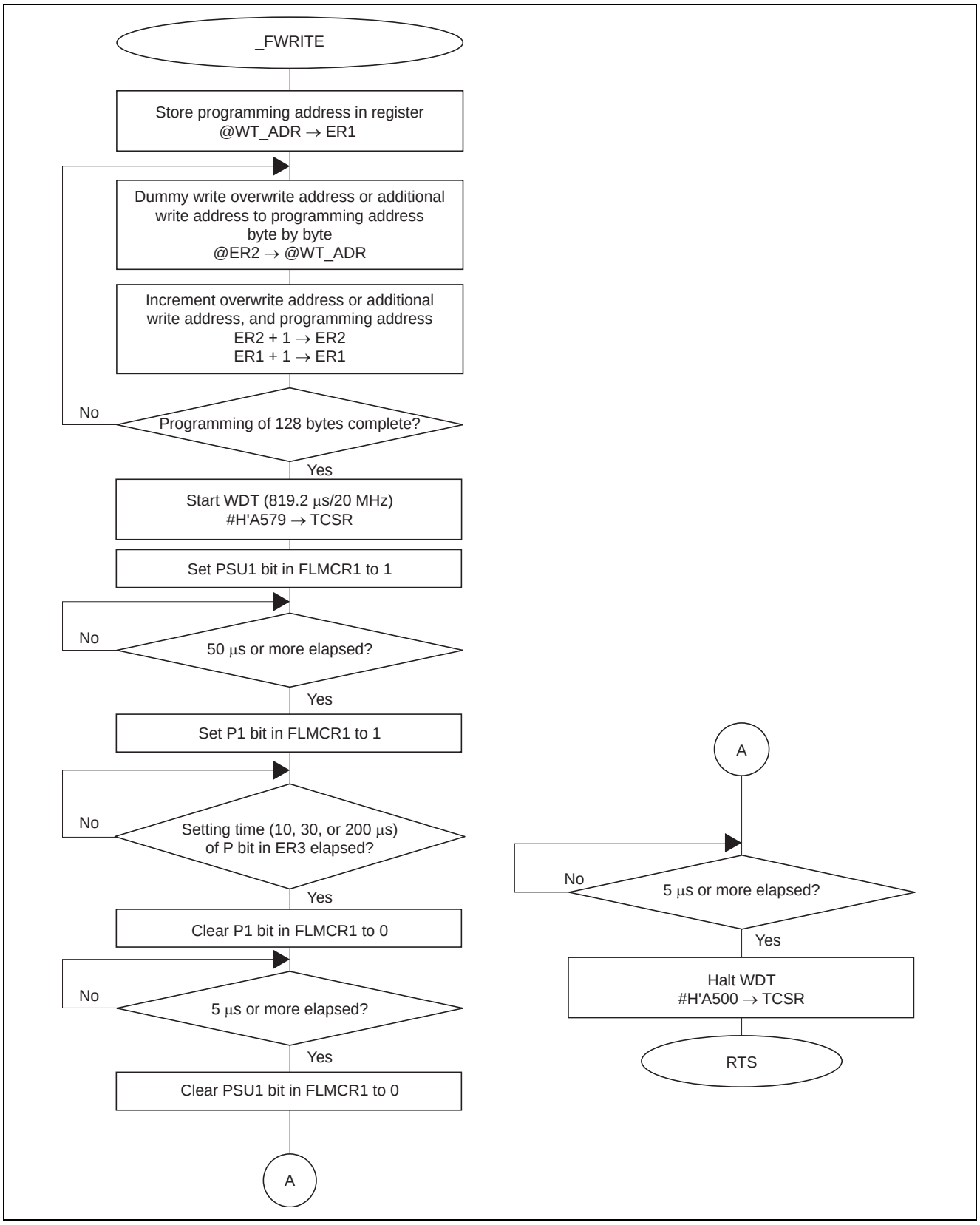

_BLK_ERASE
WDT initial setting
#H'5A5F → RSTCSR
Set SWE1 bit in FLMCR1 to 1
No
1 µs or more elapsed?
Yes
#0 → COUNT
_FERASEVF (initial verify)
R0L = OK"0"?
Yes
No
Erase and verify
_FERASE
_FERASEVF (verify)
R0L = OK"0"?
Yes
COUNT + 1 → COUNT
No
100 or more?
ET_COUNT = COUNT?
Abnormal end
Yes
Normal end
Clear SWE1 bit in FLMCR1 to 0
Clear SWE1 bit in FLMCR1 to 0
No
100 µs or more elapsed?
No
100 µs or more elapsed?
Yes
Yes
#NG"1" → R0L
#OK"0" → R0L
RTS
RTS


_FERASEVF
Store address pointers in registers
@EVF_ST → ER1 (verify start address)
@EVF_ED → ER2 (verify end address)
Set EV1 bit in FLMCR1 to 1
No
6 µs or more elapsed?
Yes
Dummy write H'FFFF to verify address
H'FFFF → @ER1
No
2 µs or more elapsed?
Yes
Erased?
@ER1 = H'FFFF?
No
Increment verify address
ER1 + 2 → ER1
No
Verify end address?
ER1 = ER2?
Yes
Clear EV1 bit in FLMCR1 to 0
No
4 µs or more elapsed?
Yes
#OK"0" → R0L
RTS
Clear EV1 bit in FLMCR1 to 0
No
4 µs or more elapsed?
Yes
#NG"1" → R1L
RTS


_FERASE
Set EBR erase target block bit to 1
@BLK_NO → EBR
Start up WDT (26 ms/20 MHz)
#H'A57C → TCSR
Set ESU1 bit in FLMCR1 to 1
No
100 µs or more elapsed?
Yes
Apply erase pulse
Set E1 bit in FLMCR1 to 1
No
10 ms elapsed?
Yes
Clear E1 bit in FLMCR1 to 0
No
10 µs or more elapsed?
Yes
Clear ESU1 bit in FLMCR1 to 0
No
10 µs or more elapsed?
Yes
Halt WDT
#H'A500 → TCSR
Clear EBR erase target block bit to 0
BLK_NO → EBR
RTS


_FWRITE_128
WDT initial setting
#H'5A5F → RSTCSR
Transfer received 128 bytes of programming data to overwrite area
@RDATA_BUF → @RW_BUFF
Set SWE1 bit in FLMCR1 to 1
No
1 µs or more elapsed?
Yes
#0 → @COUNT
_FWRITEVF (initial verify)
Yes
R0L = OK"0"?
No
B
Normal end
R0L = WNG"2"?
Yes
C
Abnormal end
No
#RW_BUFF (overwrite data) →ER2
#F_TIME_30 (30 µs) →ER3
_FWRITE (initial write)
_FWRITEVF
R0L → @VF_RET (verification result)
#OW_BUFF (additional programming data)
→ER2
#F_TIME_10 (10µs)
→ER3
_FWRITE (additional write)
Yes
@VF_RET = OK"0"?
No
B
Normal end
@VF_RET = WNG"2"?
Yes
C
Abnormal end
@COUNT + 1 → @COUNT
No
6 or more?
#OW_COUNT"6" = COUNT?
A
A
#RW_BUFF (overwrite data) →ER2
#F_TIME_200 (200 µs) →ER3
_FWRITE (6th or subsequent write)
_FWRITEVF
Yes
R0L = OK"0"?
No
B
Normal end
R0L = WNG"2"?
Yes
@COUNT + 1 → @COUNT
No
1000 or more
@WT_COUNT = COUNT?
C
Yes
Abnormal end
Clear SWE1 bit in FLMCR1 to 0
No
100 µs or more elapsed?
Yes
#NG"1" → R0L
RTS
Normal end
B
Clear SWE1 bit in FLMCR1 to 0
No
100 µs or more elapsed?
Yes
#OK"0" → R0L
RTS


_FWRITEVF
Store address pointers in registers
#RW_BUFF →ER1 (overwrite data)
#RDATA_BUFF →ER2 (programming data)
#OW_BUFF →ER3 (additional programming data)
@WT_ADR →ER4 (verify data)
Set PV1 bit in FLMCR1 to 1
No
4 µs or more elapsed?
Yes
Dummy write H'FFFF to verify address
H'FFFF → @ER4
No
2 µs or more elapsed?
Yes
Additional write operation
Verify data and rewrite data OR condition
@ER4 ∪ @ER1 → @ER3
Increment verify address
ER4 + 2 → ER4
Increment additional write address
ER3 + 2 → ER3
Overwrite operation
Verify data NOT condition and programming data
OR condition
@ER4(NOT) ∪ @ER2 → @ER1
Increment overwrite address
ER1 + 2 → ER1
Failure if D = 1 and V = 0
Verify data NOT condition and programming
data AND condition
@ER4(NOT) ∩ @ER2 = 0?
No
Yes
Increment write address
ER2 + 2 → ER2
No
Verify of 128 bytes complete?
#RDATA_BUF + 128 = ER2?
Yes
Clear PV1 bit in FLMCR1 to 0
A
A
No
2 µs or more elapsed?
Yes
#NG"1" → R0L
#RW_BUFF → ER1 (overwrite data pointer)
@ER1 (overwrite data)
No
@ER1 = #H'FFFF?
Yes
ER1 + 2 → ER1
No
All 128 bytes H'FF?
ER1 = #BUFF + 128?
Yes
#OK "0" → R0L
RTS
Increment write address
ER2 + 2 → ER2
Clear PV1 bit in FLMCR1 to 0
No
2 µs or more elapsed?
Yes
#WNG"2" → R0L
RTS


_FWRITE
Store programming address in register
@WT_ADR → ER1
Dummy write overwrite address or additional write address to programming address byte by byte
@ER2 → @WT_ADR
Increment overwrite address or additional write address, and programming address
ER2 + 1 → ER2
ER1 + 1 → ER1
No
Programming of 128 bytes complete?
Yes
Start WDT (819.2 µs/20 MHz)
#H'A579 → TCSR
Set PSU1 bit in FLMCR1 to 1
No
50 µs or more elapsed?
Yes
Set P1 bit in FLMCR1 to 1
No
Setting time (10, 30, or 200 µs) of P bit in ER3 elapsed?
Yes
Clear P1 bit in FLMCR1 to 0
No
5 µs or more elapsed?
Yes
Clear PSU1 bit in FLMCR1 to 0
A
A
No
5 µs or more elapsed?
Yes
Halt WDT
#H'A500 → TCSR
RTS

7. Description of Hardware

7.1 Data Transfer Format of IEBus

The IEBus data transfer format used in this sample task is shown in figure 18.

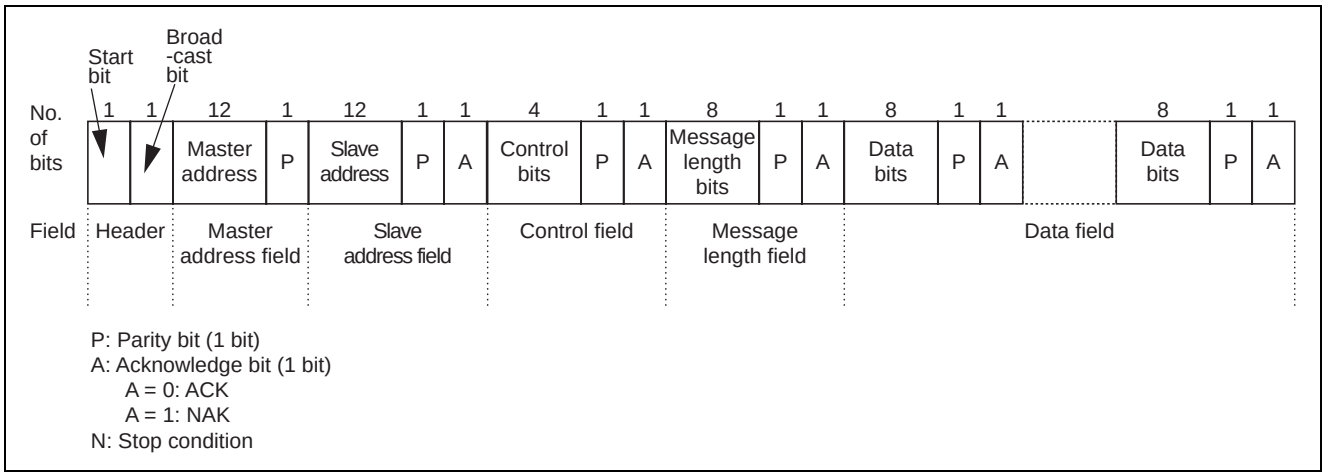
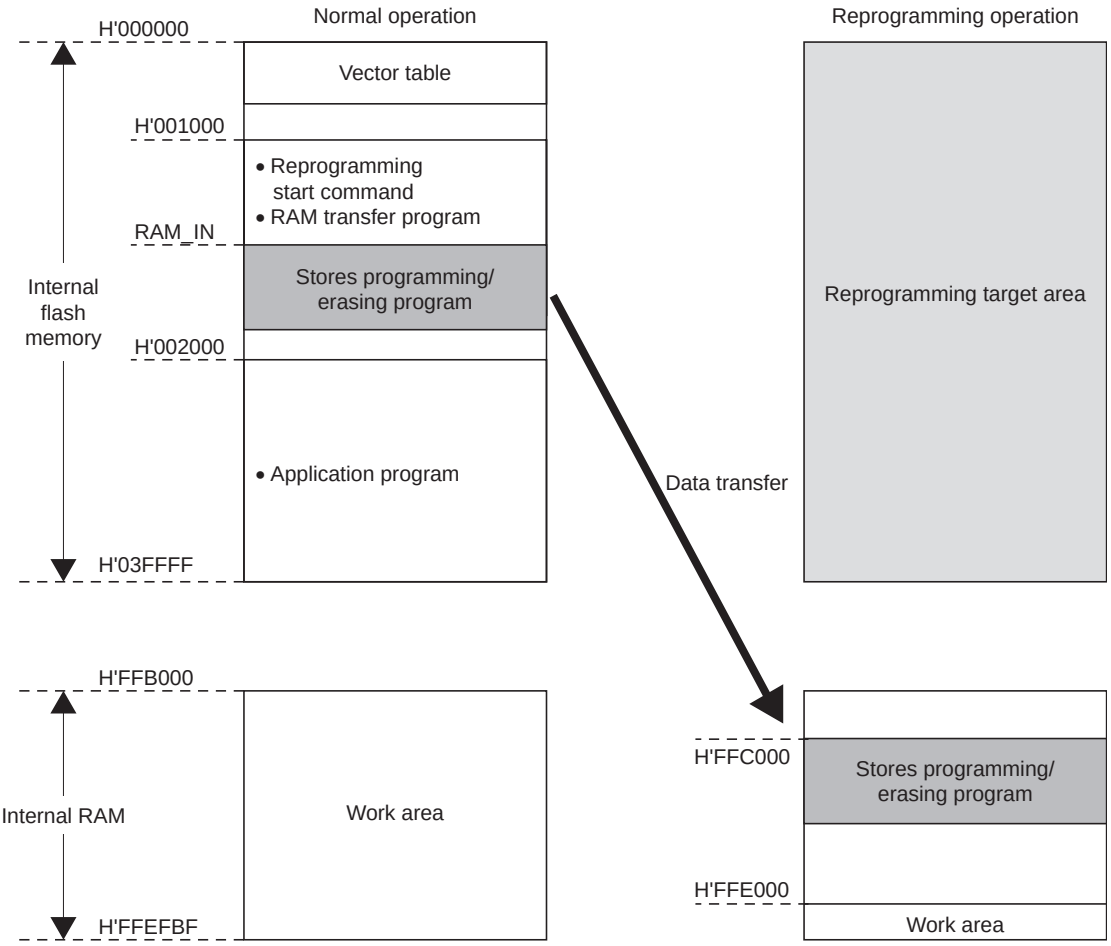


Figure 18 Data Transfer Format of IEBus

7.2 Memory Maps

Memory maps of the H8S/2258F during normal operation and during reprogramming operation is shown in figure 19.



Note: The programming/erasing program area, execution area, etc., shown in figure 19 represent a typical example.

Figure 19 Memory Maps

7.3 FWE Output/ Disabled

The output and disabled timing of the FWE signal when controlled by the programming/erasing program is shown in figure 20.

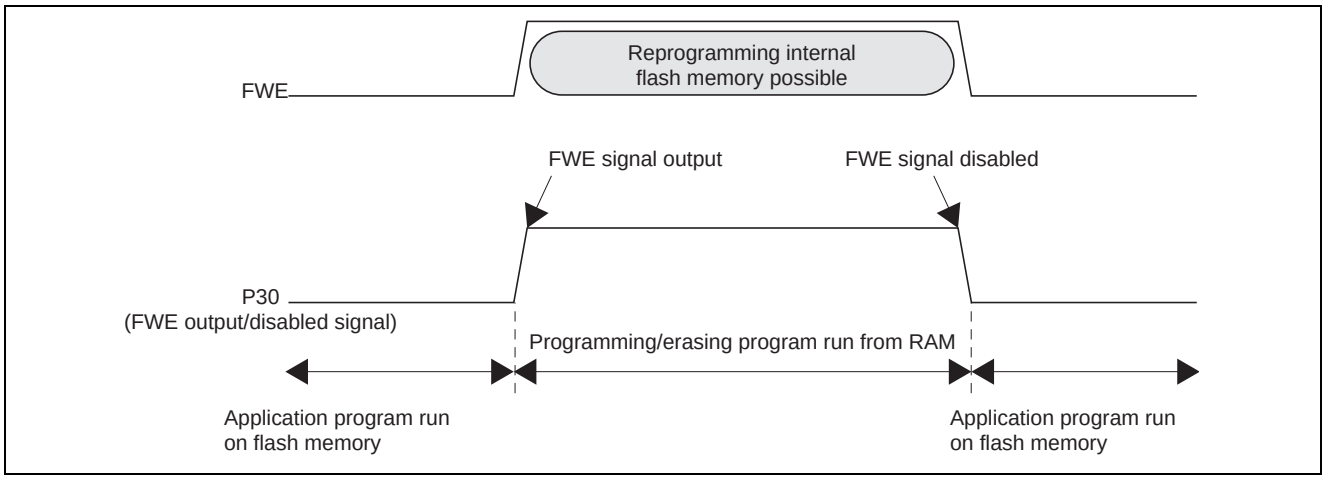
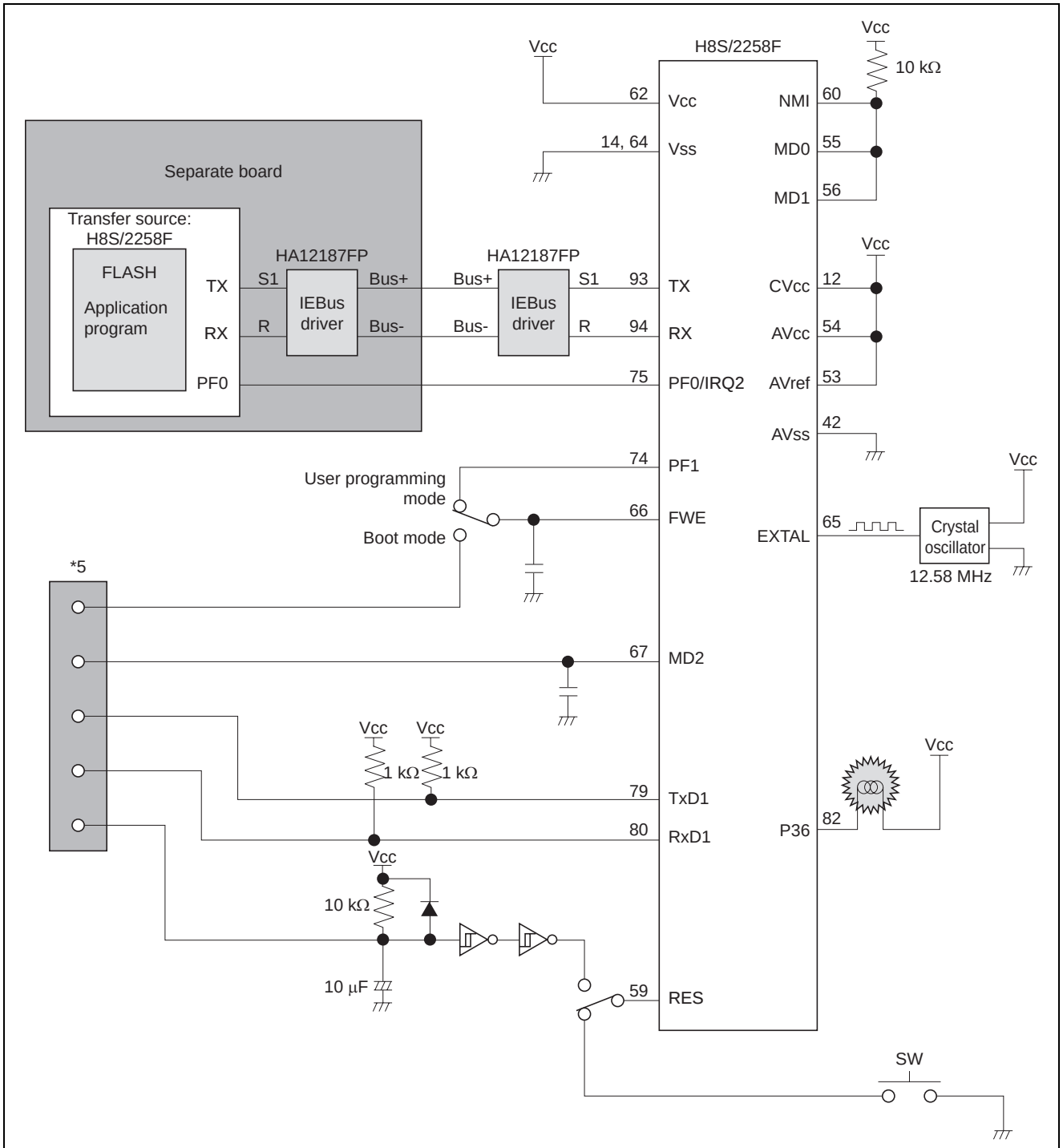


Figure 20 FWE Output/ Disabled Timing

8. Circuit Diagram



Note: Be sure to provide the pins required by the device's boot mode on the printed circuit board. This will ensure that if a write or an erase error occurs the device can be switched once again to the boot mode so that programming can be performed.

Figure 21 Diagram of On-Board Reprogramming Circuit Using the IEBus Interface

9. Program Listings

9.1 Slave Main Program

```
;---------------------------------------------------------------
;--   H8S/2258F internal IEBus controller                     --
;--   Flash rewrite program using data transmission from master --
;--   IEBus usage conditions (receiving side)                 --
;--   (1) System clock: 12.58 MHz                             --
;--   (2) Communication operating mode: Mode 2                --
;--   (3) Broadcast bit: Normal/broadcast communication       --
;---------------------------------------------------------------
        .cpu    2000A:24
 ;*****************************************
 ;*****  IEBus addresses                  *****
 ;*****************************************
 DH_BIT          .EQU    H'00                    ; Normal communication (H'00)/broadcast
                                                   communication (H'01)
 MADR_H          .EQU    H'BB                    ; Local address (upper 8 bits)
 MADR_L          .EQU    H'B0                    ; Local address (lower 4 bits)
 SADR_H          .EQU    H'AA                    ; Slave address (upper 8 bits)
 SADR_L          .EQU    H'A0                    ; Slave address (lower 4 bits)
 ;*****************************************
 ;*****  RAM transfer addresses           *****
 ;*****************************************
 PROG_ZERO       .EQU    H'00000000         ; Address 0
 PROG_TOP        .EQU    H'00001000         ; Start address of new application program
 PRESET_TOP      .EQU    H'00010000         ; Start address of reset vector
 IRQ2_TOP        .EQU    H'00012000         ; Start address of IRQ2 interrupt vector
 RAM_ST          .EQU    H'00FFC000         ; Start address of internal RAM transfer source
 RAM_TOP         .EQU    H'00FFB000         ; Start address of internal RAM
 RAM_END         .EQU    H'00FFEFBF         ; End address of internal RAM
 RAM_FINISH      .EQU    H'00FFEFC0         ; End address of internal RAM + 1
 RAM_BUFF        .EQU    H'00FFE000         ; Work area of RAM
 ;*****************************************
 ;*****  Output codes                     *****
 ;*****************************************
 OK              .EQU    H'00                    ; OK code
 NG              .EQU    H'01                    ; NG code
 E_ERR           .EQU    H'02                    ; Erase error code
 W_ERR           .EQU    H'03                    ; Flash write error code
 R_TM_ERR        .EQU    H'04                    ; Receive timing error code
 R_DB_ERR        .EQU    H'05                    ; Receive transfer byte error code
 R_P_ERR         .EQU    H'06                    ; Receive parity error code
 R_OR_ERR        .EQU    H'07                    ; Receive overrun error code
 T_AL_ERR        .EQU    H'08                    ; Transmit arbitration error code
 T_AR_ERR        .EQU    H'09                    ; Transmit underrun error code
 T_TM_ERR        .EQU    H'0A                    ; Transmit timing error code
 T_DB_ERR        .EQU    H'0B                    ; Transmit transfer byte error code
 T_AK_ERR        .EQU    H'0C                    ; Transmit ACK error code
 IRA_ERR         .EQU    H'0D                    ; Runaway error code
 Request         .EQU    H'11                    ; Transmit request code
 ;*****************************************
 ;*****  Commands        Data            *****
```

```
50    ;****************************************
51    W_E_go          .EQU    H'55                ; Rewrite start command
52    FWE_on          .EQU    H'66                ; FWE pin setting command
53    Erase_go        .EQU    H'77                ; Erase command
54    Write_go        .EQU    H'88                ; Write command
55    FWE_off         .EQU    H'99                ; Clear FWE pin command
56    ;****************************************
57    ;*****  Module stop registers          *****
58    ;****************************************
59    MSTPCRA         .EQU    H'00FFFDE8
60    DTC_STP         .EQU    6                   ; DTC module stop register
61    MSTPCRB         .EQU    H'00FFFDE9
62    MSTPCRC         .EQU    H'00FFFDEA
63    IE_STP          .EQU    3                   ; IEBus module stop register
64    ;****************************************
65    ;*****  Port registers                 *****
66    ;****************************************
67    P3DDR           .EQU    H'00FFFE32
68    PFDDR           .EQU    H'00FFFE3E
69    PGDDR           .EQU    H'00FFFE3F
70    P3DR            .EQU    H'00FFFF02
71    PFDR            .EQU    H'00FFFF0E
72    PGDR            .EQU    H'00FFFF0F
73    ;****************************************
74    ;*****  Interrupt registers            *****
75    ;****************************************
76    IER             .EQU    H'00FFFE14
77    ISR             .EQU    H'00FFFE15
78    ;****************************************
79    ;*****  Flash rewrite related          *****
80    ;****************************************
81    MAX_OW          .EQU    6                   ; Additional write count
82    MAX_ET          .EQU    100                 ; Maximum erase count
83    MAX_WT          .EQU    1000                ; Maximum write count
84    MAX_BLK1        .EQU    8                   ; Maximum block 1
85    MAX_BLK2        .EQU    4                   ; Maximum block 2
86    CHK_ADR         .EQU    H'00008000          ; Start address of target EBR2
87    SCRX            .EQU    H'00FFFDB4          ; Serial control register X
88    FLSHE           .EQU    3
89    FLMCR1          .EQU    H'00FFFFA8          ; Flash memory control register 1
90    FWE             .EQU    7
91    SWE             .EQU    6
92    ESU             .EQU    5
93    PSU             .EQU    4
94    EV              .EQU    3
95    PV              .EQU    2
96    E               .EQU    1
97    P               .EQU    0
98    FLMCR2          .EQU    H'00FFFFA9          ; Flash memory control register 2
99    FLER            .EQU    7
100    EBR1            .EQU    H'00FFFFAA          ; Erase block designation register 1
101    EBR2            .EQU    H'00FFFFAB          ; Erase block designation register 2
102    ;****************************************
103    ;****   IEBus registers                *****
```

```
104     ;****************************************
105     IECTR           .EQU    H'00FFF800    ; IEBus control register
106     IEE             .EQU    7
107     IOL             .EQU    6
108     DEE             .EQU    5
109     CKS             .EQU    4
110     RE              .EQU    3
111     LUEE            .EQU    2
112     IECMR           .EQU    H'00FFF801    ; IEBus control register
113     IEMCR           .EQU    H'00FFF802    ; IEBus master control register
114     SS              .EQU    7
115     IEAR1           .EQU    H'00FFF803    ; IEBus local address register 1
116     STE             .EQU    0
117     IEAR2           .EQU    H'00FFF804    ; IEBus local address register 2
118     IESA1           .EQU    H'00FFF805    ; IEBus slave address setting register 1
119     IESA2           .EQU    H'00FFF806    ; IEBus slave address setting register 2
120     IETBFL          .EQU    H'00FFF807    ; IEBus transmit message length register
121     IETBR           .EQU    H'00FFF808    ; IEBus transmit buffer register
122     IEMA1           .EQU    H'00FFF809    ; IEBus receive master address register 1
123     IEMA2           .EQU    H'00FFF80A    ; IEBus receive master address register 2
124     IERCTL          .EQU    H'00FFF80B    ; IEBus receive control field register
125     IERBFL          .EQU    H'00FFF80C    ; IEBus receive message length register
126     IERBR           .EQU    H'00FFF80D    ; IEBus receive buffer register
127     IELA1           .EQU    H'00FFF80E    ; IEBus lock address register 1
128     IELA2           .EQU    H'00FFF80F    ; IEBus lock address register 2
129     IEFLG           .EQU    H'00FFF810    ; IEBus general flag register
130     CMX             .EQU    7
131     MRQ             .EQU    6
132     SRQ             .EQU    5
133     SRE             .EQU    4
134     LCK             .EQU    3
135     RSS             .EQU    1
136     GG              .EQU    0
137     IETSR           .EQU    H'00FFF811    ; IEBus transmit/runaway status register
138     TxRDY           .EQU    7
139     IRA             .EQU    3
140     TxS             .EQU    2
141     TxF             .EQU    1
142     TxE             .EQU    0
143     IEIET           .EQU    H'00FFF812    ; IEBus transmit/runaway interrupt request register
144     TxRDYE          .EQU    7
145     IRAE            .EQU    3
146     TxSE            .EQU    2
147     TxFE            .EQU    1
148     TxEE            .EQU    0
149     IETEF           .EQU    H'00FFF813    ; IEBus transmit error flag register
150     AL              .EQU    4
151     UE              .EQU    3
152     TTME            .EQU    2
153     RO              .EQU    1
154     ACK             .EQU    0
155     IERSR           .EQU    H'00FFF814    ; IEBus receive status register
156     RxRDY           .EQU    7
157     RxS             .EQU    2
```

```
158     RxF             .EQU    1
159     RxE             .EQU    0
160     IEIER           .EQU    H'00FFF815    ; IEBus receive interrupt request register
161     RxRDYE          .EQU    7
162     RxSE            .EQU    2
163     RxFE            .EQU    1
164     RxEE            .EQU    0
165     IEREF           .EQU    H'00FFF816    ; IEBus receive error flag register
166     OVE             .EQU    3
167     RTME            .EQU    2
168     DLE             .EQU    1
169     PE              .EQU    0
170     ;*****************************************
171     ;*****  WDT registers                *****
172     ;*****************************************
173     TCSR1           .EQU    H'00FFFA2             ; WDT register 1
174     RSTCSR          .EQU    H'00FFFA4
175     ;*****************************************
176     ;*****  Wait times for               *****
177     ;*****  flash memory programming     *****
178     ;*****************************************
179     MHZ             .EQU    1350                  ; Operating frequency (13.5 MHz)
180     F_LOOP_1        .EQU        1*MHZ/400+1       ; LOOP WAIT TIME
181     F_LOOP_2        .EQU        2*MHZ/400+1
182     F_LOOP_4        .EQU        4*MHZ/400+1
183     F_LOOP_5        .EQU        5*MHZ/400+1
184     F_LOOP_6        .EQU        6*MHZ/400+1
185     F_LOOP_10       .EQU       10*MHZ/400+1
186     F_LOOP_30       .EQU       30*MHZ/400+1
187     F_LOOP_50       .EQU       50*MHZ/400+1
188     F_LOOP_100      .EQU      100*MHZ/400+1
189     F_TIME_10       .EQU       10*MHZ/400         ; Write wait time
190     F_TIME_30       .EQU       30*MHZ/400
191     F_TIME_200      .EQU      200*MHZ/400
192     F_TIME_10000    .EQU    10000*MHZ/400         ; Erase wait time
193     ;*****************************************************************
194     ;**     RAM transmit data                                       **
195     ;*****************************************************************
196             .SECTION        RAM,data,locate = RAM_BUFF
197     TRANS_BUFF:     .EQU    $
198     THEDDA_BUFF     .RES.B  1               ; Broadcast bit
199     SADR_BUFF_H     .RES.B  1               ; Slave address (upper 8 bits)
200     SADR_BUFF_L     .RES.B  1               ; Slave address (lower 4 bits)
201     TCONT_BUFF      .RES.B  1               ; Control bit
202     TDENBUN_BUFF    .RES.B  1               ; Message length bit
203     TDUMMY          .RES.B  1               ; DUMMY
204     TDATA_BUFF      .RES.B  128             ; Transmit data (max. 1,024 bits: 128 bytes)
205     ;*****************************************************************
206     ;**     RAM receive data                                        **
207     ;*****************************************************************
208     RECEIV_BUFF:    .EQU    $
209     RDENBUN_BUFF    .RES.B  1               ; Message length bit
210     IEBUS_ERROR     .RES.B  1               ; IEBus error variable
211     RDATA_BUFF      .RES.B  128             ; Receive data (max. 1,024 bits: 128 bytes)
```

```
212     ;*********************************************************************
213     ;**     RAM work area                                               **
214     ;*********************************************************************
215             .ALIGN  2
216     WT_ADR          .RES.B  4               ; Write address
217     RW_BUFF         .RES.B  128             ; Overwrite data
218     OW_BUFF         .RES.B  128             ; Additional programming data
219     COUNT           .RES.B  2               ; E_COUNT, W_COUNT
220     ET_COUNT        .RES.B  2               ; Maximum E_COUNT
221     WT_COUNT        .RES.B  2               ; Maximum W_COUNT
222     REST_SIZE       .RES.B  4               ; Programming data size
223             .ALIGN  2
224     EVF_ST          .RES.B  4               ; Erase verify start address
225     EVF_ED          .RES.B  4               ; Erase verify end address
226     BLK_NO          .RES.B  1               ; Erase block number
227     VF_RET          .RES.B  1               ; Verify
228     ER_COUNT        .RES.B  1               ; Erase count
229     ;######################################################################
230     ;##  Vector table                                                    ##
231     ;######################################################################
232             .SECTION        VECT,code,locate = PROG_ZERO
233             .DATA.L         _PRESET                 ; Reset vector
234             .ORG            H'001C
235             .DATA.L         _IRQ2                   ; IRQ2 interrupt vector
236     ;######################################################################
237     ;##  Main routine                                                    ##
238     ;######################################################################
239             .SECTION        PROG_R,code,locate = PRESET_TOP
240     _PRESET:        .EQU    $
241             MOV.L   #RAM_FINISH,    ER7             ; Set stack pointers
242             BCLR.B  #2,             @ISR            ; Clear IRQ2 flag
243             BCLR.B  #3,             @ISR            ; Clear IRQ3 flag
244             MOV.B   #H'0C,          R0L
245             MOV.B   R0L,            @IER            ; Enable IRQ2 and ORQ3 interrupts
246             ANDC.B  #H'7F,          CCR             ; Enable interrupts
247     ;######################################################################
248     ;##  Reset routine                                                   ##
249     ;######################################################################
250     PRESET_LOOP:
251             BRA     PRESET_LOOP
252     ;######################################################################
253     ;##  Flash programming routine                                       ##
254     ;######################################################################
255             .SECTION        PROG_2,code,locate = IRQ2_TOP
256     _IRQ2:          .EQU    $
257             BSR     _IO_INT                 ; Initialize transmit/receive pins
258             BSR     _IE_INT                 ; Initialize IEBus
259             BSR     _IE_TRS_CMD             ; To IEBus flash programming/erasing processing
260     IRQ2_LOOP:
261             BRA     IRQ2_LOOP               ; Infinite loop
262     ;######################################################################
263     ;##  _IO_INT    (Initialize transmit/receive pins)                   ##
264     ;######################################################################
265     _IO_INT:                .EQU    $
```

```
266             MOV.B   #H'3F,          R0L
267             MOV.B   R0L,            @P3DDR          ; Set P30 to P35 to output
268             BCLR.B  #2,             @PGDR           ; Fix TX pin in idle state
269             MOV.B   #H'FB,          R0L
270             MOV.B   R0L,            @PGDDR          ; Set TX pin to output
271             RTS
272     ;#####################################################################
273     ;##  _IE_INT    (Initialize IEBus)                                  ##
274     ;#####################################################################
275     _IE_INT:                .EQU    $
276             BCLR.B  #IE_STP,        @MSTPCRC    ; Clear module stop mode for IEBus controller
277             MOV.B   #H'28,          R0L         ; Enable receive operation
278             MOV.B   R0L,            @IECTR      ; Settings for master transmission
279             MOV.B   #H'F7,          R0L
280             MOV.B   R0L,            @IEMCR      ; Control field (data read-in)
281             MOV.B   #(MADR_L | H'08),R0L        ; Local address
282             MOV.B   R0L,            @IEAR1      ; Mode 2
283             MOV.B   #MADR_H,        R0L
284             MOV.B   R0L,            @IEAR2
285             MOV.B   #H'87,          R0L
286             MOV.B   R0L,            @IEIER
287             MOV.B   #H'8F,          R0L
288             MOV.B   R0L,            @IEIET
289             BSET.B  #IEE,           @IECTR      ; IEBus operation start
290             RTS
291     ;#####################################################################
292     ;##  _IE_TRS_CMD(flash programming/erasing using IEBus)             ##
293     ;##===================================================================##
294     ;##     Rewrite start command       : H'55                          ##
295     ;##     RAM transfer program          : _Trace_to_RAM               ##
296     ;##     FWE pin setting command        : H'66                       ##
297     ;##     Erase command               : H'77                          ##
298     ;##     Write command           : H'88                              ##
299     ;##     Clear FWE pin command        : H'99                        ##
300     ;#####################################################################
301     _IE_TRS_CMD:    .EQU    $
302             JSR     @_IE_RECEIV                 ; <===== Receive (command: 55/00)
303             MOV.B   @IEBUS_ERROR,   R0L
304             BNE     NG_55                       ; In case of error, infinite loop
305
306             MOV.B   @RDATA_BUFF,    R0H
307             MOV.B   #W_E_go,        R5H
308             CMP.B   R0H,            R5H         ; Rewrite start command?
309             BNE     NG_55                       ; If other than rewrite start command,
                                                      error handling
310
311             JMP     @_Trace_to_RAM              ; To RAM transfer processing
312     NG_55:
313             JSR     @_RXE_SUB                   ; To error handling
314             MOV.B   #H'00,          R4L
315             MOV.W   R4,             @TDATA_BUFF ; R4H: NG code/R4L: 00
316             MOV.B   #H'02,          R3L         ; Transmit 2 bytes (R4)
317             JSR     @_IE_TRANS                  ; =====> Transmit (NG)
318             RTE
```

```
319     ;#####################################################################
320     ;##  _Trace_to_RAM      (RAM transfer)                            ##
321     ;#####################################################################
322     _Trace_to_RAM:  .EQU    $
323             MOV.L   #RAM_IN,        ER5          ; Start address of program to be transferred
324             MOV.L   #RAM_ST,        ER6          ; Start address of internal RAM transfer
                                                       target
325             MOV.W   #PROG_END-RAM_IN,R4          ; Size of program to be transferred
326             EEPMOV.W                             ; Transfer control program
327             JMP     @RAM_ST                      ; To internal RAM control program
328
329     ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
330     ;$$  _RAM_IN    (RAM to which data is transferred)                 $$
331     ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
332     RAM_IN: .EQU    $
333             SUB.L   ER4,            ER4          ; Initialize transmit buffer
334     ;====================================================================
335     ;==  _GO_W_E    (rewrite OK)                                       ==
336     ;====================================================================
337     _GO_W_E:                .EQU    $
338             MOV.B   #OK,            R4H
339             MOV.W   R4,             @TDATA_BUFF  ; R4H: OK code/R4L: 00
340             MOV.B   #H'02,          R3L          ; Transmit 2 bytes (R4)
341             BSR     _IE_TRANS                    ; =====> Transmit (OK)
342     ;====================================================================
343     ;==  _GO_FWE_on (set FWE pin)                                       ==
344     ;==--------------------------------------------------------------==
345     ;==     FWE pin setting command        : H'66                      ==
346     ;====================================================================
347     _GO_FWE_on:     .EQU    $
348             BSR     _IE_RECEIV                   ; <===== Receive (command: 66/00)
349             MOV.B   @IEBUS_ERROR,   R0L
350             BNE     NG_66                        ; In case of error, error handling
351
352             MOV.B   @RDATA_BUFF,    R0H
353             MOV.B   #FWE_on,        R5H
354             CMP.B   R0H,            R5H          ; FWE pin setting command?
355             BNE     NG_66                        ; If other than FWE pin setting command,
                                                       error handling
356
357             BSET.B  #1,             @PFDR        ; Set PF1 pin to 1
358             MOV.B   #H'02,          R0L
359             MOV.B   R0L,            @PFDDR       ; Set PF1 pin to output
360
361             BSET.B  #FLSHE,         @SCRX        ; Flash enable register
362
363             MOV.B   #OK,            R4H
364             MOV.W   R4,             @TDATA_BUFF  ; R4H: OK code/R4L: 00
365             MOV.B   #H'02,          R3L          ; Transmit 2 bytes (R4)
366             BSR     _IE_TRANS                    ; =====> Transmit (OK)
367
368             BRA     _GO_Erase
369     NG_66:
370             BSR     _RXE_SUB                     ; To error handing
```

```
371             MOV.W   R4,             @TDATA_BUFF     ; R4H: NG code/R4L: 00
372             MOV.B   #H'02,          R3L             ; Transmit 2 bytes (R4)
373             BSR     _IE_TRANS                       ; =====> Transmit (NG)
374             RTE
375     ;===================================================================
376     ;==  _GO_Erase  (erase flash memory)                              ==
377     ;==-----------------------------------------------------------------==
378     ;==     Erase command              : H'77                        ==
379     ;===================================================================
380     _GO_Erase:      .EQU    $
381             BSR     _IE_RECEIV                  ; <===== Receive (block count/command: 7700)
382             MOV.B   @IEBUS_ERROR,   R0L
383             BNE     NG_77                       ; In case of error, error handling
384
385             MOV.L   @RDATA_BUFF,    ER0         ; E0: erase count/R0H: command/R0L: 00
386             MOV.W   E0,             R1
387             MOV.B   R1L,            @ER_COUNT   ; Erase count
388
389             MOV.B   #Erase_go,      R5H
390             CMP.B   R0H,            R5H         ; Erase command?
391             BNE     NG_77                       ; If other than erase command, error handling
392
393             MOV.B   #OK,            R4H
394             MOV.W   R4,             @TDATA_BUFF ; R4H: OK code/R4L: 00
395             MOV.B   #H'02,          R3L         ; Transmit 2 bytes (R4)
396             BSR     _IE_TRANS                   ; =====> Transmit (OK)
397
398             MOV.B   @ER_COUNT,      R1L
399             CMP.B   #0,             R1L         ; Erase count
400             BEQ     _GO_Write                   ; Erase block = 0: program flash memory
401
402             MOV.W   #MAX_ET,        R0          ; Set maximum erase count
403             MOV.W   R0,             @ET_COUNT
404     NEXT_BLOCK:
405             BSR     _IE_RECEIV                  ; <===== Receive (block)
406             MOV.B   @IEBUS_ERROR,   R0L
407             BNE     NG_77                       ; In case of error, error handling
408
409             MOV.L   @RDATA_BUFF,    ER3         ; ER3: designated erase block
410             BSR     _BLK_CHECK                  ; Erase block check
411             CMP.B   #OK,            R0L         ; Compare _BLK_CHECK code
412             BNE     NG_77E                      ; In case of error, error handling
413
414             BSR     _BLK_ERASE                  ; Erase flash memory
415             CMP.B   #OK,            R0L         ; Compare _BLK_ERASE code
416             BNE     NG_77E                      ; In case of error, error handling
417
418             MOV.B   @ER_COUNT,      R3L
419             DEC.B   R3L                         ; Erase count – 1
420             MOV.B   R3L,            @ER_COUNT
421             CMP.B   #0,             R3L
422             BEQ     END_BLOCK                   ; Erase end
423
424             MOV.B   #Request,       R4H
```

```
425             BRA     ER_TRANS
426     END_BLOCK:
427             MOV.B   #OK,            R4H
428     ER_TRANS:
429             MOV.L   @EVF_ST,        ER0
430             MOV.W   R0,             E4          ; E4 :@EVF_ST
431             MOV.B   @BLK_NO,        R4L
432             MOV.L   ER4,            @TDATA_BUFF ; R4H: OK code/R4L: @BLK_NO
433             MOV.B   #H'04,          R3L         ; Transmit 2 bytes (R4)
434             BSR     _IE_TRANS                   ; =====> Transmit (OK)
435
436             MOV.B   @ER_COUNT,      R3L
437             CMP.B   #0,             R3L
438             BNE     NEXT_BLOCK                  ; Set next erase count
439             BRA     _GO_Write
440     NG_77:
441             BSR     _RXE_SUB                    ; To error handling
442             MOV.W   R4,             @TDATA_BUFF ; R4H:NG/R4L:00
443             MOV.B   #H'02,          R3L         ; Transmit 2 bytes (R4)
444             BSR     _IE_TRANS                   ; =====> Transmit (NG)
445             RTE
446     NG_77E:
447             MOV.B   R0L,            R4L
448             MOV.B   #E_ERR,         R4H
449             MOV.W   R4,             @TDATA_BUFF ; R4H: erase error code/R4L: 01
450             MOV.B   #H'02,          R3L         ; Transmit 2 bytes (R4)
451             BSR     _IE_TRANS                   ; =====> Transmit (erase error)
452             RTE
453     ;=====================================================================
454     ;==  _GO_Write  (program flash memory)                              ==
455     ;==-----------------------------------------------------------------==
456     ;==     Write command           : H'88                              ==
457     ;=====================================================================
458     _GO_Write:      .EQU    $
459             BSR     _IE_RECEIV                  ; <===== Receive (command: 88/00)
460             MOV.B   @IEBUS_ERROR,   R0L
461             BNE     NG_88                       ; In case of error, error handling
462
463             MOV.B   @RDATA_BUFF,    R0H
464             MOV.B   #Write_go,      R5H
465             CMP.B   R0H,            R5H         ; Write command?
466             BNE     NG_88                       ; If other than write command, error handling
467
468             MOV.B   #OK,            R4H
469             MOV.W   R4,             @TDATA_BUFF ; R4H: OK code/R4L: 00
470             MOV.B   #H'02,          R3L         ; Transmit 2 bytes (R4)
471             BSR     _IE_TRANS                   ; =====> Transmit (OK)
472
473             BSR     _IE_RECEIV                  ; <===== Receive (address)
474             MOV.B   @IEBUS_ERROR,   R0L
475             BNE     NG_88                       ; In case of error, error handling
476
477             MOV.L   @RDATA_BUFF,    ER0
478             MOV.L   ER0,            @WT_ADR     ; Write start address
```

```
479
480             MOV.B   #H'00,          R4L
481             MOV.B   #OK,            R4H
482             MOV.W   R4,             @TDATA_BUFF ; R4H: OK code/R4L: 00
483             MOV.B   #H'02,          R3L         ; Transmit 2 bytes (R4)
484             BSR     _IE_TRANS                   ; =====> Transmit (OK)
485
486             BSR     _IE_RECEIV                  ; <===== Receive (number of bytes)
487             MOV.B   @IEBUS_ERROR,   R0L
488             BNE     NG_88                       ; In case of error, error handling
489
490             MOV.L   @RDATA_BUFF,    ER0
491             MOV.L   ER0,            @REST_SIZE  ; Write byte count
492
493             MOV.W   #MAX_WT,        R1          ; Set maximum write count
494             MOV.W   R1,             @WT_COUNT
495     NEXT_DATA:
496             MOV.B   #H'00,          R4L
497             MOV.B   #Request,       R4H
498             MOV.W   R0,             E4
499             MOV.L   ER4,            @TDATA_BUFF ; E4H: byte count/R4H: request code/R4L: 00
500             MOV.B   #H'04,          R3L         ; Transmit 4 bytes (ER4)
501             BSR     _IE_TRANS                   ; =====> Transmit (request)
502
503             BSR     _IE_RECEIV                  ; <===== Receive (application data)
504             MOV.B   @IEBUS_ERROR,   R0L
505             BNE     NG_88                       ; In case of error, error handling
506
507             BSR     _FWRITE_128                 ; Program flash memory
508             CMP.B   #OK,            R4L
509             BNE     NG_88
510
511             MOV.L   @WT_ADR,        ER1
512             ADD.L   #128,           ER1         ; Address + 128
513             MOV.L   ER1,            @WT_ADR     ; Set next address
514
515             MOV.L   @REST_SIZE,     ER0
516             SUB.L   #128,           ER0         ; Transfer size – 128
517             MOV.L   ER0,            @REST_SIZE  ; Set next transfer size
518             CMP.L   #128,           ER0
519             BLS     END_DATA                    ; Is transfer size 128 or less?
520             BRA     NEXT_DATA
521     END_DATA:
522             MOV.B   #H'00,          R4L
523             MOV.B   #Request,       R4H
524             MOV.W   R0,             E4
525             MOV.L   ER4,            @TDATA_BUFF ; E4: byte count/R4H: request code/R4L: 00
526             MOV.B   #H'04,          R3L         ; Transmit 4 bytes (ER4)
527             BSR     _IE_TRANS                   ; =====> Transmit (request)
528
529             BSR     _IE_RECEIV                  ; <===== Receive (application data)
530             MOV.B   @IEBUS_ERROR,   R0L
531             BNE     NG_88                       ; In case of error, error handling
532
```

```
533             BSR     _FWRITE_128                 ; Program flash memory
534             CMP.B   #OK,            R4L
535             BNE     NG_88
536     OK_88:
537             MOV.B   #OK,            R4H
538             MOV.W   R4,             @TDATA_BUFF ; R4H: OK code/R4L: 00
539             MOV.B   #H'02,          R3L         ; Transmit 2 bytes (R4)
540             BSR     _IE_TRANS                   ; =====> Transmit (OK)
541             BRA     _GO_FWE_off
542     NG_88:
543             BSR     _RXE_SUB                    ; To error handling
544             MOV.W   R4,             @TDATA_BUFF ; R4H: NG code/R4L: 00
545             MOV.B   #H'02,          R3L         ; To error handling
546             BSR     _IE_TRANS                   ; =====> Transmit (NG)
547             RTE
548     ;===================================================================
549     ;==  _GO_FWE_off        (clear FWE pin)                          ==
550     ;==-----------------------------------------------------------------==
551     ;==     Clear FWE pin command        : H'99                       ==
552     ;===================================================================
553     _GO_FWE_off:    .EQU    $
554             BSR     _IE_RECEIV                      ; <===== Receive (command: 99/00)
555             MOV.B   @IEBUS_ERROR,   R0L
556             BNE     NG_99                           ; In case of error, error handling
557
558             MOV.B   @RDATA_BUFF,    R0H
559             MOV.B   #FWE_off,       R5H
560             CMP.B   R0H,            R5H             ; Clear FWE pin command?
561             BNE     NG_99                           ; If other than clear FWE pin command,
                                                          error handling
562
563             BCLR.B  #1,             @PFDR           ; Clear PF1 pin to 0
564             MOV.B   #H'00,          R0L
565             MOV.B   R0L,            @PFDDR          ; Set PF1 pin to input
566
567             MOV.B   #H'BF,          R0H
568             MOV.B   R0H,            @P3DR
569             MOV.B   #H'40,          R0L
570             MOV.B   R0L,            @P3DDR          ; LED(•_ooooo)
571
572             MOV.B   #OK,            R4H
573             MOV.W   R4,             @TDATA_BUFF     ; R4H: OK code/R4L: 00
574             MOV.B   #H'02,          R3L             ; Transmit 2 bytes (R4)
575             BSR     _IE_TRANS                       ; =====> Transmit (OK)
576
577             BRA     _RAM_OUT
578     NG_99:
579             BSR     _RXE_SUB                        ; To error handling
580             MOV.W   R4,             @TDATA_BUFF     ; R4H: NG code/R4L: 00
581             MOV.B   #H'02,          R3L             ; Transmit 2 bytes (R4)
582             BSR     _IE_TRANS                       ; =====> Transmit (NG)
583             RTE
584     ;===================================================================
585     ;==  _RAM_OUT   (new application program)                         ==
```

```
586     ;=====================================================================
587     _RAM_OUT:               .EQU    $
588             JMP     @PROG_TOP                       ; To new application in flash memory
589
590     ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
591     ;$$     End transfer to RAM                                             $$
592     ;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
593
594
595     ;*********************************************************************
596     ;**  _IE_RECEIV (receive 1 frame)                                    **
597     ;*********************************************************************
598     _IE_RECEIV:     .EQU    $
599             SUB.L   ER0,            ER0             ; Clear ER0
600     RECEIV_WAIT:
601             BTST.B  #RxS,           @IERSR          ; Verify start of reception by slave
602             BEQ     RECEIV_WAIT
603
604             BCLR.B  #RxS,           @IERSR          ; Detect start of reception by slave
605             MOV.B   @IERBFL,        R0L
606             MOV.B   R0L,            @RDENBUN_BUFF ; Receive data size
607
608             MOV.L   #RDATA_BUFF,    ER1             ; Receive data start address
609     IE_RCV_DATA:
610             BTST.B  #RxE,           @IERSR
611             BNE     IE_RCV_ERR                      ; To error handling routine
612             BTST.B  #IRA,           @IETSR
613             BNE     IE_RCV_IRA                      ; To runaway processing routine
614             BTST.B  #RxRDY,         @IERSR
615             BEQ     IE_RCV_DATA                     ; Wait for RxRDY flag 1 (reception complete)
616
617             MOV.B   @IERBR,         R0H
618             MOV.B   R0H,            @ER1            ; RDATA_BUFF
619             INC.L   #1,             ER1             ; RDATA_BUFF +1
620
621             BCLR.B  #RxRDY,         @IERSR          ; Clear RxRDY (receive)
622             DEC.B   R0L                             ; Count – 1
623             CMP.B   #0,             R0L
624             BNE     IE_RCV_DATA                     ; If receive byte count = 0, end
625
626             MOV.B   #H'00,          R0H
627             MOV.B   R0H,            @IEBUS_ERROR  ; IEBUS_ERROR
628     IE_RCV_RET:
629             BTST.B  #RxE,           @IERSR
630             BNE     IE_RCV_ERR                      ; To error handling routine
631             BTST.B  #IRA,           @IETSR
632             BNE     IE_RCV_IRA                      ; To runaway processing routine
633             BTST.B  #RxF,           @IERSR          ; Reception complete?
634             BEQ     IE_RCV_RET
635             BRA     IE_RCV_END
636     IE_RCV_ERR:
637             BSR     _RXE_SUB                        ; Error handling routine
638             BRA     IE_RCV_END
639     IE_RCV_IRA:
```

```
640             BSR     _IRA_SUB                        ; Runaway processing routine
641     IE_RCV_END:
642             BCLR.B  #RxF,           @IERSR          ; Receive end
643             RTS
644     ;***************************************************************
645     ;**  _IE_TRANS  (transmit command)                             **
646     ;**=============================================================**
647     ;**     R4H             : Transmit data                        **
648     ;***************************************************************
649     _IE_TRANS:      .EQU    $
650             MOV.B   #DH_BIT,        R0L
651             MOV.B   R0L,            @THEDDA_BUFF    ; Broadcast bit
652             MOV.B   #SADR_H,        R0L
653             MOV.B   R0L,            @SADR_BUFF_H    ; Slave address (high)
654             MOV.B   #SADR_L,        R0L
655             MOV.B   R0L,            @SADR_BUFF_L    ; Slave address (low)
656             MOV.B   #H'0F,          R0L
657             MOV.B   R0L,            @TCONT_BUFF     ; Control field (data write)
658             MOV.B   R3L,            @TDENBUN_BUFF   ; Transmit message length n bytes
659
660             MOV.L   #TRANS_BUFF,    ER1             ; Start address of transmit frame
661             MOV.B   @ER1+,          R0L             ; Determine normal or broadcast
662             BEQ     TRS_SETUP_NR
663
664             BCLR.B  #SS,            @IEMCR          ; Set to broadcast communication
665             BRA     TRS_SETUP_DH
666     TRS_SETUP_NR:
667             BSET.B  #SS,            @IEMCR          ; Set to normal communication
668     TRS_SETUP_DH:                                   ; Set transfer destination slave address
669             MOV.B   @ER1+,          R0L
670             MOV.B   R0L,            @IESA2          ; Set slave address
671
672             MOV.B   @ER1+,          R0L
673             MOV.B   R0L,            @IESA1          ; Set slave address
674
675             MOV.B   @IEMCR,         R0L
676             MOV.B   #H'F0,          R0H
677             AND.B   R0H,            R0L
678             MOV.B   @ER1+,          R0H
679             OR.B    R0H,            R0L
680             MOV.B   R0L,            @IEMCR          ; Set control field
681
682             MOV.B   @ER1+,          R0H
683             MOV.B   R0H,            @IETBFL         ; Set transmit message length
684
685             ADDS    #1,             ER1             ; debug
686
687             MOV.B   @ER1+,          R0L
688             MOV.B   R0L,            @IETBR          ; Set data
689
690             BCLR.B  #TxRDY,         @IETSR          ; Clear TxRDY (transmit)
691     IE_TRS_CMD:
692             BTST.B  #CMX,           @IEFLG
693             BNE     IE_TRS_CMD                      ; Wait for CMX flag 0
```

```
694
695             MOV.B   #H'02,          R0L         ; Request data transfer as master
696             MOV.B   R0L,            @IECMR
697     IE_TRS_ST:
698             BTST.B  #TxE,           @IETSR
699             BNE     IE_TRS_ERR                  ; To error handling routine
700             BTST.B  #IRA,           @IETSR
701             BNE     IE_TRS_IRA                  ; To runaway processing routine
702             BTST.B  #TxS,           @IETSR      ; Verify start of transmission by master
703             BEQ     IE_TRS_ST
704
705             BCLR.B  #TxS,           @IETSR      ; Detect start of transmission by master
706     IE_TRS_DATA:
707             BTST.B  #TxE,           @IETSR
708             BNE     IE_TRS_ERR                  ; To error handling routine
709             BTST.B  #IRA,           @IETSR
710             BNE     IE_TRS_IRA                  ; To runaway processing routine
711             BTST.B  #TxRDY,         @IETSR
712             BEQ     IE_TRS_DATA                 ; Wait for TxRDY flag 1 (transmission complete)
713
714             DEC.B   R0H                         ; Counter value – 1
715             CMP.B   #0,             R0H
716             BEQ     IE_TRS_FINISH               ; If counter value = 0, branch
717
718             MOV.B   @ER1+,          R0L
719             MOV.B   R0L,            @IETBR      ; Set next data
720
721             BCLR.B  #TxRDY,         @IETSR      ; Clear TxRDY (transmit next data)
722             BRA     IE_TRS_DATA
723     IE_TRS_FINISH:
724             BTST.B  #TxF,           @IETSR      ; Transmission complete?
725             BEQ     IE_TRS_FINISH
726             BRA     IE_TRS_END
727     IE_TRS_ERR:
728             BSR     _TXE_SUB                    ; Transmit error
729             BCLR.B  #TxF,           @IETSR      ; Transmit end
730             RTE
731     IE_TRS_IRA:
732             BSR     _IRA_SUB                    ; Runaway
733             BCLR.B  #TxF,           @IETSR      ; Transmit end
734             RTE
735     IE_TRS_END:
736             BCLR.B  #TxF,           @IETSR      ; Transmit end
737             RTS
738     ;*****************************************************************
739     ;**  _RXE_SUB   (slave receive error processing)                 **
740     ;**=============================================================**
741     ;**     H'04            : Timing error                           **
742     ;**     H'05            : Transfer byte error                    **
743     ;**     H'06            : Parity error                           **
744     ;**     H'07            : Overrun error                          **
745     ;*****************************************************************
746     _RXE_SUB:       .EQU    $
747             BCLR.B  #RxE,           @IERSR      ; Reception of specified message length not
                                                      complete
```

```
748             BTST.B  #RTME,          @IEREF      ; Search error contents
749             BNE     RXE_SUB_RTME
750             BTST.B  #DLE,           @IEREF      ; Transfer byte overrun
751             BNE     RXE_SUB_DLE
752             BTST.B  #RSS,           @IEFLG      ; Parity error during reception in normal mode
753             BEQ     RXE_SUB_RSS
754     RXE_SUB_RXE:
755             BTST.B  #RxE,           @IERSR      ; Drop frame during reception
756             BEQ     RXE_SUB_RXE                 ; (Wait for transfer byte overrun error)
757             BRA     _RXE_SUB                    ; In broadcast mode, clear error immediately
758     RXE_SUB_RTME:
759             MOV.B   #R_TM_ERR,      R4H         ; Error code = timing error
760             BCLR.B  #RTME,          @IEREF      ; Clear timing error
761             BRA     RXE_SUB_ERR
762     RXE_SUB_DLE:
763             MOV.B   #R_DB_ERR,      R4H         ; Error code = transfer byte error
764             BCLR.B  #DLE,           @IEREF      ; Clear transfer byte error
765     RXE_SUB_RSS:
766             BTST.B  #PE,            @IEREF
767             BEQ     RXE_SUB_PE
768             MOV.B   #R_P_ERR,       R4H         ; Error code = parity error
769             BCLR.B  #PE,            @IEREF      ; Clear parity error
770             BRA     RXE_SUB_ERR
771     RXE_SUB_PE:
772             BTST.B  #OVE,           @IEREF
773             BEQ     RXE_SUB_ERR
774             MOV.B   #R_OR_ERR,      R4H         ; Error code = overrun error
775             BCLR.B  #OVE,           @IEREF      ; Clear overrun error
776             BCLR.B  #RxRDY,         @IERSR
777     RXE_SUB_ERR:
778             MOV.B   R4H,            @IEBUS_ERROR
779             RTS
780     ;*****************************************************************
781     ;**  _TXE_SUB   (master transmit error processing)              **
782     ;**=============================================================**
783     ;**     H'09            : Underrun error                       **
784     ;**     H'0A            : Timing error                         **
785     ;**     H'0B            : Transfer byte error                  **
786     ;**     H'0C            : ACK error                            **
787     ;*****************************************************************
788     _TXE_SUB:       .EQU    $
789             BCLR.B  #TxE,           @IETSR
790             BTST.B  #UE,            @IETEF          ; Search error contents
791             BNE     TXE_SUB_UE
792             BTST.B  #TTME,          @IETEF
793             BNE     TXE_SUB_TTME
794             BCLR.B  #RO,            @IETEF          ; Clear transfer byte overrun error
795             MOV.B   #T_DB_ERR,      R4H             ; Error code = transfer byte overrun error
796             BRA     TXE_SUB_ACK
797     TXE_SUB_UE:
798             BCLR.B  #UE,            @IETEF          ; Clear underrun error
799             MOV.B   #T_AR_ERR,      R4H             ; Error code = underrun error
800             BRA     TXE_SUB_END
801     TXE_SUB_TTME:
```

```
802             BCLR.B  #TTME,          @IETEF          ; Clear timing error
803             MOV.B   #T_TM_ERR,      R4H             ; Error code = timing error
804             BRA     TXE_SUB_END
805     TXE_SUB_ACK:
806             BTST.B  #ACK,           @IETEF
807             BEQ     TXE_SUB_END
808             BCLR.B  #ACK,           @IETEF          ; Clear ACK error
809             MOV.B   #T_AK_ERR,      R4H             ; Error code = ACK error
810     TXE_SUB_END:
811             MOV.B   R4H,            @IEBUS_ERROR
812             RTS
813     ;*****************************************************************
814     ;**  _IRA_SUB   (IEBus runaway processing)                      **
815     ;**=============================================================**
816     ;**     H'0C            : Reset during runaway                   **
817     ;*****************************************************************
818     _IRA_SUB:       .EQU    $
819             BCLR.B  #IRA,           @IETSR
820             MOV.B   #IRA_ERR,       R4H
821             MOV.B   R4H,            @IEBUS_ERROR
822             BSET.B  #IE_STP,        @MSTPCRC      ; IEBus controller module stop mode (reset)
823             RTS
824     ;*****************************************************************
825     ;**  _BLK_CHECK (erase block check)                             **
826     ;**=============================================================**
827     ;**     ER3             : Erase start address                    **
828     ;*****************************************************************
829     _BLK_CHECK:     .EQU    $
830             MOV.B   #0,             R4H             ; Clear R4H
831
832             MOV.L   #FLMCR1,        ER6             ; Set flash memory control register 1
833             MOV.L   #EBR1,          ER5             ; Set erase target block designation
                                                          register 1
834             MOV.L   #(BLOCK_ADR1-RAM_IN+RAM_ST),ER0 ; EBR1 start address
835             MOV.B   #MAX_BLK1,      R4L             ; Maximum block 1
836             CMP.L   #CHK_ADR,       ER3
837             BCS     BLK_CHK_1                       ; ERASE_ADDRESS < EBR2 start address
838
839             MOV.L   #EBR2,          ER5             ; Set erase target block designation
                                                          register 2
840             MOV.L   #(BLOCK_ADR2-RAM_IN+RAM_ST),ER0 ; EBR2 start address
841             MOV.B   #MAX_BLK2,      R4L             ; Maximum block 2
842     BLK_CHK_1:
843             MOV.L   @ER0+,          ER1
844             CMP.L   ER1,            ER3
845             BEQ     BLK_CHK_2                       ; COMPARE BLOCK_START_ADDRESS
846
847             INC.B   R4H                             ; Block number + 1
848             CMP.B   R4L,            R4H             ; BLOCK NUMBER MAX?
849             BEQ     BLK_ERR
850             BRA     BLK_CHK_1
851     BLK_CHK_2:
852             MOV.B   R4H,            @BLK_NO
853             MOV.L   ER3,            @EVF_ST
```

```
854             MOV.L   @ER0+,          ER1             ; ERASE END ADDRESS
855             MOV.L   ER1,            @EVF_ED
856             MOV.B   #OK,            R0L
857             RTS
858     BLK_ERR:
859             MOV.B   #NG,            R0L             ; ERASE BLOCK ADDRESS ERROR
860             RTS
861     ;***************************************************************
862     ;**  _BLK_ERASE (block erase)                                  **
863     ;**=============================================================**
864     ;**     ER6             : FLMCR address                        **
865     ;**     ER5             : EBR address                          **
866     ;**     @EVF_ST         : Erase start address                  **
867     ;**     @EVF_ED         : Erase end address                    **
868     ;**     @BLK_NO         : Bit number of erase target block     **
869     ;**     @ET_COUNT       : Maximum erase count                  **
870     ;***************************************************************
871     _BLK_ERASE:     .EQU    $
872             MOV.W   #H'5A5F,        R0              ; WDT initial setting
873             MOV.W   R0,             @RSTCSR
874
875             MOV.W   #F_LOOP_1,      R0
876             BSET.B  #SWE,           @ER6            ; Set SWE bit
877     BLK_ST_WAIT:
878             SUB.W   #1,             R0              ; Wait after setting SWE (1 µs or more)
879             BNE     BLK_ST_WAIT
880
881             XOR.W   R0,             R0              ; Clear erase count counter
882             MOV.W   R0,             @COUNT
883
884             BSR     _FERASEVF                       ; Initial erase verify
885             CMP.B   #OK,            R0L
886             BEQ     BLK_ERASE_OK                    ; Initial verify end
887     BLK_COUNT:                                      ; ===== Erase and verify =====
888             BSR     _FERASE                         ; Erase
889             BSR     _FERASEVF                       ; Erase/verify
890             CMP.B   #OK,            R0L
891             BEQ     BLK_ERASE_OK                    ; Erase end
892
893             MOV.W   @COUNT,         R0              ; Erase count counter @COUNT + 1
894             INC.W   #1,             R0
895             MOV.W   R0,             @COUNT
896
897             MOV.W   @ET_COUNT,      E0
898             CMP.W   E0,             R0
899             BNE     BLK_COUNT                       ; Determine count (maximum erase count)
900     BLK_ERASE_NG:                                   ; ----- Abnormal end -----
901             MOV.W   #F_LOOP_100,    R0
902             BCLR.B  #SWE,           @ER6            ; Clear SWE bit
903     BLK_NG_WAIT:
904             DEC.W   #1,             R0              ; Wait after clearing SWE (100 µs or more)
905             BNE     BLK_NG_WAIT
906
907             MOV.B   #NG,            R0L             ; Set NG
```

```
908             RTS
909     BLK_ERASE_OK:                                   ; ----- Normal end -----
910             MOV.W   #F_LOOP_100,    R0
911             BCLR.B  #SWE,           @ER6            ; Clear SWE bit
912     BLK_OK_WAIT:
913             SUB.W   #1,             R0              ; Wait after clearing SWE (100 µs or more)
914             BNE     BLK_OK_WAIT
915
916             MOV.B   #OK,            R0L             ; Set OK
917             RTS
918     ;****************************************************************
919     ;**  _FERASEVF  (erase verify)                                  **
920     ;**============================================================**
921     ;**     ER6             : FLMCR address                         **
922     ;**     @EVF_ST         : Erase start address                   **
923     ;**     @EVF_ED         : Erase end address                     **
924     ;****************************************************************
925     _FERASEVF:      .EQU    $
926             MOV.L   @EVF_ST,        ER1
927             MOV.L   @EVF_ED,        ER3
928             MOV.W   #H'FFFF,        E0              ; Dummy write, erase verification data
929
930             MOV.W   #F_LOOP_6,      R0
931             BSET.B  #EV,            @ER6            ; Set EV bit
932     VRF_EV_WAIT:
933             DEC.W   #1,             R0              ; Wait after setting EV (6 µs or more)
934             BNE     VRF_EV_WAIT
935     VRF_COUNT:
936             MOV.W   #F_LOOP_2,      R0
937             MOV.W   E0,             @ER1            ; Dummy write (address latch)
938     VRF_LACH_WAIT:
939             DEC.W   #1,             R0              ; Wait after latch (2 µs or more)
940             BNE     VRF_LACH_WAIT
941
942             MOV.W   @ER1+,          R0
943             CMP.W   E0,             R0              ; Verify
944             BNE     VRF_NG                          ; If target address not erased, end
945
946             CMP.L   ER1,            ER3
947             BNE     VRF_COUNT
948
949             MOV.W   #F_LOOP_4,      R0
950             BCLR.B  #EV,            @ER6            ; Clear EV bit
951     VRF_OK_WAIT:
952             DEC.W   #1,             R0              ; Wait after clearing EV (4 µs or more)
953             BNE     VRF_OK_WAIT
954
955             MOV.B   #OK,            R0L             ; Set OK flag
956             RTS
957     VRF_NG:
958             MOV.W   #F_LOOP_4,      R0              ; ----- Abnormal end -----
959             BCLR.B  #EV,            @ER6            ; Clear EV bit
960     VRF_NG_WAIT:
961             SUB.W   #1,             R0              ; Wait after clearing EV (4 µs or more)
```

```
962             BNE     VRF_NG_WAIT
963
964             MOV.B   #NG,            R0L             ; Set NG flag
965             RTS
966     ;***************************************************************
967     ;**  _FERASE    (erase)                                        **
968     ;**=============================================================**
969     ;**     ER6             : FLMCR address                        **
970     ;**     ER5             : EBR address                          **
971     ;**     @BLK_NO         : Bit number of erase target block     **
972     ;***************************************************************
973     _FERASE:        .EQU    $
974             MOV.B   @BLK_NO,        R0H
975             BSET.B  R0H,            @ER5            ; Set bit of erase target block in EBR
976
977             MOV.W   #H'A57C,        R0              ; 25 MHz (20.8 ms)
978             MOV.W   R0,             @TCSR1          ; Set watchdog timer 1
979
980             MOV.W   #F_LOOP_100,    R0
981             BSET.B  #ESU,           @ER6            ; Set ESU bit
982     FER_ESU_WAIT:
983             DEC.W   #1,             R0              ; Wait after setting ESU (100 µs or more)
984             BNE     FER_ESU_WAIT
985
986             MOV.L   #F_TIME_10000,  ER0             ; 10 mS
987             BSET.B  #E,             @ER6            ; Set E bit (erase)
988     FER_ERASE_WAIT:                                 ; ===== Apply erase pulse =====
989             DEC.L   #1,             ER0             ; Erase time: 10 ms
990             BNE     FER_ERASE_WAIT
991
992             MOV.W   #F_LOOP_10,     R0
993             BCLR.B  #E,             @ER6            ; Clear E bit
994     FER_E_WAIT:
995             DEC.W   #1,             R0              ; Wait after clearing E (10 µs or more)
996             BNE     FER_E_WAIT
997
998             MOV.W   #F_LOOP_10,     R0
999             BCLR.B  #ESU,           @ER6            ; Clear ESU bit
1000     FER_OK_WAIT:
1001             DEC.W   #1,             R0              ; Wait after clearing ESU (10 µs or more)
1002             BNE     FER_OK_WAIT
1003
1004             MOV.W   #H'A500,        R0
1005             MOV.W   R0,             @TCSR1          ; Halt watchdog timer 1
1006             MOV.B   @BLK_NO,        R0H
1007             BCLR.B  R0H,            @ER5            ; Clear erase target block in EBR
1008             RTS
1009     ;***************************************************************
1010     ;**  _FWRITE_128         (write 128 bytes)                     **
1011     ;**=============================================================**
1012     ;**     @WT_ADR         : Write address                        **
1013     ;**     @RDATA_BUFF     : 128 bytes of programming data        **
1014     ;**     @WT_COUNT       : Maximum write count                  **
1015     ;***************************************************************
```

```
_FWRITE_128:    .EQU    $
        MOV.W   #H'5A5F,        R0          ; WDT initial setting
        MOV.W   R0,             @RSTCSR

        MOV.W   #128,           R4
        MOV.L   #RDATA_BUFF,    ER5
        MOV.L   #RW_BUFF,       ER6
        EEPMOV.W                            ; Block transfer from RDATA_BUFF to RW_BUFF

        MOV.L   #FLMCR1,        ER6         ; Flash control register pointer
        MOV.W   #F_LOOP_1,      R0
        BSET.B  #SWE,           @ER6        ; Set SWE bit
FW128_SWE_WAIT:
        SUB.W   #1,             R0          ; Wait after setting SWE (1 µs or more)
        BNE     FW128_SWE_WAIT

       XOR.W   R0,             R0          ; Clear write count counter
        MOV.W   R0,             @COUNT
        BSR     _FWRITEVF                   ; ===== Initial verify =====
        CMP.B   #OK,            R0L
        BEQ     FW128_OK                    ; Initial verify end

        CMP.B   #W_ERR,         R0L
        BEQ     FW128_NG                    ; Write completed error
FW128_RE_COUNT:                             ; ===== Initial write (with additions)=====
        MOV.L   #RW_BUFF,       ER2         ; Overwrite data
        MOV.L   #F_TIME_30,     ER3         ; Apply P pulse (30 µs)
        BSR     _FWRITE                     ; Write
        BSR     _FWRITEVF                   ; Write verify
        MOV.B   R0L,            @VF_RET
        MOV.L   #OW_BUFF,       ER2         ; Additional programming data
        MOV.L   #F_TIME_10,     ER3         ; Apply P pulse (10 µs)
        BSR     _FWRITE                     ; Additional write
        MOV.B   @VF_RET,        R0L
        CMP.B   #OK,            R0L
        BEQ     FW128_OK                    ; Write end

        CMP.B   #W_ERR,         R0L
        BEQ     FW128_NG                    ; Write completed error

        MOV.W   @COUNT,         R0          ; Write count counter @COUNT + 1
        ADD.W   #1,             R0
        MOV.W   R0,             @COUNT
        CMP.W   #MAX_OW,        R0
        BNE     FW128_RE_COUNT              ; Determine count (additional write count)
FW128_COUNT:                                ; ===== Normal write (no additions)=====
        MOV.L   #RW_BUFF,       ER2         ; Overwrite data
        MOV.L   #F_TIME_200,    ER3         ; Apply P pulse (200 µs)
        BSR     _FWRITE                     ; Write
        BSR     _FWRITEVF                   ; Write verify
        CMP.B   #OK,            R0L
        BEQ     FW128_OK                    ; Write end

        CMP.B   #W_ERR,         R0L
```

```
1070            BEQ     FW128_NG                    ; Write completed error
1071
1072            MOV.W   @COUNT,         R0          ; Write count counter @COUNT + 1
1073            ADD.W   #1,             R0
1074            MOV.W   R0,             @COUNT
1075            MOV.W   @WT_COUNT,      E0
1076            CMP.W   E0,             R0
1077            BNE     FW128_COUNT                 ; Determine count (maximum write count)
1078    FW128_NG:                                   ; ----- Abnormal end -----
1079            MOV.W   #F_LOOP_100,    R0
1080            BCLR.B  #SWE,           @ER6        ; Clear SWE bit
1081    FW128_NG_WAIT:
1082            SUB.W   #1,             R0          ; Wait after clearing SWE (100 µs or more)
1083            BNE     FW128_NG_WAIT
1084            MOV.B   #NG,            R4L         ; NG code ->(R4L)
1085            RTS
1086    FW128_OK:                                   ; ----- Normal end -----
1087            MOV.W   #F_LOOP_100,    R0
1088            BCLR.B  #SWE,           @ER6        ; Clear SWE bit
1089    FW128_OK_WAIT:
1090            SUB.W   #1,             R0          ; Wait after clearing SWE (100 µs or more)
1091            BNE     FW128_OK_WAIT
1092
1093            MOV.B   #OK,            R4L         ; OK code ->(R4L)
1094            RTS RTS
1095    ;***************************************************************
1096    ;**  _FWRITEVF  (verify, overwrite)                           **
1097    ;**=============================================================**
1098    ;**     @WT_ADR      : Write address                          **
1099    ;**     @RDATA_BUFF  : 128 bytes of programming data          **
1100    ;**     @RW_BUFF     : 128 bytes of overwrite data            **
1101    ;**     @OW_BUFF     : 128 bytes of additional programming data **
1102    ;***************************************************************
1103    _FWRITEVF:      .EQU    $
1104            MOV.L   #RW_BUFF,       ER1             ; Overwrite data buffer
1105            MOV.L   #RDATA_BUFF,    ER2             ; Programming data buffer
1106            MOV.L   #OW_BUFF,       ER3             ; Additional programming data buffer
1107            MOV.L   @WT_ADR,        ER4             ; Flash ROM write address
1108            MOV.W   #H'FFFF,        R5              ; Dummy write data for address latch
1109
1110            MOV.W   #F_LOOP_4,      E0
1111            BSET.B  #PV,            @ER6            ; Set PV bit
1112    FWVF_PV_WAIT:
1113            SUB.W   #1,             E0              ; Wait after setting PV (4 µs or more)
1114            BNE     FWVF_PV_WAIT
1115    FWVF_COUNT:
1116            MOV.W   #F_LOOP_2,      E0
1117            MOV.W   R5,             @ER4            ; Dummy write (latch)
1118    FWVF_LACH_WAIT:
1119            SUB.W   #1,             E0              ; Latch wait (2 µs or more)
1120            BNE     FWVF_LACH_WAIT
1121
1122            MOV.W   @ER4+,          R0              ; Flash ROM data
1123            ;======== Create additional programming data =============
```

```
1124            MOV.W   @ER1,           E0              ; Initial write (used 6 times only)
1125            OR.W    R0,             E0              ; @COUNT = 0,1,2,3,4,5 valid data
1126            MOV.W   E0,             @ER3            ; @COUNT = 6 to 999      invalid data
1127            ADDS    #2,             ER3
1128
1129            NOT.W   R0
1130            MOV.W   @ER2,           E0
1131            OR.W    R0,             E0              ; Inverted data OR programming data
1132            MOV.W   E0,             @ER1            ; Set overwrite data
1133            ADDS    #2,             ER1
1134            MOV.W   @ER2+,          E0
1135            AND.W   E0,             R0              ; If read data 0 and write data 1
1136            BNE     FWVF_NG                         ; To cannot write error
1137
1138            CMP.L   #RDATA_BUFF+128,ER2
1139            BNE     FWVF_COUNT                      ; Verify 128 bytes
1140
1141            MOV.W   #F_LOOP_2,      E0
1142            BCLR.B  #PV,            @ER6            ; Clear PV bit
1143    FWVF_RE_PV_WAIT:
1144            SUB.W   #1,             E0              ; Wait after clearing PV (2 µs or more)
1145            BNE     FWVF_RE_PV_WAIT
1146
1147            MOV.B   #NG,            R0L             ; Set NG
1148            MOV.L   #RW_BUFF,       ER1             ; Start address of overwrite data
1149    FWVF_RE_COUNT:
1150            MOV.W   @ER1+,          E0
1151            CMP.W   R5,             E0              ; All 128 bytes of overwrite data FF?
1152            BNE     FWVF_ERR                        ; Verify error if other than H'FF
1153
1154            CMP.L   #RW_BUFF+128,   ER1
1155            BNE     FWVF_RE_COUNT                   ; Verify 128 bytes
1156
1157            MOV.B   #OK,            R0L             ; Set OK
1158    FWVF_ERR:
1159            RTS
1160    FWVF_NG:
1161            MOV.W   #F_LOOP_2,      E0              ; ----- Abnormal end -----
1162            BCLR.B  #PV,            @ER6            ; Clear PV bit
1163    FWVF_NG_WAIT:
1164            SUB.W   #1,             E0              ; Wait after clearing PV (2 µs or more)
1165            BNE     FWVF_NG_WAIT
1166
1167            MOV.B   #W_ERR,         R0L             ; Set W_ERR
1168            RTS
1169    ;*****************************************************************
1170    ;**  _FWRITE    (write)                                         **
1171    ;**=============================================================**
1172    ;**     ER6          : Address in FLMCR register                **
1173    ;**     @WT_ADR      : Write address                            **
1174    ;**     ER2          : Overwrite/additional write start address **
1175    ;**     ER3          : P bit setting time (10, 30, 2,000 µs)    **
1176    ;*****************************************************************
1177    _FWRITE:        .EQU    $
```

```
1178    MOV.L   @WT_ADR,        ER1                     ; Write address
1179            MOV.W   #128,           E0
1180    FWRT_DUMMY:
1181            MOV.B   @ER2+,          R0L             ; Overwrite data (byte units)
1182            MOV.B   R0L,            @ER1            ; Dummy write (byte units)
1183            ADDS    #1,             ER1
1184            SUB.W   #1,             E0
1185            BNE     FWRT_DUMMY                      ; Repeat, 128 bytes at a time
1186
1187            MOV.W   #H'A579,        R0              ; 25 MHZ(655.36 µs)
1188            MOV.W   R0,             @TCSR1          ; Set watchdog timer 1
1189            MOV.W   #F_LOOP_50,     E0
1190            BSET.B  #PSU,           @ER6            ; Set PSU bit
1191    FWRT_PSU_WAIT:
1192            SUB.W   #1,             E0              ; Wait after setting PSU (50 µs or more)
1193            BNE     FWRT_PSU_WAIT
1194
1195            BSET.B  #P,             @ER6            ; Set P bit (write)
1196    FWRT_WRITE_WAIT:                                ; ===== Apply write pulse =====
1197            SUB.L   #1,             ER3             ; Write time: 10, 30, 200 µs
1198            BNE     FWRT_WRITE_WAIT
1199
1200            MOV.W   #F_LOOP_5,      E0
1201            BCLR.B  #P,             @ER6            ; Clear P bit
1202    FWRT_P_WAIT:
1203            SUB.W   #1,             E0              ; Wait after clearing P (5 µs)
1204            BNE     FWRT_P_WAIT
1205
1206            MOV.W   #F_LOOP_5,      E0
1207            BCLR.B  #PSU,           @ER6            ; Clear PSU bit
1208    FWRT_END_WAIT:
1209            SUB.W   #1,             E0              ; Wait after clearing PSU (5 µs or more)
1210            BNE     FWRT_END_WAIT
1211
1212            MOV.W   #H'A500,        R0
1213            MOV.W   R0,             @TCSR1          ; Halt watchdog timer 1
1214            RTS
1215    ;*****************************************************************
1216    ;**     Flash memory block address table                        **
1217    ;*****************************************************************
1218            .ALIGN  2
1219    BLOCK_ADR1:     .EQU    $
1220            .DATA.L         H'00000000      ; EB0   4KBYTES
1221            .DATA.L         H'00001000      ; EB1   4KBYTES
1222            .DATA.L         H'00002000      ; EB2   4KBYTES
1223            .DATA.L         H'00003000      ; EB3   4KBYTES
1224            .DATA.L         H'00004000      ; EB4   4KBYTES
1225            .DATA.L         H'00005000      ; EB5   4KBYTES
1226            .DATA.L         H'00006000      ; EB6   4KBYTES
1227            .DATA.L         H'00007000      ; EB7   4KBYTES
1228            .DATA.L         H'00008000      ; END_ADRESS1
1229    BLOCK_ADR2:     .EQU    $
1230            .DATA.L         H'00008000      ; EB8  32KBYTES
1231            .DATA.L         H'00010000      ; EB9  64KBYTES
```

```
1232              .DATA.L         H'00020000      ; EB10 64KBYTES
1233              .DATA.L         H'00030000      ; EB11 64KBYTES
1234              .DATA.L         H'00040000      ; END_ADRESS2
1235              .DATA.L         H'00000000      ; DUMMY
1236              .DATA.L         H'00000000      ; DUMMY
1237              .DATA.L         H'00000000      ; DUMMY
1238              .DATA.L         H'00000000      ; DUMMY
1239     PROG_END:        .EQU    $
1240              .END
  *****TOTAL ERRORS       0
  *****TOTAL WARNINGS     0
```

```
*** CROSS REFERENCE LIST

NAME             SECTION  ATTR VALUE            SEQUENCE

ACK                       EQU  00000000    154*  806   808
AL                        EQU  00000004    150*
BLK_CHK_1        PROG_2        000126AA    837   842*  850
BLK_CHK_2        PROG_2        000126BE    845   851*
BLK_COUNT        PROG_2        0001270A    887*  899
BLK_ERASE_NG     PROG_2        00012730    900*
BLK_ERASE_OK     PROG_2        00012740    886   891   909*
BLK_ERR          PROG_2        000126DC    849   858*
BLK_NG_WAIT      PROG_2        00012738    903*  905
BLK_NO           RAM           00FFE21E    226*  431   852   974  1006
BLK_OK_WAIT      PROG_2        00012748    912*  914
BLK_ST_WAIT      PROG_2        000126F2    877*  879
BLOCK_ADR1       PROG_2   EQU  000129FC    834  1219*
BLOCK_ADR2       PROG_2   EQU  00012A20    840  1229*
CHK_ADR                   EQU  00008000     86*  836
CKS                       EQU  00000004    109*
CMX                       EQU  00000007    130*  692
COUNT            RAM           00FFE20C    219*  882   893   895  1033  1056  1058  1072  1074
DEE                       EQU  00000005    108*
DH_BIT                    EQU  00000000     13*  650
DLE                       EQU  00000001    168*  750   764
DTC_STP                   EQU  00000006     60*
E                         EQU  00000001     96*  987   993
EBR1                      EQU  00FFFFAA    100*  833
EBR2                      EQU  00FFFFAB    101*  839
END_BLOCK        PROG_2        000121AC    422   426*
END_DATA         PROG_2        000122E2    519   521*
ER_COUNT         RAM           00FFE220    228*  387   398   418   420   436
ER_TRANS         PROG_2        000121AE    425   428*
ESU                       EQU  00000005     92*  981   999
ET_COUNT         RAM           00FFE20E    220*  403   897
EV                        EQU  00000003     94*  931   950   959
EVF_ED           RAM           00FFE21A    225*  855   927
EVF_ST           RAM           00FFE216    224*  429   853   926
E_ERR                     EQU  00000002     35*  448
Erase_go                  EQU  00000077     53*  389
FER_ERASE_WAIT   PROG_2        000127D4    988*  990
FER_ESU_WAIT     PROG_2        000127C6    982*  984
FER_E_WAIT       PROG_2        000127E0    994*  996
FER_OK_WAIT      PROG_2        000127EC   1000* 1002
FLER                      EQU  00000007     99*
FLMCR1                    EQU  00FFFFA8     89*  832  1025
FLMCR2                    EQU  00FFFFA9     98*
FLSHE                     EQU  00000003     88*  361
FW128_COUNT      PROG_2        000128A2   1061* 1077
FW128_NG         PROG_2        000128DC   1039  1054  1070  1078*
FW128_NG_WAIT    PROG_2        000128E4   1081* 1083
FW128_OK         PROG_2        000128EE   1036  1051  1067  1086*
FW128_OK_WAIT    PROG_2        000128F6   1089* 1091
FW128_RE_COUNT   PROG_2        00012850   1040* 1060
```

```
FW128_SWE_WAIT    PROG_2        00012832   1028* 1030
FWE                       EQU   00000007     90*
FWE_off                   EQU   00000099     55*  559
FWE_on                    EQU   00000066     52*  353
FWRT_DUMMY        PROG_2        000129A4   1180* 1185
FWRT_END_WAIT     PROG_2        000129EA   1208* 1210
FWRT_PSU_WAIT     PROG_2        000129C2   1191* 1193
FWRT_P_WAIT       PROG_2        000129DC   1202* 1204
FWRT_WRITE_WAIT   PROG_2        000129CC   1196* 1198
FWVF_COUNT        PROG_2        0001292C   1115* 1139
FWVF_ERR          PROG_2        00012984   1152  1158*
FWVF_LACH_WAIT    PROG_2        00012932   1118* 1120
FWVF_NG           PROG_2        00012986   1136  1160*
FWVF_NG_WAIT      PROG_2        0001298E   1163* 1165
FWVF_PV_WAIT      PROG_2        00012926   1112* 1114
FWVF_RE_COUNT     PROG_2        00012972   1149* 1155
FWVF_RE_PV_WAIT   PROG_2        00012964   1143* 1145
F_LOOP_1                  EQU   00000004    180*  875  1026
F_LOOP_10                 EQU   00000022    185*  992   998
F_LOOP_100                EQU   00000152    188*  901   910   980  1079  1087
F_LOOP_2                  EQU   00000007    181*  936  1116  1141  1161
F_LOOP_30                 EQU   00000066    186*
F_LOOP_4                  EQU   0000000E    182*  949   958  1110
F_LOOP_5                  EQU   00000011    183* 1200  1206
F_LOOP_50                 EQU   000000A9    187* 1189
F_LOOP_6                  EQU   00000015    184*  930
F_TIME_10                 EQU   00000021    189* 1047
F_TIME_10000              EQU   000083D6    192*  986
F_TIME_200                EQU   000002A3    191* 1063
F_TIME_30                 EQU   00000065    190* 1042
GG                        EQU   00000000    136*
IEAR1                     EQU   00FFF803    115*  282
IEAR2                     EQU   00FFF804    117*  284
IEBUS_ERROR       RAM           00FFE087    210*  303   349   382   406   460   474   487   504
                                              530   555   627   778   811   821
IECMR                     EQU   00FFF801    112*  696
IECTR                     EQU   00FFF800    105*  278   289
IEE                       EQU   00000007    106*  289
IEFLG                     EQU   00FFF810    129*  692   752
IEIER                     EQU   00FFF815    160*  286
IEIET                     EQU   00FFF812    143*  288
IELA1                     EQU   00FFF80E    127*
IELA2                     EQU   00FFF80F    128*
IEMA1                     EQU   00FFF809    122*
IEMA2                     EQU   00FFF80A    123*
IEMCR                     EQU   00FFF802    113*  280   664   667   675   680
IER                       EQU   00FFFE14     76*  245
IERBFL                    EQU   00FFF80C    125*  605
IERBR                     EQU   00FFF80D    126*  617
IERCTL                    EQU   00FFF80B    124*
IEREF                     EQU   00FFF816    165*  748   750   760   764   766   769   772   775
IERSR                     EQU   00FFF814    155*  601   604   610   614   621   629   633   642
                                              747   755   776
IESA1                     EQU   00FFF805    118*  673
```

```
IESA2                   EQU  00FFF806    119*  670
IETBFL                  EQU  00FFF807    120*  683
IETBR                   EQU  00FFF808    121*  688   719
IETEF                   EQU  00FFF813    149*  790   792   794   798   802   806   808
IETSR                   EQU  00FFF811    137*  612   631   690   698   700   702   705   707
                                               709   711   721   724   729   733   736   789
                                               819
IE_RCV_DATA     PROG_2       000123BC    609*  615   624
IE_RCV_END      PROG_2       00012430    635   638   641*
IE_RCV_ERR      PROG_2       00012424    611   630   636*
IE_RCV_IRA      PROG_2       0001242C    613   632   639*
IE_RCV_RET      PROG_2       000123FE    628*  634
IE_STP                  EQU  00000003     63*  276   822
IE_TRS_CMD      PROG_2       000124BE    691*  693
IE_TRS_DATA     PROG_2       000124FA    706*  712   722
IE_TRS_END      PROG_2       00012560    726   735*
IE_TRS_ERR      PROG_2       00012544    699   708   727*
IE_TRS_FINISH   PROG_2       00012536    716   723*  725
IE_TRS_IRA      PROG_2       00012552    701   710   731*
IE_TRS_ST       PROG_2       000124D0    697*  703
IOL                     EQU  00000006    107*
IRA                     EQU  00000003    139*  612   631   700   709   819
IRAE                    EQU  00000003    145*
IRA_ERR                 EQU  0000000D     46*  820
IRQ2_LOOP       PROG_2       0001200C    260*  261
IRQ2_TOP                EQU  00012000     24*  255
ISR                     EQU  00FFFE15     77*  242   243
LCK                     EQU  00000003    134*
LUEE                    EQU  00000002    111*
MADR_H                  EQU  000000BB     14*  283
MADR_L                  EQU  000000B0     15*  281
MAX_BLK1                EQU  00000008     84*  835
MAX_BLK2                EQU  00000004     85*  841
MAX_ET                  EQU  00000064     82*  402
MAX_OW                  EQU  00000006     81* 1059
MAX_WT                  EQU  000003E8     83*  493
MHZ                     EQU  00000546    179*  180   181   182   183   184   185   186   187
                                               188   189   190   191   192
MRQ                     EQU  00000006    131*
MSTPCRA                 EQU  00FFFDE8     59*
MSTPCRB                 EQU  00FFFDE9     61*
MSTPCRC                 EQU  00FFFDEA     62*  276   822
NEXT_BLOCK      PROG_2       00012168    404*  438
NEXT_DATA       PROG_2       0001227E    495*  520
NG                      EQU  00000001     34*  859   907   964  1084  1147
NG_55           PROG_2       0001208A    304   309   312*
NG_66           PROG_2       0001210C    350   355   369*
NG_77           PROG_2       000121DA    383   391   407   440*
NG_77E          PROG_2       000121EC    412   416   446*
NG_88           PROG_2       00012320    461   466   475   488   505   509   531   535   542*
NG_99           PROG_2       00012380    556   561   578*
OK                      EQU  00000000     33*  338   363   393   411   415   427   468   481
                                               508   534   537   572   856   885   890   916
                                               955  1035  1050  1066  1093  1157
```

```
OK_88             PROG_2        0001230E    536*
OVE                       EQU   00000003    166*  772   775
OW_BUFF           RAM           00FFE18C    218* 1046  1106
P                         EQU   00000000     97* 1195  1201
P3DDR                     EQU   00FFFE32     67*  267   570
P3DR                      EQU   00FFFF02     70*  568
PE                        EQU   00000000    169*  766   769
PFDDR                     EQU   00FFFE3E     68*  359   565
PFDR                      EQU   00FFFF0E     71*  357   563
PGDDR                     EQU   00FFFE3F     69*  270
PGDR                      EQU   00FFFF0F     72*  268
PRESET_LOOP       PROG_R        00010020    250*  251
PRESET_TOP                EQU   00010000     23*  239
PROG_2            PROG_2  SCT   00012000    255*
PROG_END          PROG_2  EQU   00012A44    325  1239*
PROG_R            PROG_R  SCT   00010000    239*
PROG_TOP                  EQU   00001000     22*  588
PROG_ZERO                 EQU   00000000     21*  232
PSU                       EQU   00000004     93* 1190  1207
PV                        EQU   00000002     95* 1111  1142  1162
RAM               RAM     SCT   00FFE000    196*
RAM_BUFF                  EQU   00FFE000     29*  196
RAM_END                   EQU   00FFEFBF     27*
RAM_FINISH                EQU   00FFEFC0     28*  241
RAM_IN            PROG_2  EQU   000120B6    323   325   332*  834   840
RAM_ST                    EQU   00FFC000     25*  324   327   834   840
RAM_TOP                   EQU   00FFB000     26*
RDATA_BUFF        RAM           00FFE088    211*  306   352   385   409   463   477   490   558
                                                  608  1021  1105  1138
RDENBUN_BUFF      RAM           00FFE086    209*  606
RE                        EQU   00000003    110*
RECEIV_BUFF       RAM     EQU   00FFE086    208*
RECEIV_WAIT       PROG_2        00012398    600*  602
REST_SIZE         RAM           00FFE212    222*  491   515   517
RO                        EQU   00000001    153*  794
RSS                       EQU   00000001    135*  752
RSTCSR                    EQU   00FFFFA4    174*  873  1018
RTME                      EQU   00000002    167*  748   760
RW_BUFF           RAM           00FFE10C    217* 1022  1041  1062  1104  1148  1154
RXE_SUB_DLE       PROG_2        000125B0    751   762*
RXE_SUB_ERR       PROG_2        000125F2    761   770   773   777*
RXE_SUB_PE        PROG_2        000125D4    767   771*
RXE_SUB_RSS       PROG_2        000125BA    753   765*
RXE_SUB_RTME      PROG_2        000125A2    749   758*
RXE_SUB_RXE       PROG_2        00012596    754*  756
R_DB_ERR                  EQU   00000005     38*  763
R_OR_ERR                  EQU   00000007     40*  774
R_P_ERR                   EQU   00000006     39*  768
R_TM_ERR                  EQU   00000004     37*  759
Request                   EQU   00000011     47*  424   497   523
RxE                       EQU   00000000    159*  610   629   747   755
RxEE                      EQU   00000000    164*
RxF                       EQU   00000001    158*  633   642
RxFE                      EQU   00000001    163*
```

```
RxRDY                   EQU  00000007    156*  614   621   776
RxRDYE                  EQU  00000007    161*
RxS                     EQU  00000002    157*  601   604
RxSE                    EQU  00000002    162*
SADR_BUFF_H     RAM          00FFE001    199*  653
SADR_BUFF_L     RAM          00FFE002    200*  655
SADR_H                  EQU  000000AA     16*  652
SADR_L                  EQU  000000A0     17*  654
SCRX                    EQU  00FFFDB4     87*  361
SRE                     EQU  00000004    133*
SRQ                     EQU  00000005    132*
SS                      EQU  00000007    114*  664   667
STE                     EQU  00000000    116*
SWE                     EQU  00000006     91*  876   902   911  1027  1080  1088
TCONT_BUFF      RAM          00FFE003    201*  657
TCSR1                   EQU  00FFFFA2    173*  978  1005  1188  1213
TDATA_BUFF      RAM          00FFE006    204*  315   339   364   371   394   432   442   449
                                              469   482   499   525   538   544   573   580
TDENBUN_BUFF    RAM          00FFE004    202*  658
TDUMMY          RAM          00FFE005    203*
THEDDA_BUFF     RAM          00FFE000    198*  651
TRANS_BUFF      RAM     EQU  00FFE000    197*  660
TRS_SETUP_DH    PROG_2       00012480    665   668*
TRS_SETUP_NR    PROG_2       00012478    662   666*
TTME                    EQU  00000002    152*  792   802
TXE_SUB_ACK     PROG_2       00012644    796   805*
TXE_SUB_END     PROG_2       0001265A    800   804   807   810*
TXE_SUB_TTME    PROG_2       00012636    793   801*
TXE_SUB_UE      PROG_2       00012628    791   797*
T_AK_ERR                EQU  0000000C     45*  809
T_AL_ERR                EQU  00000008     41*
T_AR_ERR                EQU  00000009     42*  799
T_DB_ERR                EQU  0000000B     44*  795
T_TM_ERR                EQU  0000000A     43*  803
TxE                     EQU  00000000    142*  698   707   789
TxEE                    EQU  00000000    148*
TxF                     EQU  00000001    141*  724   729   733   736
TxFE                    EQU  00000001    147*
TxRDY                   EQU  00000007    138*  690   711   721
TxRDYE                  EQU  00000007    144*
TxS                     EQU  00000002    140*  702   705
TxSE                    EQU  00000002    146*
UE                      EQU  00000003    151*  790   798
VECT            VECT    SCT  00000000    232*
VF_RET          RAM          00FFE21F    227* 1045  1049
VRF_COUNT       PROG_2       00012772    935*  947
VRF_EV_WAIT     PROG_2       0001276E    932*  934
VRF_LACH_WAIT   PROG_2       00012778    938*  940
VRF_NG          PROG_2       00012798    944   957*
VRF_NG_WAIT     PROG_2       000127A0    960*  962
VRF_OK_WAIT     PROG_2       00012790    951*  953
WT_ADR          RAM          00FFE108    216*  478   511   513  1107  1178
WT_COUNT        RAM          00FFE210    221*  494  1075
W_ERR                   EQU  00000003     36* 1038  1053  1069  1167
```

```
W_E_go                  EQU  00000055     51*  307
Write_go                EQU  00000088     54*  464
_BLK_CHECK       PROG_2 EQU  0001267C    410   829*
_BLK_ERASE       PROG_2 EQU  000126E0    414   871*
_FERASE          PROG_2 EQU  000127AA    888   973*
_FERASEVF        PROG_2 EQU  00012752    884   889   925*
_FWRITE          PROG_2 EQU  00012998   1043  1048  1064  1177*
_FWRITEVF        PROG_2 EQU  00012900   1034  1044  1065  1103*
_FWRITE_128      PROG_2 EQU  00012806    507   533  1016*
_GO_Erase        PROG_2 EQU  0001211E    368   380*
_GO_FWE_off      PROG_2 EQU  00012332    541   553*
_GO_FWE_on       PROG_2 EQU  000120C6    347*
_GO_W_E          PROG_2 EQU  000120B8    337*
_GO_Write        PROG_2 EQU  000121FE    400   439   458*
_IE_INT          PROG_2 EQU  00012028    258   275*
_IE_RECEIV       PROG_2 EQU  00012396    302   348   381   405   459   473   486   503   529
                                               554   598*
_IE_TRANS        PROG_2 EQU  0001243A    317   341   366   373   396   434   444   451   471
                                               484   501   527   540   546   575   582   649*
_IE_TRS_CMD      PROG_2 EQU  0001206A    259   301*
_IO_INT          PROG_2 EQU  0001200E    257   265*
_IRA_SUB         PROG_2 EQU  00012662    640   732   818*
_IRQ2            PROG_2 EQU  00012000    235   256*
_PRESET          PROG_R EQU  00010000    233   240*
_RAM_OUT         PROG_2 EQU  00012392    577   587*
_RXE_SUB         PROG_2 EQU  0001256A    313   370   441   543   579   637   746*  757
_TXE_SUB         PROG_2 EQU  000125FA    728   788*
_Trace_to_RAM    PROG_2 EQU  0001209E    311   322*


*** SECTION DATA LIST

SECTION          ATTRIBUTE    SIZE            START

RAM              ABS-DATA    000000221        00FFE000
VECT             ABS-CODE    000000020        00000000
PROG_R           ABS-CODE    000000022        00010000
PROG_2           ABS-CODE    000000A44        00012000
```

Revision Record

		Description	
Rev.	Date	Page	Summary
1.00	Mar.09.05	—	First edition issued

Keep safety first in your circuit designs!

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

Notes regarding these materials

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