Date: Apr. 26, 2024

RENESAS TECHNICAL UPDATE

TOYOSU FORESIA, 3-2-24, Toyosu, Koto-ku, Tokyo 135-0061, Japan
Renesas Electronics Corporation

Product Category	MPU/MCU		Document No.	TN-RL*-A0134A/E	Rev.	2.00
Title	Correction for Incorrect Description Notice RL78/G22 Descriptions in the User's Manual: Hardware Rev. 1.00 Changed		Information Category	Technical Notification		
Applicable Product	RL78/G22 Group	Lot No. All lots	Reference Document	RL78/G22 User's Manual: Hardware Rev. 1.00 R01UH0978EJ0100 (Dec. 2022)		

This document describes misstatements found in the RL78/G22 User's Manual: Hardware Rev. 1.00 (R01UH0978JJ0100).

Corrections

Applicable Item	Applicable Page	Contents
3.1 Memory Space	Page 94, Page 95, Page 100	Incorrect descriptions revised
30.6.1 Self-programming procedure	Page 1065	Incorrect descriptions revised
30.10.1 Overview of the data flash memory	Page 1116	Incorrect descriptions revised
31.3 Security Settings for On-Chip Debugging	Page 1119	Incorrect descriptions revised

Document Improvement

The above corrections will be made for the next revision of the User's Manual: Hardware.

Corrections in the User's Manual: Hardware

No.	Corrections and Applicable Items			Pages in this document for corrections
		Document No.	English	R01UH0978EJ0100
1	26.3.3 Sequencer instruction registers p (SMSIp) (p = 0 to 31)			Page 964, Page 965	Page 3, Page 4
2	26.4 Operations of the SNOOZE Mode Sequencer			Page 974	Page 5
3	26.4.1 Internal operations of the SNOOZE mode sequencer			Page 971	Page 6
4	26.4.4 Procedures for running the SNOOZE mode sequencer			Page 975	Page 7
5	26.4.5 States of the SNOOZE mode sequencer			Page 977	Page 8
6	26.5.20 Interrupt plus termination			Page 999	Page 9
7	26.6 Operation in Standby Modes			Page 1001	Page 10
8	34.4 AC Characteristics			Page 1164	Page 11
9	3.1 Memory Space			Page 94, Page 95, Page 100	Page 12 to Page 14
10	30.6.1 Self-programming procedure			Page 1065	Page 15
11	30.10.1 Overview of the data flash memory			Page 1116	Page 16
12	31.3 Security Settings for On-Chip Debugging			Page 1119	Page 17

Incorrect: Bold with underline; Correct: Gray hatched

Revision History

RL78/G22 Correction for incorrect description notice

Document Number	Issue Date	Description
TN-RL*A0131A/E	Dec. 19, 2023	First edition issued Corrections No.1 to No.8 revised
TN-RL*-A0134A/E	Apr. 26, 2024	Second edition issued Corrections No.9 to No.12 revised (this document)

1. 26.3.3 Sequencer instruction registers p (SMSIp) (p = 0 to 31) (Page 964, Page 965)

Incorrect:

(Page 964)

Table 26 - 1 Correspondences between the Memory Addresses of the SMSIp Registers and Values of the SMSCV[4:0] Bits

SMSIp	Address	SMSCV[4:0]
SMSI15	F039EH, F039FH	01111B
SMSI14	F039CH, F039DH	01110B
SMSI13	F039AH, F039BH	01101B
SMSI12	F0398H, F0399H	01100B
SMSI11	F0396H, F0397H	01011B
SMSI10	F0394H, F0395H	01010B
SMSI9	F0392H, F0393H	01001B
SMSI8	F0390H, F0391H	01000B
SMSI7	F038EH, F038FH	00111B
SMSI6	F038CH, F038DH	00110B
SMSI5	F038AH, F038BH	00101B
SMSI4	F0388H, F0389H	00100B
SMSI3	F0386H, F0387H	00011B
SMSI2	F0384H, F0385H	00010B
SMSI1	F0382H, F0383H	00001B
SMSI0	F0380H, F0381H	00000B

SMSIp	Address	SMSCV[4:0]
SMSI31	F03BEH, F03BFH	11111B
SMSI30	F03BCH, F03BDH	11110B
SMSI29	F03BAH, F03BBH	11101B
SMSI28	F03B8H, F03B9H	11100B
SMSI27	F03B6H, F03B7H	11011B
SMSI26	F03B4H, F03B5H	11010B
SMSI25	F03B2H, F03B3H	11001B
SMSI24	F03B0H, F03B1H	11000B
SMSI23	F03AEH, F03AFH	10111B
SMSI22	F03ACH, F03ADH	10110B
SMSI21	F03AAH, F03ABH	10101B
SMSI20	F03A8H, F03A9H	10100B
SMSI19	F03A6H, F03A7H	10011B
SMSI18	F03A4H, F03A5H	10010B
SMSI17	F03A2H, F03A3H	10001B
SMSI16	F03A0H, F03A1H	10000B

Caution 1. Only set the SMSIp registers while the operation of the sequencer is stopped. Re-writing the SMSIp registers while the sequencer is handling the commands results in an undefined operation of the sequencer.

Caution 2. No register follows the SMSI31 register once the processing it defines has finished.
Therefore, set the SMSI31 register for processing for termination command or interrupt plus termination command to stop processing by the sequencer, or for branch processing so that the processing at the branch destination register is run.

Correct:

Table 26 - 1 Correspondences between the Memory Addresses of the SMSIp Registers and Values of the SMSCV[4:0] Bits

SMSIp	Address	SMSCV[4:0]
SMSI15	F039EH, F039FH	01111B
SMSI14	F039CH, F039DH	01110B
SMSI13	F039AH, F039BH	01101B
SMSI12	F0398H, F0399H	01100B
SMSI11	F0396H, F0397H	01011B
SMSI10	F0394H, F0395H	01010B
SMSI9	F0392H, F0393H	01001B
SMSI8	F0390H, F0391H	01000B
SMSI7	F038EH, F038FH	00111B
SMSI6	F038CH, F038DH	00110B
SMSI5	F038AH, F038BH	00101B
SMSI4	F0388H, F0389H	00100B
SMSI3	F0386H, F0387H	00011B
SMSI2	F0384H, F0385H	00010B
SMSI1	F0382H, F0383H	00001B
SMSI0	F0380H, F0381H	00000B

SMSIp	Address	SMSCV[4:0]
SMSI31	F03BEH, F03BFH	11111B
SMSI30	F03BCH, F03BDH	11110B
SMSI29	F03BAH, F03BBH	11101B
SMSI28	F03B8H, F03B9H	11100B
SMSI27	F03B6H, F03B7H	11011B
SMSI26	F03B4H, F03B5H	11010B
SMSI25	F03B2H, F03B3H	11001B
SMSI24	F03B0H, F03B1H	11000B
SMSI23	F03AEH, F03AFH	10111B
SMSI22	F03ACH, F03ADH	10110B
SMSI21	F03AAH, F03ABH	10101B
SMSI20	F03A8H, F03A9H	10100B
SMSI19	F03A6H, F03A7H	10011B
SMSI18	F03A4H, F03A5H	10010B
SMSI17	F03A2H, F03A3H	10001B
SMSI16	F03A0H, F03A1H	10000B

Caution 1. Only set the SMSIp registers while the operation of the sequencer is stopped. Re-writing the SMSIp registers while the sequencer is handling the commands results in an undefined operation of the sequencer.

Caution 2. No register follows the SMSI31 register once the processing it defines has finished.
If the command for terminating processing of commands or branch processing is not set in the SMSI31 register, the processing for termination is automatically executed once the processing defined in the SMSI31 register has finished.

(Page 965)

Table 26 - 2 Types of Processing Specified by the SMSIp Registers

Name of Processing	Operation Note 4	Sequencer Code	First Operand (4 Bits)	Second Operand (4 Bits)	Additional Byte (4 Bits)
8-bit data transfer 1	[SMSGn + Byte] ← SMSGm	0000	nth of SMSGn Note 1	mth of SMSGm Note 1	Byte Note 2
8-bit data transfer 2	SMSGm ← [SMSGn + Byte]	0001	nth of SMSGn Note 1	mth of SMSGm Note 1	Byte Note 2
16-bit data transfer 1	[SMSGn + Byte] ← SMSGm	0010	nth of SMSGn Note 1	mth of SMSGm Note 1	Byte Note 2
16-bit data transfer2	SMSGm ← [SMSGn + Byte]	0011	nth of SMSGn Note 1	mth of SMSGm Note 1	Byte Note 2
1-bit data setting	[SMSGn + Byte].bit ← 1	0100	nth of SMSGn Note 1	bit Note 2	Byte Note 2
1-bit data clearing	[SMSGn + Byte].bit ← 0	0101	nth of SMSGn Note 1	bit Note 2	Byte Note 2
1-bit data transfer	SCY ← [SMSGn + Byte].bit	0110	nth of SMSGn Note 1	bit Note 2	Byte Note 2
Word addition	SMSGn, SCY ← SMSGn + SMSGm	0111	nth of SMSGn Note 1	mth of SMSGm Note 1	0000
Word subtraction	SMSGn, SCY ← SMSGn - SMSGm	0111	nth of SMSGn Note 1	mth of SMSGm Note 1	0001
Word comparison	SMSGn - SMSGm	0111	nth of SMSGn Note 1	mth of SMSGm Note 1	0010
Logical shift right	SCY ← SMSGn.0, SMSGm.15 ← 0, SMSGn.m-1 ← SMSGn.m	0111	nth of SMSGn Note 1	0000	0011
Branch 1 (SCY = 1)	SMSS[4:0] ← SMSS[4:0] + jdisp8 if SCY = 1	1000	$addr5 Note 3		0000
Branch 2 (SCY = 0)	SMSS[4:0] ← SMSS[4:0] + jdisp8 if SCY = 0	1000	$addr5 Note 3		0001
Branch 3 (SZ = 1)	SMSS[4:0] ← SMSS[4:0] + jdisp8 if SZ = 1	1000	$addr5 Note 3		0010
Branch 4 (SZ = 0)	SMSS[4:0] ← SMSS[4:0] + jdisp8 if SZ = 0	1000	$addr5 Note 3		0011
Wait	Holding processing pending for a certain period	1001	IM1		IM2
Conditional wait 1 (bit = 1)	SMSS[4:0] if [SMSGn + Byte].bit = 1	1010	nth of SMSGn Note 1	Bit Note 2	Byte Note 2
Conditional wait 2 (bit = 0)	SMSS[4:0] if [SMSGn + Byte].bit = 0	1011	nth of SMSGn Note 1	Bit Note 2	Byte Note 2
Termination	SMSS[4:0] ← 0, Stopping the sequencer	1111	0000	0000	0000
Interrupt plus termination	**SMSS[4:0] ← 0, Stopping the sequencer after issuing an interrupt**	**1111**	**0000**	**0000**	**0001**
DTC activation	Output of a DTC activating source signal	1111	0000	0000	0010

Note 1. Specify values in the range from 0 to 15 (from 0000B to 1111B) for n and m.

Note 2. Specify values in the range from 0 to 7 (from 0000B to 0111B) for the bytes.

Note 3. This is an 8-bit displacement value. Specify a relative address in the ranges from -31 to -1 and 1 to 31 (0000 0001B to 0001 1111B, 1111 1111B to 1110 0001B).

Note 4. For details on the terms, see 26.5 Commands for Use in Processing by the Sequencer.

Table 26 - 2 Types of Processing Specified by the SMSIp Registers

Name of Processing	Operation Note 4	Sequencer Code	First Operand (4 Bits)	Second Operand (4 Bits)	Additional Byte (4 Bits)
8-bit data transfer 1	[SMSGn + Byte] ← SMSGm	0000	nth of SMSGn Note 1	mth of SMSGm Note 1	Byte Note 2
8-bit data transfer 2	SMSGm ← [SMSGn + Byte]	0001	nth of SMSGn Note 1	mth of SMSGm Note 1	Byte Note 2
16-bit data transfer 1	[SMSGn + Byte] ← SMSGm	0010	nth of SMSGn Note 1	mth of SMSGm Note 1	Byte Note 2
16-bit data transfer2	SMSGm ← [SMSGn + Byte]	0011	nth of SMSGn Note 1	mth of SMSGm Note 1	Byte Note 2
1-bit data setting	[SMSGn + Byte].bit ← 1	0100	nth of SMSGn Note 1	bit Note 2	Byte Note 2
1-bit data clearing	[SMSGn + Byte].bit ← 0	0101	nth of SMSGn Note 1	bit Note 2	Byte Note 2
1-bit data transfer	SCY ← [SMSGn + Byte].bit	0110	nth of SMSGn Note 1	bit Note 2	Byte Note 2
Word addition	SMSGn, SCY ← SMSGn + SMSGm	0111	nth of SMSGn Note 1	mth of SMSGm Note 1	0000
Word subtraction	SMSGn, SCY ← SMSGn - SMSGm	0111	nth of SMSGn Note 1	mth of SMSGm Note 1	0001
Word comparison	SMSGn - SMSGm	0111	nth of SMSGn Note 1	mth of SMSGm Note 1	0010
Logical shift right	SCY ← SMSGn.0, SMSGm.15 ← 0, SMSGn.m-1 ← SMSGn.m	0111	nth of SMSGn Note 1	0000	0011
Branch 1 (SCY = 1)	SMSS[4:0] ← SMSS[4:0] + jdisp8 if SCY = 1	1000	$addr5 Note 3		0000
Branch 2 (SCY = 0)	SMSS[4:0] ← SMSS[4:0] + jdisp8 if SCY = 0	1000	$addr5 Note 3		0001
Branch 3 (SZ = 1)	SMSS[4:0] ← SMSS[4:0] + jdisp8 if SZ = 1	1000	$addr5 Note 3		0010
Branch 4 (SZ = 0)	SMSS[4:0] ← SMSS[4:0] + jdisp8 if SZ = 0	1000	$addr5 Note 3		0011
Wait	Holding processing pending for a certain period	1001	IM1		IM2
Conditional wait 1 (bit = 1)	SMSS[4:0] if [SMSGn + Byte].bit = 1	1010	nth of SMSGn Note 1	Bit Note 2	Byte Note 2
Conditional wait 2 (bit = 0)	SMSS[4:0] if [SMSGn + Byte].bit = 0	1011	nth of SMSGn Note 1	Bit Note 2	Byte Note 2
Termination	SMSS[4:0] ← 0, Stopping the sequencer	1111	0000	0000	0000
DTC activation	Output of a DTC activating source signal	1111	0000	0000	0010

Note 1. Specify values in the range from 0 to 15 (from 0000B to 1111B) for n and m.

Note 2. Specify values in the range from 0 to 7 (from 0000B to 0111B) for the bytes.

Note 3. This is an 8-bit displacement value. Specify a relative address in the ranges from -31 to -1 and 1 to 31 (0000 0001B to 0001 1111B, 1111 1111B to 1110 0001B).

Note 4. For details on the terms, see 26.5 Commands for Use in Processing by the Sequencer.

2. 26.4 Operations of the SNOOZE Mode Sequencer (Page 974)

Incorrect:

26.4.3 Sequencer flags

The sequencer has flags that are set or reset in response to the results of operations.

a) Sequencer zero flag (SZ)
The SZ flag is an internal flag of the sequencer. The flag is set to 1 when the result of addition, subtraction, or comparison is 0. Otherwise, the flag is cleared to 0. The flag is only for use in the internal processing by the sequencer. For details, see 26.5 Commands for Use in Processing by the Sequencer.

b) Sequencer carry flag (SCY)
The SCY flag reflects the state of addition or subtraction producing an overflow or underflow, the value of the shifted-out bit in logical shifting processing, or the result of 1-bit data transfer. The flag is only for use in the internal processing by the sequencer. For details, see 26.5 Commands for Use in Processing by the Sequencer.

The values of the SZ and SCY flags of the sequencer can be read from the corresponding bits of the SMSS register. See 26.3.6 Sequencer status register (SMSS).

Correct:

26.4.3 Sequencer flags

The sequencer has flags that are set or reset in response to the results of operations.

a) Sequencer zero flag (SZ)
The SZ flag is an internal flag of the sequencer. The flag is set to 1 when the result of addition, subtraction, or comparison is 0. Otherwise, the flag is cleared to 0. The flag is only for use in the internal processing by the sequencer. For details, see 26.5 Commands for Use in Processing by the Sequencer.

b) Sequencer carry flag (SCY)
The SCY flag reflects the state of addition or subtraction producing an overflow or underflow, the value of the shifted-out bit in logical shifting processing, or the result of 1-bit data transfer. The flag is only for use in the internal processing by the sequencer. For details, see 26.5 Commands for Use in Processing by the Sequencer.

The values of the SZ and SCY flags of the sequencer can be read from the corresponding bits of the SMSS register. See 26.3.6 Sequencer status register (SMSS).

26.4.4 Interrupt from the SNOOZE Mode Sequencer

The SMSEMK bit controls generation of the INTSMSE interrupt from the SNOOZE mode sequencer. Before starting the SNOOZE mode sequencer operation (by setting SMSSTART to 1), use the CPU to set the SMSEMK and SMSEIF bits to 1. The SMSEIF and SMSEMK bits are respectively set to 1 and 0 by setting the SMSEMK bit to 0 within the SNOOZE mode sequencer processing, which leads to generation of the INTSMSE interrupt. When the interrupt is disabled (DI), the SMSEMK bit being 0 indicates the end of the sequencer operation.

Caution 1. Do not use a CPU instruction to set the SMSEMK bit in the MK0H register or SMSEIF bit in the IF0H register to 0 while the setting of the SMSSTART bit in the SMSC register is 1.

Caution 2. If processing by the SNOOZE mode sequencer and that for the INTSMSE interrupt involve access to the same area in the SFR or RAM, ensure that the two types of processing do not run at the same time.

3. 26.4.1 Internal operations of the SNOOZE mode sequencer (Page 971)

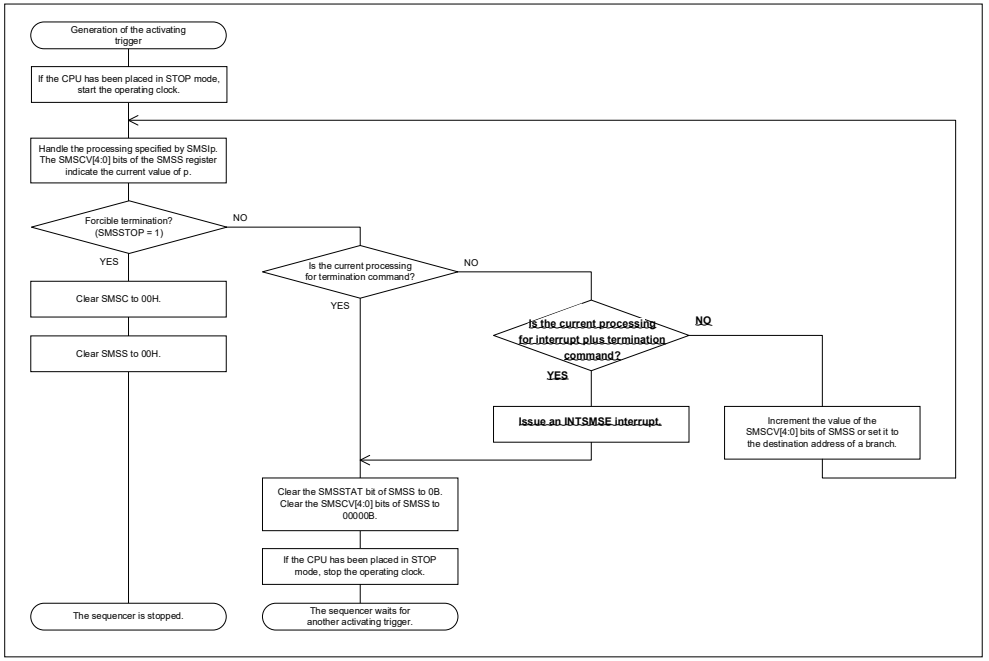
Incorrect:

26.4.1 Internal operations of the SNOOZE mode sequencer

Sequencing by the SNOOZE mode sequencer starts in response to the activating trigger specified by the SMSTRGSEL[3:0] bits of the SMSC register. Following activation, the sequencer handles the processing specified by the SMSI0 register, and then handles the processing specified by the SMSIp register indicated by the SMSCV[4:0] bits of the SMSS register. After execution of the processing for termination command or interrupt plus termination command, the sequencer has finished one round of processing and waits for another activating trigger. Moreover, setting the SMSSTOP bit of the SMSC register to 1 to generate a trigger for forcible termination leads to the sequencer being stopped.

Figure 26 - 8 shows the flow of internal operations of the SNOOZE mode sequencer.

Figure 26 - 8 Flow of Internal Operations of the SNOOZE Mode Sequencer



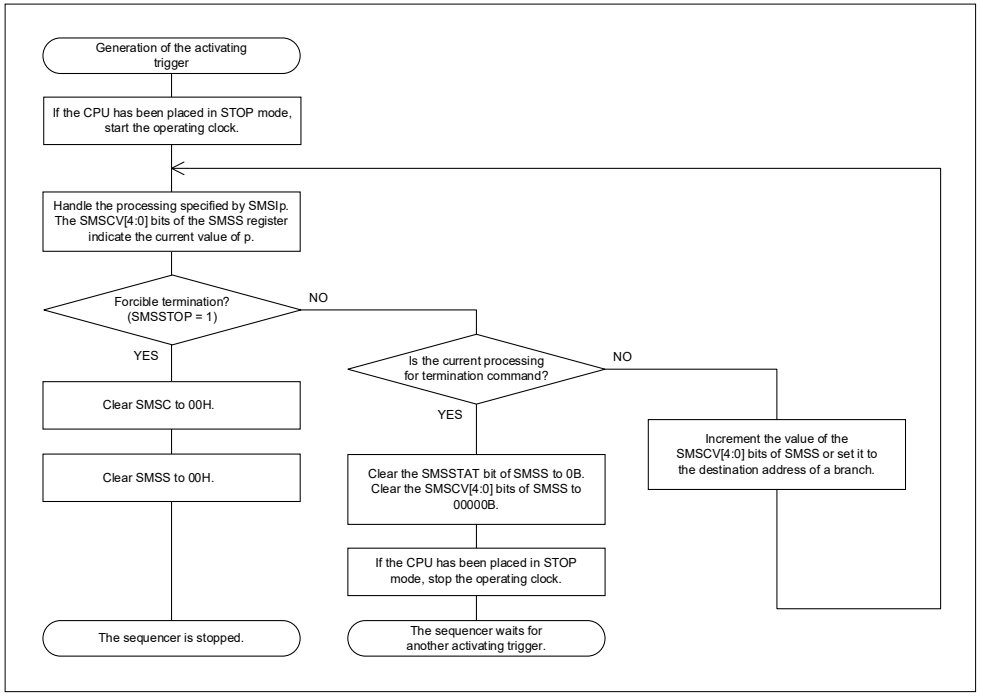
Correct:

26.4.1 Internal operations of the SNOOZE mode sequencer

Sequencing by the SNOOZE mode sequencer starts in response to the activating trigger specified by the SMSTRGSEL0 to SMSTRGSEL4 bits of the SMSC register. Following activation, the sequencer handles the processing specified by the SMSI0 register, and then handles the processing specified by the SMSIp register indicated by the SMSCV0 to SMSCV4 bits of the SMSS register. After execution of the processing for termination command, the sequencer has finished one round of processing and waits for another activating trigger. Moreover, setting the SMSSTOP bit of the SMSC register to 1 to generate a trigger for forcible termination leads to the sequencer being stopped.

Figure 26 - 8 shows the flow of internal operations of the SNOOZE mode sequencer.

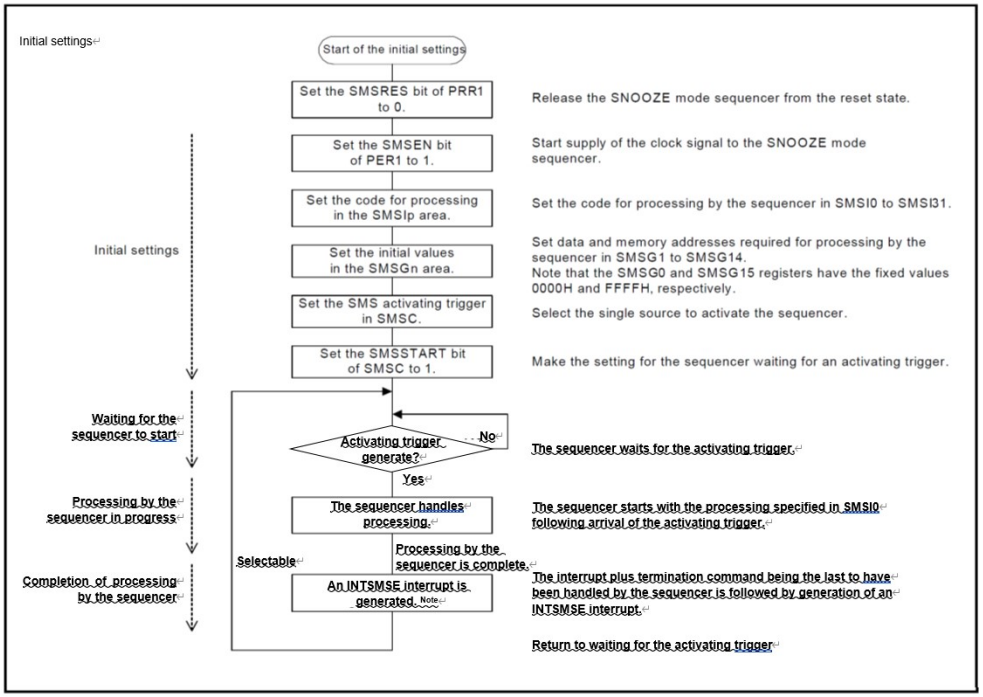
Figure 26 - 8 Flow of Internal Operations of the SNOOZE Mode Sequencer



4. 26.4.4 Procedures for running the SNOOZE mode sequencer (Page 975)

Incorrect:

Figure 26 - 11 Flow of Activating and Running the SNOOZE Mode Sequencer

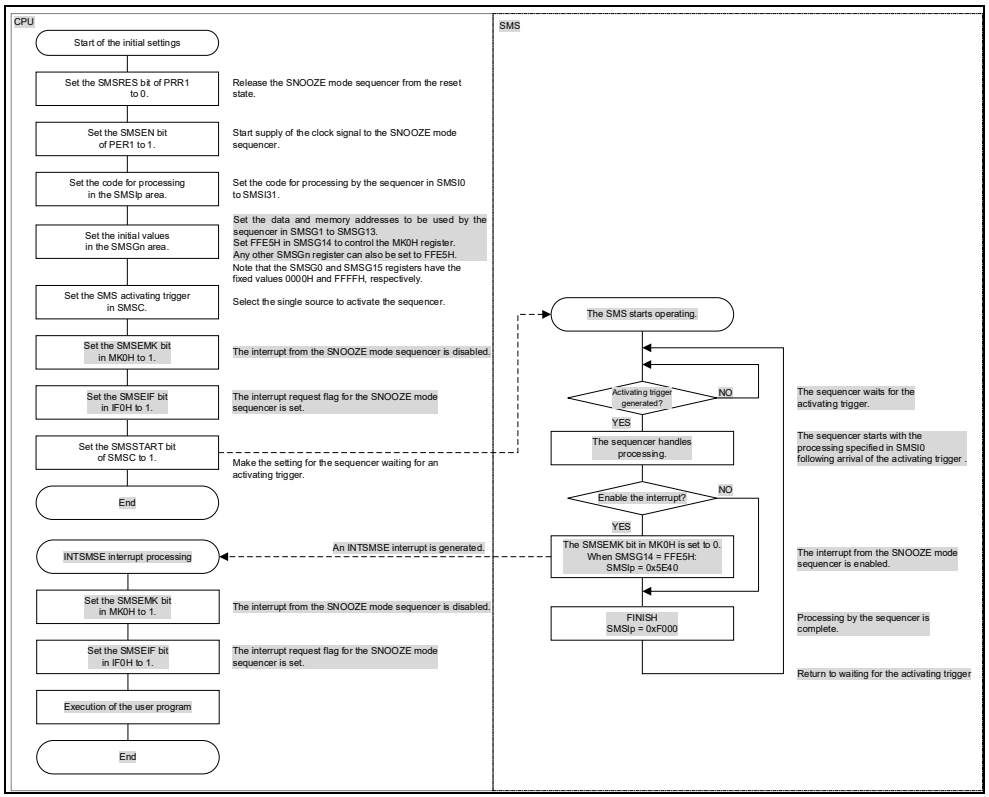


Note If processing by the sequencer ends following the processing for termination command or setting of the trigger bit for forcible termination (the SMSSTOP bit in the SMSC register), an INTSMSE interrupt will not be generated.

If processing by the sequencer ends for the latter reason (setting of the trigger bit for forcible termination), the SMSC register itself will be initialized. Therefore, to restart processing by the sequencer, make the initial settings of the SMSC register again (the SMSIp and SMSGn registers are not reset).

Correct:

Figure 26 - 11 Flow of Activating and Running the SNOOZE Mode Sequencer



Caution 1. If processing by the sequencer ends for the latter reason (setting of the trigger bit for forcible termination), the SMSC register itself will be initialized. Therefore, to restart processing by the sequencer, make the initial settings of the SMSC register again (the SMSIp and SMSGn registers are not reset).

Caution 2. Do not use a CPU instruction to set the SMSEMK bit in the MK0H register or SMSEIF bit in the IF0H register to 0 while the setting of the SMSSTART bit in the SMSC register is 1.

5. 26.4.5 States of the SNOOZE mode sequencer (Page 977)

Incorrect:

Sequencer operating state

The sequencer operating state is that in which the sequencer is operating and is handling processing specified by the SMSIp registers.

Executing the termination or interrupt plus termination command places the sequencer in the activating trigger waiting state. If operation of the sequencer is forcibly terminated by setting the SMSSTOP bit of the SMSC register to 1, the sequencer enters the sequencer stopped state.

Correct:

Sequencer operating state

The sequencer operating state is that in which the sequencer is operating and is handling processing specified by the SMSIp registers.

Executing the termination command places the sequencer in the activating trigger waiting state. If operation of the sequencer is forcibly terminated by setting the SMSSTOP bit of the SMSC register to 1, the sequencer enters the sequencer stopped state.

6. 26.5.20 Interrupt plus termination (Page 999)

Incorrect:

29.5.20 Interrupt plus termination

The interrupt plus termination command issues an interrupt signal and then stops the SNOOZE mode sequencer. Issuing the interrupt signal enables starting the CPU when it has been placed on standby. Specifically, execution of the command issues the interrupt signal, stops the SNOOZE mode sequencer, clears the SMSSTAT and SMSCV[4:0] bits in the SMSS register to 0, and places the sequencer in the activating trigger waiting state.

Set the additional byte to 0001B. Set all bits of the first and second operands to 0.

Sequencer code: 1111B (additional byte: 0001B)

Number of clock cycles for processing: 1 cycle of fCLK

Flags: The states of the SZ and SCY flags are retained.

Equivalent CPU command: WAKEUP

Equivalent CPU operation: SMSS[4:0] ← 0, stopping the sequencer after issuing an interrupt

Symbol	15	14	13	12	11	10	9	8
SMSIp	1	1	1	1	0	0	0	0

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	1

Example of a statement: 1111 0000 0000 0001 B

The equivalent CPU command in this case is WAKEUP.

The interrupt plus termination command stops the sequencer after issuing an INTSMSE interrupt, clears the SMSSTAT and SMSCV[4:0] bits of the SMSS register to 0, and places the sequencer in the activating trigger waiting state.

Correct:

7. 26.6 Operation in Standby Modes (Page 1001)

Incorrect:

26.6 Operation in Standby Modes

State	Operation of the SNOOZE Mode Sequencer
HALT mode	Operation continues.Note 1
STOP mode	The activating trigger for the SNOOZE mode sequencer can be accepted.Note 3
SNOOZE mode	Operation continues.Notes 2, 4, 5, 6

Note 1. When the subsystem clock is selected as fCLK, operation is disabled if the RTCLPC bit of the OSMC register is 1.

Note 2. The SNOOZE mode can only be set when the high-speed on-chip oscillator clock or middle-speed on-chip oscillator clock is selected as fCLK.

Note 3. Detection of an SMS activating trigger in STOP mode places the chip in SNOOZE mode, making the SNOOZE mode sequencer capable of operation. The state of the chip returns to the STOP mode after the operations of the SMS are completed. Note that the sequencer does not have access to certain memory areas in SNOOZE mode. For details, see 26.4.2 Memory space allocated to the sequencer.

Note 4. When a transfer end interrupt from the CSIp in SNOOZE mode is being used as the activating trigger for the SNOOZE mode sequencer, use the interrupt plus termination command to release the chip from the SNOOZE mode and start processing by the CPU, or make the settings for reception by the CSIp (writing 1 to the STm0 bit, writing 0 to the SWCm bit, setting the SSCm register, and writing 1 to the SSm0 bit) again before the processing for termination.

Note 5. When a transfer end interrupt from the UARTq in SNOOZE mode is being used as the activating trigger for the SNOOZE mode sequencer, use the interrupt plus termination command to release the chip from the SNOOZE mode and start processing by the CPU, or make the settings for reception by the UARTq (writing 1 to the STm1 bit, writing 0 to the SWCm bit, setting the SSCm register, and writing 1 to the SSm1 bit) again before the processing for termination.

Note 6. When an A/D conversion end interrupt from the A/D converter in SNOOZE mode is being used as the activating trigger for the SNOOZE mode sequencer, use the interrupt plus termination command to release the chip from the SNOOZE mode and start processing by the CPU, or make the settings for the SNOOZE mode function of the A/D converter (writing 1 to the AWC bit after having written 0 to it) again before the processing for termination.

Correct:

26.6 Operation in Standby Modes

State	Operation of the SNOOZE Mode Sequencer
HALT mode	Operation continues.Note 1
STOP mode	The activating trigger for the SNOOZE mode sequencer can be accepted.Note 3
SNOOZE mode	Operation continues.Notes 2, 4, 5, 6

Note 1. When the subsystem clock is selected as fCLK, operation is disabled if the RTCLPC bit of the OSMC register is 1.

Note 2. The SNOOZE mode can only be set when the high-speed on-chip oscillator clock or middle-speed on-chip oscillator clock is selected as fCLK.

Note 3. Detection of an SMS activating trigger in STOP mode places the chip in SNOOZE mode, making the SNOOZE mode sequencer capable of operation. The state of the chip returns to the STOP mode after the operations of the SMS are completed. Note that the sequencer does not have access to certain memory areas in SNOOZE mode. For details, see 26.4.2 Memory space allocated to the sequencer.

Note 4. When a transfer end interrupt from CSI00 is being used as the activating trigger for the SNOOZE mode sequencer but the transfer end interrupt is disabled (CSIMK = 1), proceed with the following steps before the processing for termination of the SNOOZE mode sequencer. Write 0 to the SMSEMK bit in the MK0H register, and release the chip from the SNOOZE mode to start processing by the CPU, or make the settings for reception by CSI00 (writing 1 to the ST00 bit, writing 0 to the SWC0 bit, setting the SSC0 register, and writing 1 to the SS00 bit) again.

Note 5. When a transfer end interrupt from UART0 is being used as the activating trigger for the SNOOZE mode sequencer, but the transfer end interrupt is disabled (SRMK0 = 1), proceed with the following steps before the processing for termination of the SNOOZE mode sequencer. Write 0 to the SMSEMK bit in the MK0H register, and release the chip from the SNOOZE mode to start processing by the CPU, or make the settings for reception by UART0 (writing 1 to the ST01 bit, writing 0 to the SWC0 bit, setting the SSC0 register, and writing 1 to the SS01 bit) again.

Note 6. When an A/D conversion end interrupt from the A/D converter is being used as the activating trigger for the SNOOZE mode sequencer, but the A/D conversion end interrupt is disabled (ADMK = 1), proceed with the following steps before the processing for termination of the SNOOZE mode sequencer. Write 0 to the SMSEMK bit in the MK0H register, and release the chip from the SNOOZE mode to start processing by the CPU, or make the settings for the SNOOZE mode of the A/D converter (writing 1 to the AWC bit after having written 0 to it) again.

8. 34.4 AC Characteristics (Page 1164)

Incorrect:

(T_A = -40 to +105°C, 1.6 V ≤ V_{DD} ≤ 5.5 V, V_{SS} = 0 V)

Item	Symbol	Conditions			Min.	Typ.	Max.	Unit
Instruction cycle (minimum instruction execution time)	T_{CY}	Main system clock (f_{MAIN}) operation	HS (high-speed main) mode	1.8 V ≤ V_{DD} ≤ 5.5 V	0.03125		1	µs
				1.6 V ≤ V_{DD} ≤ 1.8 V	0.25		1	µs
			LS (low-speed main) mode	1.8 V ≤ V_{DD} ≤ 5.5 V	0.04167		1	µs
				1.6 V ≤ V_{DD} ≤ 1.8 V	0.25		1	µs
			LP (low-power main) mode	1.6 V ≤ V_{DD} ≤ 5.5 V	0.5		1	µs
		Subsystem clock (f_{SUB}) operation		1.6 V ≤ V_{DD} ≤ 5.5 V	26.041	30.5	31.3	µs
		In the self-programming mode	HS (high-speed main) mode	1.8 V ≤ V_{DD} ≤ 5.5 V	0.03125		1	µs
			LS (low-speed main) mode	1.8 V ≤ V_{DD} ≤ 5.5 V	0.04167		1	µs
External system clock frequency	f_{EX}	1.8 V ≤ V_{DD} ≤ 5.5 V			1.0		20.0	MHz
		1.6 V ≤ V_{DD} < 1.8 V			1.0		4.0	MHz
	f_{EXS}				32		38.4	kHz
External system clock input high-level width, low-level width	t_{EXH}, t_{EXL}	1.8 V ≤ V_{DD} ≤ 5.5 V			15			ns
		1.6 V ≤ V_{DD} < 1.8 V			120			ns
	t_{EXHS}, t_{EXLS}				13.7			µs

Correct:

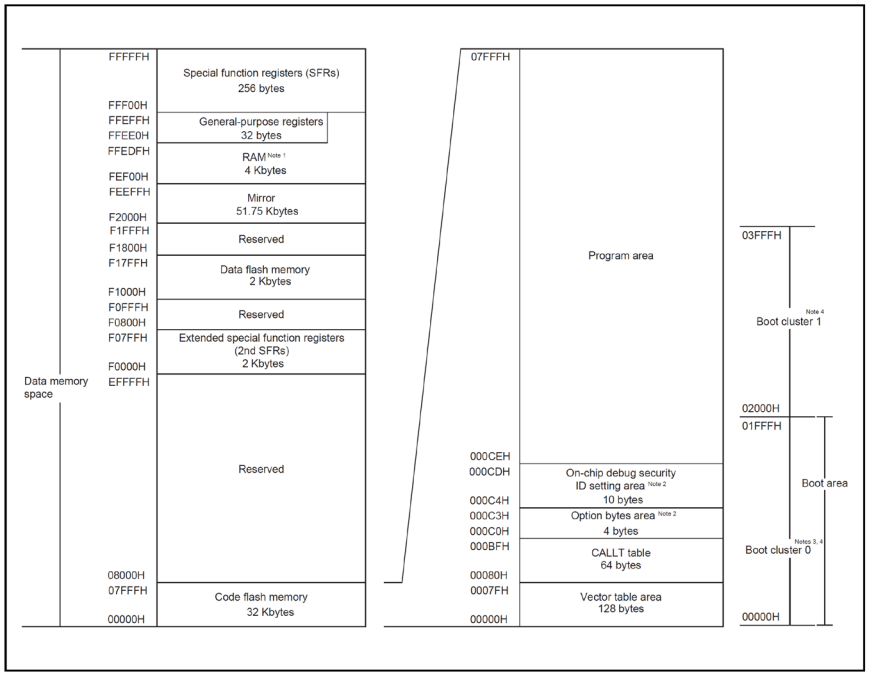
(T_A = -40 to +105°C, 1.6 V ≤ V_{DD} ≤ 5.5 V, V_{SS} = 0 V)

Item	Symbol	Conditions			Min.	Typ.	Max.	Unit
Instruction cycle (minimum instruction execution time)	T_{CY}	Main system clock (f_{MAIN}) operation	HS (high-speed main) mode	1.8 V ≤ V_{DD} ≤ 5.5 V	0.03125		1	µs
				1.6 V ≤ V_{DD} ≤ 1.8 V	0.25		1	µs
			LS (low-speed main) mode	1.8 V ≤ V_{DD} ≤ 5.5 V	0.04167		1	µs
				1.6 V ≤ V_{DD} ≤ 1.8 V	0.25		1	µs
			LP (low-power main) mode	1.6 V ≤ V_{DD} ≤ 5.5 V	0.5		1	µs
		Subsystem clock (f_{SUB}) operation		1.6 V ≤ V_{DD} ≤ 5.5 V	26.041	30.5	31.3	µs
		In the self-programming mode	HS (high-speed main) mode	1.8 V ≤ V_{DD} ≤ 5.5 V	0.03125		1	µs
			LS (low-speed main) mode	1.8 V ≤ V_{DD} ≤ 5.5 V	0.04167		1	µs
External system clock frequency	f_{EX}	1.8 V ≤ V_{DD} ≤ 5.5 V			1.0		20.0	MHz
		1.6 V ≤ V_{DD} < 1.8 V			1.0		4.0	MHz
	f_{EXS}				32		38.4	kHz
External system clock input high-level width, low-level width	t_{EXH}, t_{EXL}	1.8 V ≤ V_{DD} ≤ 5.5 V			24			ns
		1.6 V ≤ V_{DD} < 1.8 V			120			ns
	t_{EXHS}, t_{EXLS}				13.7			µs

9. 3.1 Memory Space (Page 94, Page 95,Page 100)

Incorrect:
(Page 94)
Products in the RL78/G22 can access a 1 MB address space. Figures 3 - 1 and 3 - 2 show the memory maps.

Figure 3 - 1 Memory Map (R7F102GxC (x = 4, 6, 7, 8, A, B, C, E, F, G))



Note 1. Instructions can be executed from the RAM area excluding the general-purpose register area.

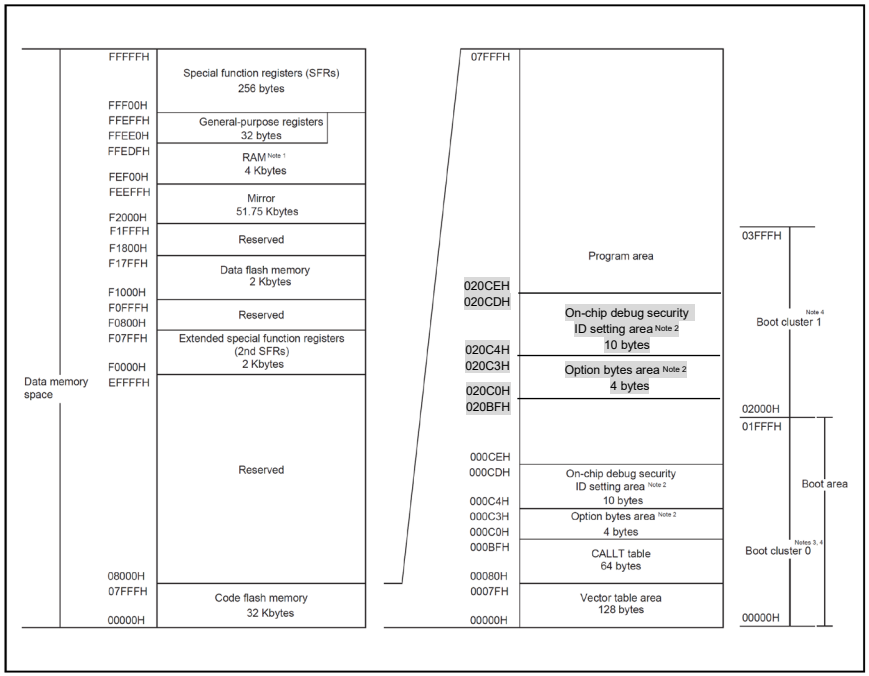
Note 2. **When boot swap is not used**: Set the option bytes to 000C0H to 000C3H, and the on-chip debug security IDs to 000C4H to 000CDH.
When boot swap is used: Set the option bytes to 000C0H to 000C3H and 020C0H to 020C3H, and the on-chip debug security IDs to 000C4H to 000CDH and 020C4H to 020CDH.

(omitted)

Correct:

Products in the RL78/G22 can access a 1 MB address space. Figures 3 - 1 and 3 - 2 show the memory maps.

Figure 3 - 1 Memory Map (R7F102GxC (x = 4, 6, 7, 8, A, B, C, E, F, G))



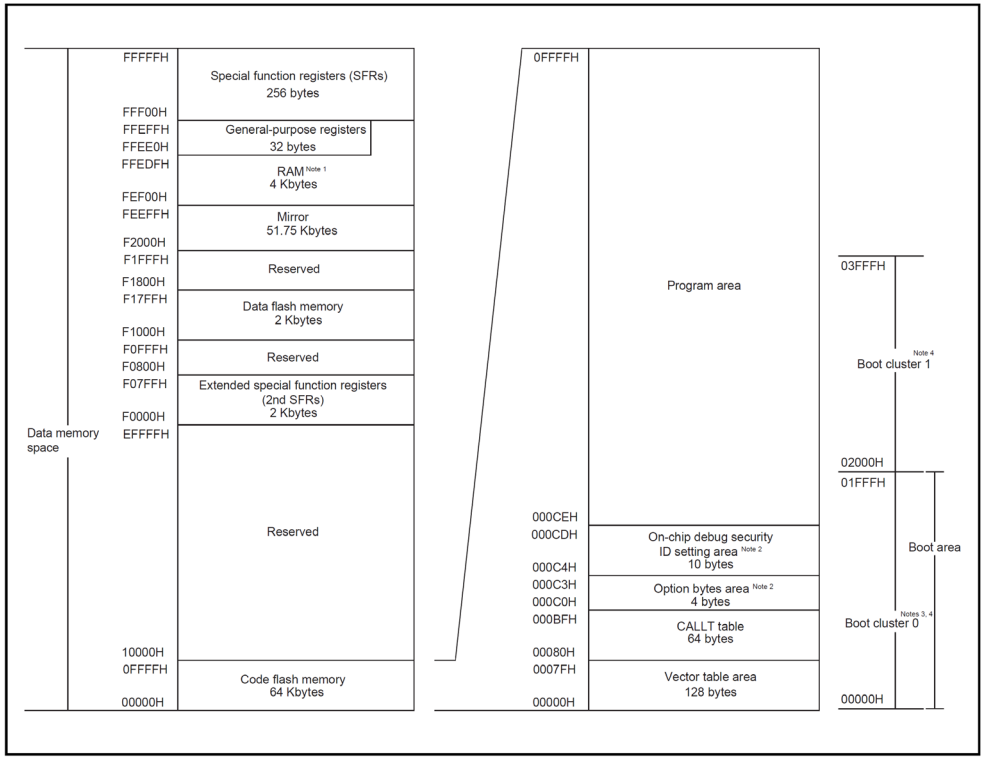
Note 1. Instructions can be executed from the RAM area excluding the general-purpose register area.

Note 2. When boot swapping is not to be used, that is, when the value of the BTFLG bit in the FLSEC register is 1, set the option bytes to 000C0H to 000C3H, and the on-chip debug security IDs to 000C4H to 000CDH.
When boot swapping is to be used or the value of the BTFLG bit in the FLSEC register is 0, set the option bytes to 000C0H to 000C3H and 020C0H to 020C3H, and the on-chip debug security IDs to 000C4H to 000CDH and 020C4H to 020CDH.

(omitted)

(Page 95)

Figure 3 - 2 Memory Map (R7F102GxE (x = 4, 6, 7, 8, A, B, C, E, F, G))

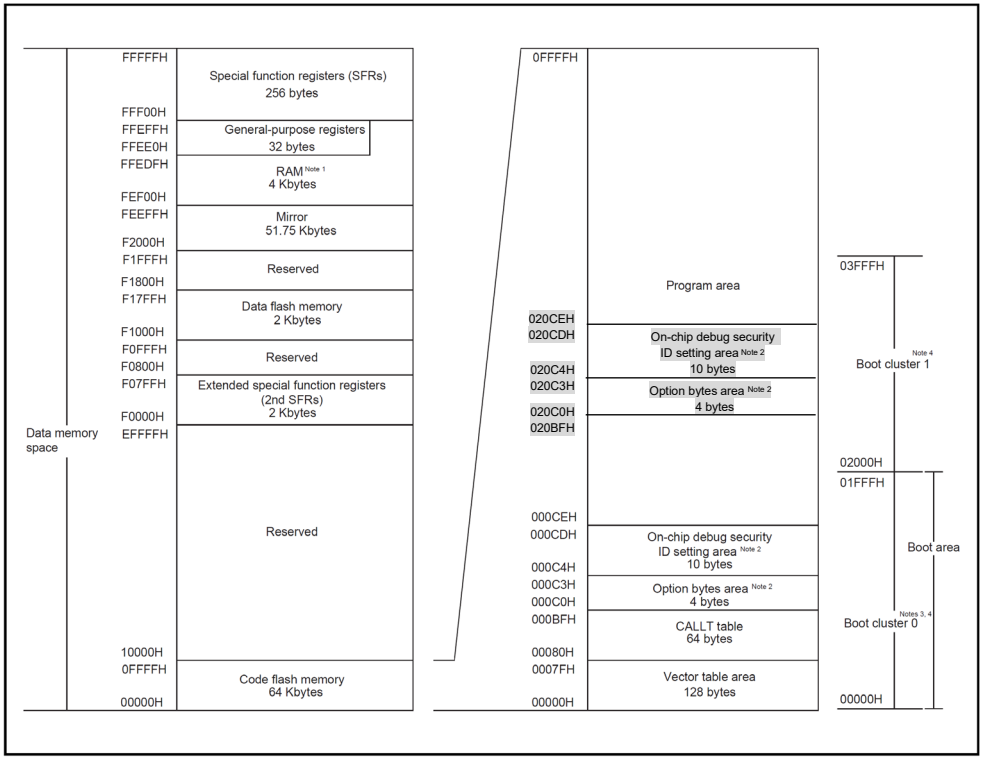


Note 1. Instructions can be executed from the RAM area excluding the general-purpose register area.

Note 2. **When boot swap is not used**: Set the option bytes to 000C0H to 000C3H, and the on-chip debug security IDs to 000C4H to 000CDH.
When boot swap is used: Set the option bytes to 000C0H to 000C3H and 020C0H to 020C3H, and the on-chip debug security IDs to 000C4H to 000CDH and 020C4H to 020CDH.

(omitted)

Figure 3 - 2 Memory Map (R7F102GxE (x = 4, 6, 7, 8, A, B, C, E, F, G))



Note 1. Instructions can be executed from the RAM area excluding the general-purpose register area.

Note 2. When boot swapping is not to be used, that is, when the value of the BTFLG bit in the FLSEC register is 1, set the option bytes to 000C0H to 000C3H, and the on-chip debug security IDs to 000C4H to 000CDH.
When boot swapping is to be used or the value of the BTFLG bit in the FLSEC register is 0, set the option bytes to 000C0H to 000C3H and 020C0H to 020C3H, and the on-chip debug security IDs to 000C4H to 000CDH and 020C4H to 020CDH.

(omitted)

(Page 100)

(omitted)

(3) Option bytes area

A 4-byte area of 000C0H to 000C3H can be used as an option bytes area. Set the option byte at 020C0H to 020C3H when **the boot swap is used**. For details, see Section 29 Option Bytes.

(4) On-chip debug security ID setting area

A 10-byte area of 000C4H to 000CDH and 020C4H to 020CDH can be used as an on-chip debug security ID setting area. Set the on-chip debug security ID of 10 bytes at 000C4H to 000CDH when **the boot swap is not used** and at 000C4H to 000CDH and at 020C4H to 020CDH when **the boot swap is used**. For details, see Section 31 On-chip Debugging.

(omitted)

(3) Option bytes area

A 4-byte area of 000C0H to 000C3H can be used as an option bytes area. Set the option byte at 040C0H to 040C3H when boot swapping is to be used or the value of the BTFLG bit in the FLSEC register is 0. For details, see Section 29 Option Bytes.

(4) On-chip debug security ID setting area

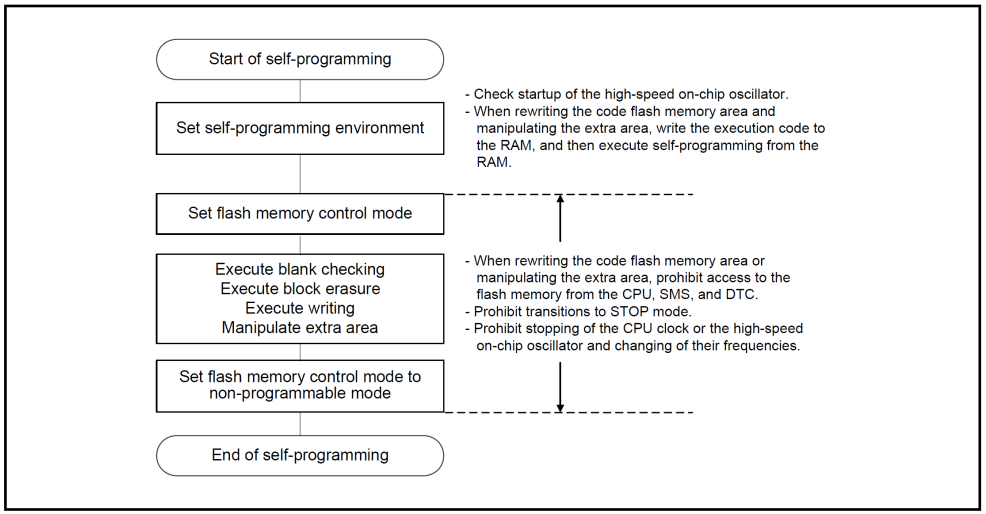
A 10-byte area of 000C4H to 000CDH and 020C4H to 020CDH can be used as an on-chip debug security ID setting area. Set the 10-byte security ID for on-chip debugging at 000C4H to 000CDH when boot swapping is not to be used, that is, the value of the BTFLG bit in the FLSEC register is 1, and at both 000C4H to 000CDH and 020C4H to 020CDH when boot swapping is to be used or the value of the BTFLG bit in the FLSEC register is 0. For details, see Section 31 On-chip Debugging.

10. 30.6.1 Self-programming procedure (Page 1065)

Incorrect:

The following figure illustrates a flow for rewriting the flash memory by using self-programming.
For details on registers for use in self-programming, see 30.6.2 Registers for controlling the flash memory.

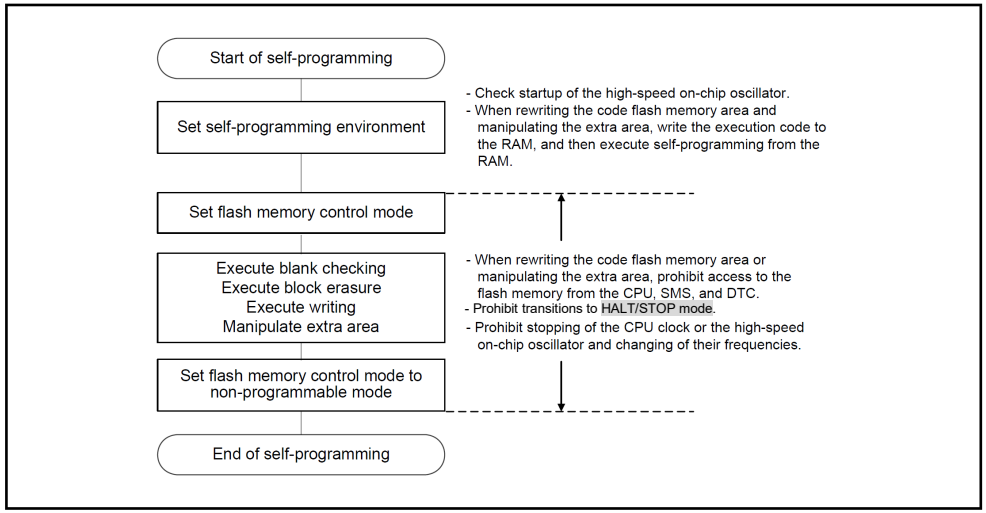
Figure 30 - 8 Flow of Self-Programming (Rewriting the Flash Memory)



Correct:

The following figure illustrates a flow for rewriting the flash memory by using self-programming.
For details on registers for use in self-programming, see 30.6.2 Registers for controlling the flash memory.

Figure 30 - 8 Flow of Self-Programming (Rewriting the Flash Memory)



11. 30.10.1 Overview of the data flash memory (Page 1116)

Incorrect:

An overview of the data flash memory is provided below.

(omitted)

- Manipulating the DFLCTL register is prohibited while rewriting the data flash memory.
- Transition to the **STOP mode** is prohibited while rewriting the data flash memory.

Correct:

An overview of the data flash memory is provided below.

(omitted)

- Manipulating the DFLCTL register is prohibited while rewriting the data flash memory.
- Transition to the HALT/STOP mode is prohibited while rewriting the data flash memory.

12. 31.3 Security Settings for On-Chip Debugging (Page 1119)

Incorrect:

To protect against third parties reading the contents of memory, on-chip debugging includes the following functionality.

– Disabling of connection between the RL78 microcontroller and the programmer or on-chip debugger (see 30.9 Security Settings in Section 30 Flash Memory).
– On-chip debugging control bits in the flash memory at 000C3H (see Section 29 Option Bytes)
– An area in the range from 000C4H to 000CDH to hold the security ID code for on-chip debugging.Note 1

Table 31 - 1 On-Chip Debug Security ID

Address	Security ID Code for On-Chip Debugging
000C4H to 000CDH	Any 10-byte ID code Note 2
020C4H to 020CDH	

Note 1. The area to hold the security ID code for use in on-chip debugging is also used to hold the ID code for the programmer connection ID when a programmer is to be used.

Note 2. The setting FFFFFFFFFFFFFFFFFFFFH is not allowed.

Correct:

To protect against third parties reading the contents of memory, on-chip debugging includes the following functionality.

– Disabling of connection between the RL78 microcontroller and the programmer or on-chip debugger (see 30.9 Security Settings in Section 30 Flash Memory).
– On-chip debugging control bits in the flash memory at 000C3H (see Section 29 Option Bytes)
– An area in the range from 000C4H to 000CDH to hold the security ID code for on-chip debugging.Note

Note. The area to hold the security ID code for use in on-chip debugging is also used to hold the ID code for the programmer connection ID when a programmer is to be used.

Table 31 - 1 On-Chip Debug Security ID

Address	Security ID Code for On-Chip Debugging
000C4H to 000CDH	Any 10-byte ID code Note 1, 2
020C4H to 020CDH	

Note 1. The setting FFFFFFFFFFFFFFFFFFFFH is not allowed.

Note 2. Set the 10-byte security ID for on-chip debugging at both 000C4H to 000CDH and 020C4H to 020CDH when boot swapping is to be used or the value of the BTFLG bit in the FLSEC register is 0.