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.

SH7618 Group

HIF Boot Mode

Introduction

The SH7618 integrates an Ethernet controller module and HIF (Host Interface) module, which makes connection to another microcomputer easier, with the RISC-type SH-2 CPU core (RISC: Reduced Instruction Set Computer).

The HIF module facilitates the transfer of data between devices connected to the HIF. A feature of the HIF is the HIF boot mode, in which the SH7618 is started up by a boot program stored in the HIFRAM (internal RAM used exclusively by the HIF). When the boot program is used as a data transfer program, an external device can transfer application programs to the SH7618 via the HIFRAM.

A transferred application program is stored in SH7618 internal RAM or RAM connected to the SH7618 external bus. Transferring application programs from an external device to the SH7618 eliminates the need for program storage ROM for the SH7618.

This application note describes the startup method in the HIF boot mode, and therefore may be useful for designing user software.

Although the operation of the programs described in this application note has been verified, always conduct an operation check in your operating environment before using them.

Target Device

SH7618

Contents

1. Overview of the HIF

1.1 HIF Features

1.1.1 HIF and HIFRAM Features

The HIF and HIFRAM have the following features:

- The HIF pins can be set to a high impedance state by inputting a low level to the HIF enable (HIFEBL) pin. By placing the HIF pins in the high impedance state, you can prevent the device from being damaged as a result of the device being driven by the HIF pins while the HIF is in the module standby mode or an external device is in the standby mode.
- The SH7618 contains two 1-KB banks of HIFRAM accessible from the CPU and from an external device connected to the HIF. Since the CPU and HIF can concurrently access each bank, efficient data transfer is enabled.
- By using the HIF registers, you can set individual HIFRAM banks to be accessed from the SH7618 CPU and the external device.
- Because the HIF has its own bus separate from the SH bus, an external device connected to the HIF can access HIFRAM asynchronously with SH7618 operation (that is, the SH7618 does not have to release the SH bus).
- An external device connected to the HIF can access HIFRAM via the HIF in 32 bits.
- The SH7618 can access the HIFRAM in 8-, 16-, or 32-bit units.
- The HIF boot mode allows the SH7618 to be started up by a program in the HIFRAM.

The HIFRAM consists of two banks that have the same address mapping. When an external device connected to the HIF accesses the HIFRAM, the target HIFRAM address should be specified in the HIFADR register. Table 1 describes HIFRAM address mapping.

Table 1 HIFRAM Address Mapping

Type	Start Address	End Address	Size
Mapping as viewed from the SH7618 CPU	H'F84E0000	H'F84E03FF	1 KB
Mapping as viewed from an external device*	H'0000	H'03FF	1 KB

Note: *The value set by the external device in the HIFADR register.

1.1.2 Connection to an External Device

Table 2 shows the HIF pin configuration.

Table 2 HIF Pin Configuration

Name	Abbreviation	I/O	Description
HIF data pins	HIFD15 to HIFD0	I/O	Address, data, or command input to or output from the HIF.
HIF chip select	$\overline{\text{HIFCS}}$	Input	Chip select input to the HIF.
HIF register select	HIFRS	Input	Switching between HIF access types: 0: Accesses the register specified in the HIF index register. 1: Writes to the HIF index register or reads the HIF general status register.
HIF write	$\overline{\text{HIFWR}}$	Input	Write strobe signal. A low level is input when an external device writes data to the HIF.
HIF read	$\overline{\text{HIFRD}}$	Input	Read strobe signal. A low level is input when an external device reads data from the HIF.
HIF interrupt	$\overline{\text{HIFINT}}$	Output	Interrupt request to an external device from the HIF.
HIF mode	HIFMD	Input	Selects whether the SH7618 is started up in the HIF boot mode. If a power-on reset is released while a high level is input, the SH7618 is started up in the HIF boot mode.
HIFDMAC transfer request	HIFDREQ	Output	Requests that the external device perform a DMA (Direct Memory Access) to HIFRAM.
HIF boot ready	HIFRDY	Output	When this pin is asserted, indicates that an HIF reset is canceled and that access from an external device to the HIF can be accepted. After 10 clock cycles (max.) of the peripheral clock following negation of the reset input pin of the SH7618, this pin is asserted.
HIF pin enable	HIFEBL	Input	Selects whether to enable or disable the HIF pins. 0: Disables the HIF pins (high impedance state). 1: Enables the HIF pins (the HIF pins are available).

Figure 1 shows an example of connection between the HIF and an external device.

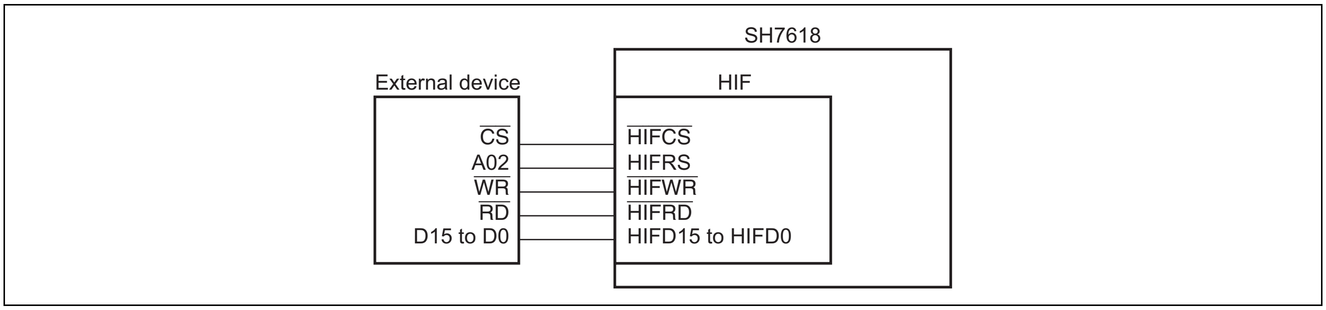


Figure 1 Example of Connection between an External Device and the HIF

1.2 Access from an External Device

1.2.1 Access to the HIF Registers

The HIF has the following registers:

- HIF Index Register (HIFIDX)
- HIF General Status Register (HIFGSR)
- HIF Status/Control Register (HIFSCR)
- HIF Memory Control Register (HIFMCR)
- HIF Internal Interrupt Control Register (HIFIICR)
- HIF External Interrupt Control Register (HIFEICR)
- HIF Address Register (HIFADR)
- HIF Data Register (HIFDATA)
- HIF Boot Control Register (HIFBCR)
- HIFDREQ Rrigger Register (HIFDTR)
- HIF Bank Interrupt Control Register (HIFBICR)

When an external device accesses an HIF register, the HIFIDX register is used to specify the target register and the word position (upper or lower two bytes). Whether to access the HIFIDX register or the register specified by the HIFIDX register is selected by the HIFRS pin setting.

Table 3 shows the combinations of the $\overline{\text{HIFCS}}$, HIFRS, $\overline{\text{HIFWR}}$, and $\overline{\text{HIFRD}}$ pins, and the corresponding operations.

Table 3 HIF Operations

$\overline{\text{HIFCS}}$	HIFRS	$\overline{\text{HIFWR}}$	$\overline{\text{HIFRD}}$	Operation
1	—	—	—	No operation (NOP)
0	0	1	0	Read from the register specified by HIFIDX
0	0	0	1	Write to the register specified by HIFIDX
0	1	1	0	Read from HIFGSR
0	1	0	1	Write to HIFIDX
0	—	1	1	No operation (NOP)
0	—	0	0	Setting prohibited

Note: —: Don't Care

1.2.2 Writing to HIFRAM

The following describes the sequence of operations when an external device writes to HIFRAM. Figure 2 illustrates the sequence.

(1) The HIFADR register is used to specify the HIFRAM address to which data is to be written.
The HIFIDX register is used to specify the lower 16 bits of the HIFADR register for indicating the HIFRAM address to which data is written.

(2) Data is written to the HIFDATA register.
Data is written by using the HIFIDX register to specify the upper 16 bits of the HIFDATA register. Then the HIFDATA register is accessed and data is written to the lower 16 bits of the register.

(3) The WT bit of the HIFMCR register is set to 1.
When the WT bit of the HIFMCR register is set to 1, the 32-bit data in the HIFDATA register is written to the HIFRAM address specified by the HIFADR register. After the data is written, the WT bit is automatically cleared to 0.

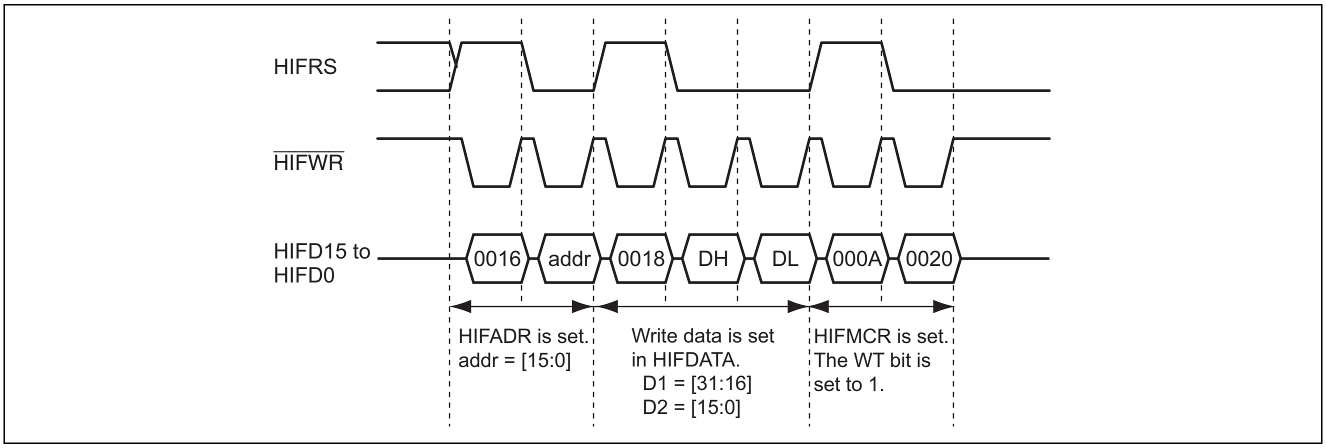


Figure 2 Sequence of a Write to HIFRAM

It is possible to write data to the consecutive addresses that follow the specified HIFRAM address. The following describes the sequence of operations when data is written to consecutive addresses. Figure 3 illustrates the sequence.

(1) The HIFADR register is used to specify the start address for writing to HIFRAM.
The HIFIDX register is used to specify the start address for writing to HIFRAM in the lower 16 bits of the HIFADR register.

(2) The first data is written to the HIFDATA register.
Data is written by using the HIFIDX register to specify the upper 16 bits of the HIFDATA register. Then the HIFDATA register is accessed and data is written to the lower 16 bits of the register.

(3) Writing to consecutive HIFRAM addresses is enabled.
Writing to consecutive HIFRAM addresses is enabled by setting both the WT bit and the LOCK bit of the HIFMCR register to 1 at the same time. The AI/AD bit of the HIFMCR register is used to select whether to increment or decrement the HIFRAM address by four.

(4) The HIFIDX register is used to specify the upper 16 bits of the HIFDATA register.

(5) Data is written to the HIFDATA register.
Each time the HIFDATA register is accessed, the HIFADR register values change according to the AI/AD bit of the HIFMCR register. In this way, data is written to consecutive HIFRAM addresses.

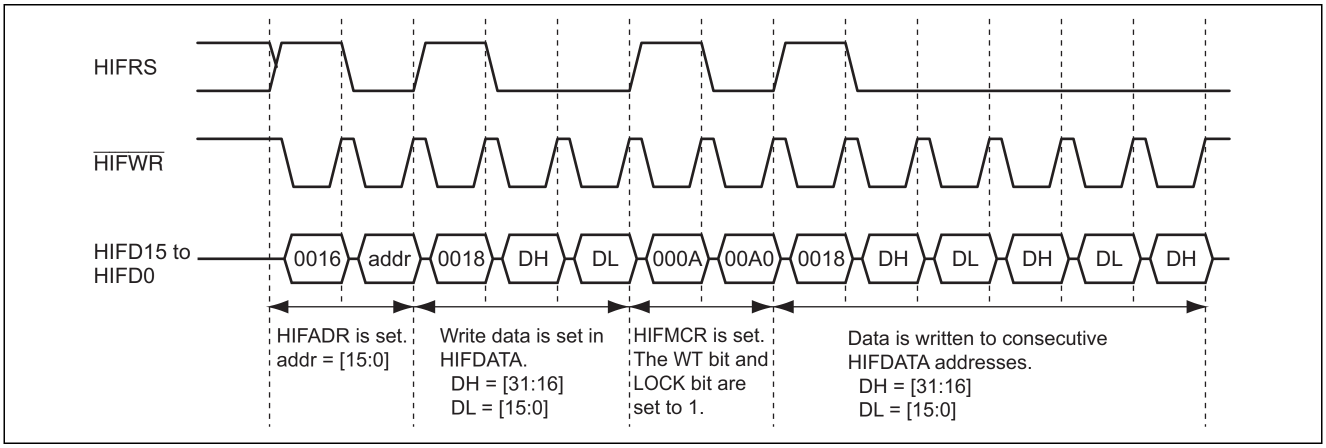


Figure 3 Sequence of Writing to Consecutive HIFRAM Addresses

1.2.3 Reading from HIFRAM

The following describes the sequence of operations when data is read from HIFRAM. Figure 4 illustrates the sequence.

(1) The HIFADR register is used to specify the HIFRAM address from which data is to be read.
The HIFIDX register is used to specify the lower 16 bits of the HIFADR register to indicate the HIFRAM address from which data is to be read.

(2) The RD bit of the HIFMCR register is set to 1.
When the RD bit of the HIFMCR register is set to 1, the data at the HIFRAM address corresponding to the address specified by the HIFADR register is fetched to the HIFDATA register. After the data is read from the HIFRAM, the RD bit is automatically cleared to 0.

(3) Data is read from the HIFDATA register.
Data is read from the HIFDATA register by using the HIFIDX register to specify the upper 16 bits of the HIFDATA register. Then, data in the lower 16 bits of the HIFDATA register is read.

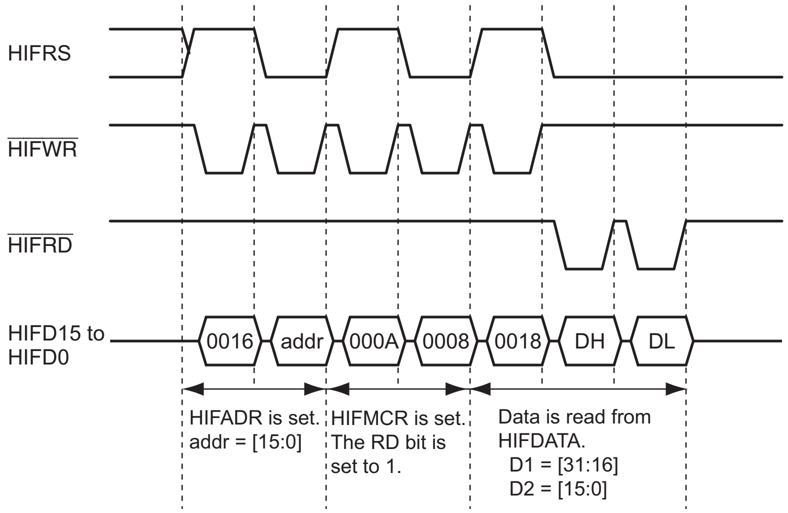


Figure 4 Sequence of a Read from HIFRAM

It is possible to read data from consecutive addresses that follow the specified HIFRAM address. The following describes the sequence of operations in which data is read from consecutive addresses. Figure 5 illustrates the sequence.

(1) The HIFADR register is used to specify the start HIFRAM address for reading.
The lower 16 bits of the HIFADR register is specified using the HIFIDX register to determine the HIFRAM address from which data is to be read.

(2) Reading from consecutive HIFRAM addresses is enabled.
Reading from consecutive HIFRAM addresses is enabled by setting both the RD bit and the LOCK bit of the HIFMCR register to 1 at the same time. The AI/AD bit of the HIFMCR register is used to select whether to increment or decrement the HIFRAM address by four.

(3) The HIFIDX register is used to specify the upper 16 bits of the HIFDATA register.

(4) Data is read from the HIFDATA register.
Each time data is read from the HIFDATA register, the HIFADR register values change according to the AI/AD bit of the HIFMCR register. In this way, data is read from consecutive HIFRAM addresses.

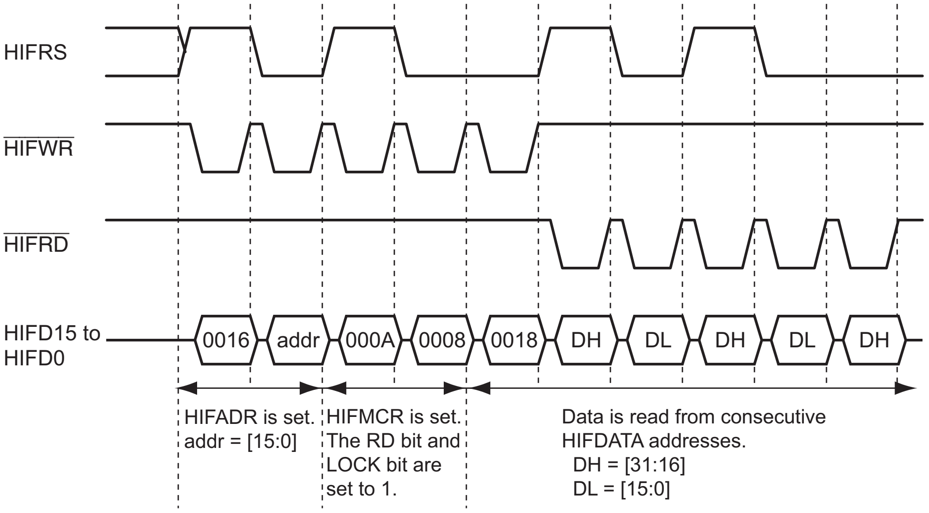


Figure 5 Sequence of Reading from Consecutive HIFRAM Addresses

1.3 HIF Boot Mode

The SH7618 provides the HIF boot mode. In this mode, the SH7618 is started up by a program in the HIFRAM.

In the HIF boot mode, the SH7618 boot program written to the HIFRAM by an external device connected to the HIF starts up the SH7618. By storing the boot program in the same ROM as external device programs, the ROM for starting up the SH7618 can be eliminated.

The following describes the sequence of operations in which the SH7618 is started up in the HIF boot mode. Figure 6 illustrates the sequence.

(1) The SH7618 is placed in HIF boot mode.
The SH7618 enters the HIF boot mode when the power-on reset state is released while the HIFMD signal is high. Note that the HIFMD signal must remain high even after the power-on reset state has been released.

(2) An external device connected to the HIF writes an SH7618 boot program to the HIFRAM.
When the SH7618 is started up in the HIF boot mode, it enters the standby state. In this state, the external device writes the SH7618 boot program to the HIFRAM. Also, when the SH7618 is activated in the HIF boot mode, the contents of the HIFRAM are mapped to the first 32 MB of area 0. Accordingly, the reset vector is written to the start address of the HIFRAM.

(3) The SH7618 is activated.
After the boot program is written to the HIFRAM, when the external device clears the AC bit of the HIFBCR register to 0, the SH7618 reads the start address of the HIFRAM as the reset vector and executes the program.

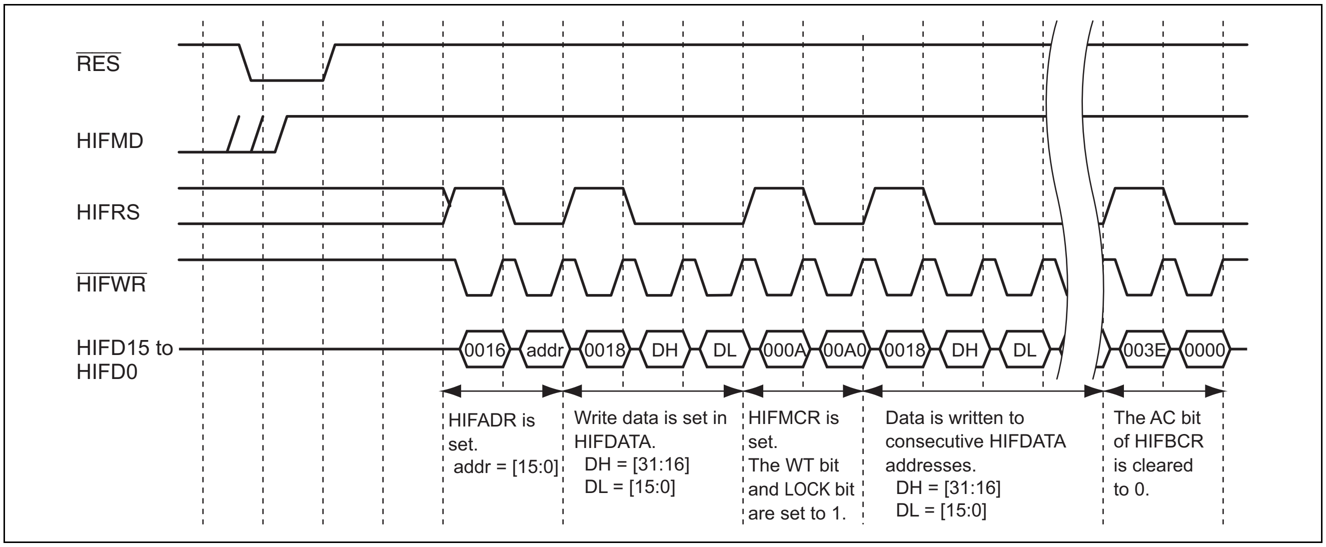
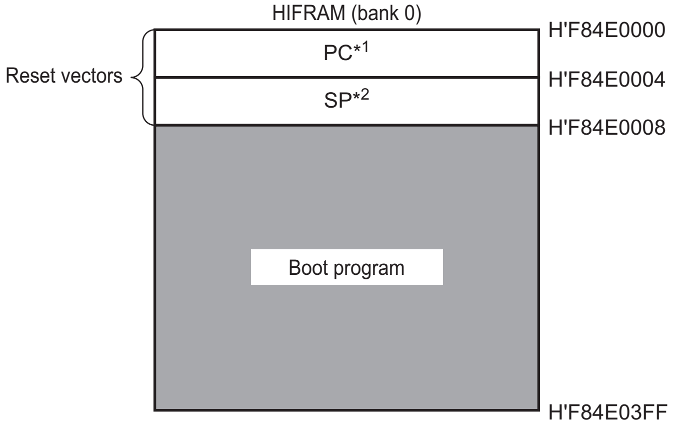

RES
HIFMD
HIFRS
HIFWR
HIFD15 to HIFD0
0016
addr
0018
DH
DL
000A
00A0
0018
DH
DL
003E
0000
HIFADR is set. addr = [15:0]
Write data is set in HIFDATA. DH = [31:16] DL = [15:0]
HIFMCR is set. The WT bit and LOCK bit are set to 1.
Data is written to consecutive HIFDATA addresses. DH = [31:16] DL = [15:0]
The AC bit of HIFBCR is cleared to 0.

Figure 6 HIF Boot Sequence

Figure 7 shows the HIFRAM memory map in the HIF boot mode.



Notes: *1. PC: Program counter
*2. SP: Stack pointer

Figure 7 HIFRAM Memory Map in HIF Boot Mode

Since the SH7618 CPU and an external device access bank 0 of the HIFRAM in the HIF boot mode, the boot program is stored in bank 0 of the HIFRAM, not bank 1. Therefore, do not change the bank to be accessed by the CPU while the program in the HIFRAM is running.

2. Examples of Using HIF Boot Mode

2.1 Startup in HIF Boot Mode

2.1.1 Overview

This section provides a sample task that starts up the SH7618 in the HIF boot mode.

In this example, an external device connected to the HIF writes a boot program to HIFRAM, and the program starts up the SH7618. Since the boot program is stored in the same ROM as external device programs, the ROM exclusively for activating the SH7618 can be eliminated.

2.1.2 Specifications

(1) The SH7641 is used as the external device connected to the HIF, and the HIF is connected to the CS5A space (H'14000000 to H'15FFFFFF) of the SH7641.
(2) Release the power-on reset while the HIFMD signal is high, so that the SH7618 enters the HIF boot mode.
(3) The SH7641 writes the boot program to the HIFRAM.
(4) After the boot program is written, the SH7641 clears the AC bit of the HIFBCR register to 0.
(5) The SH7618 reads the data at the start address of the HIFRAM as a reset vector, and starts up.
(6) In this sample task, the SH7641 operates with a 100-MHz CPU clock, a 50-MHz external bus clock, and a 25-MHz peripheral clock. Also, the SH7618 operates with a 50-MHz internal clock, a 50-MHz external bus clock, and a 12.5-MHz peripheral clock (mode 5, 25-MHz input clock), since the SH7618 in the HIF boot mode cannot set the HIF registers until the AC bit of the HIFBCR register is cleared to 0.

Figure 8 shows an example of connection of the SH7618 and the SH7641.

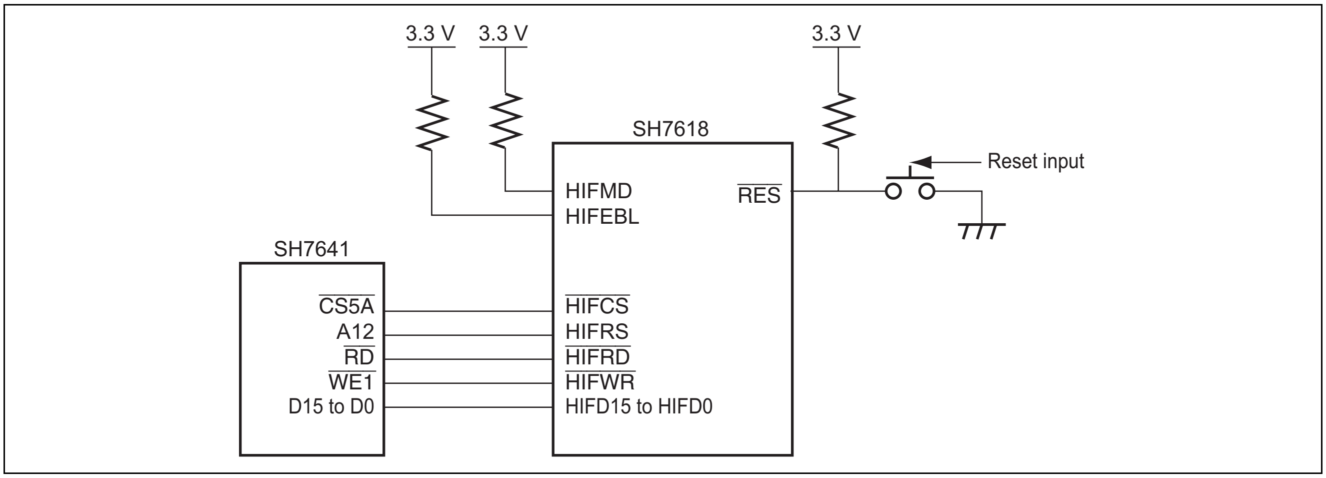


Figure 8 Connection Example

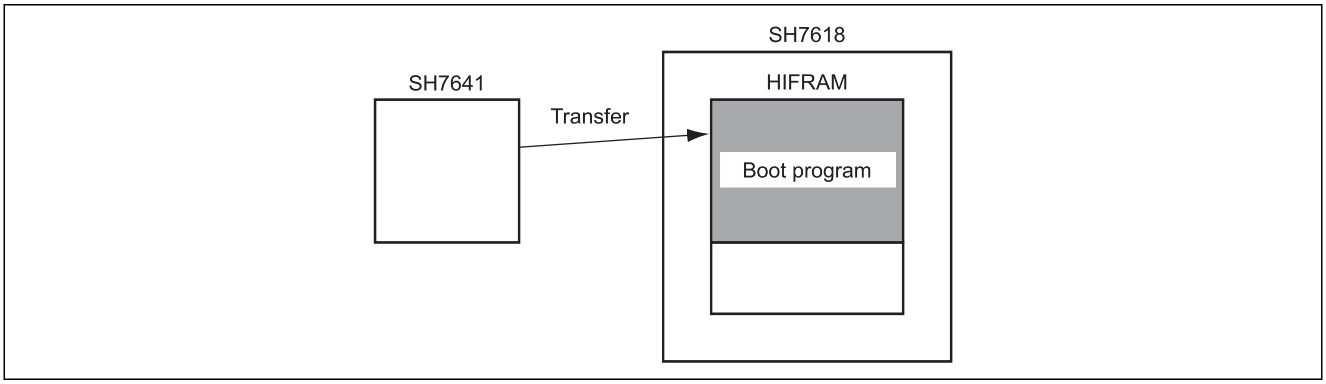
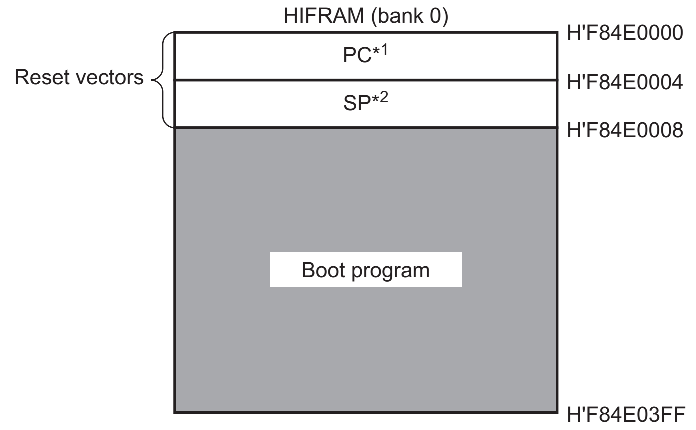


Figure 9 Startup in HIF Boot Mode

Figure 10 shows the memory map of the HIFRAM in the HIF boot mode.



Notes: *1. PC: Program counter
*2. SP: Stack pointer

Figure 10 HIFRAM Memory Map

In this sample task, the SH7618 HIF is connected to the CS5A space of the SH7641. Table 4 is a list of addresses used when the SH7641 accesses the HIF registers.

Table 4 Addresses Used When the SH7641 Accesses HIF Registers

Address	Register To Be Accessed
H'B4000000	The HIF register specified by the HIFIDX register
H'B4001000	When written: HIFIDX register When read: HIFGSR register

2.1.3 Description of Software

(1) Modules

Table 5 lists the SH7641 modules used in this sample task.

Table 5 SH7641 Modules

Module Name	Label Name	Description
SH7641 main routine	main_7641	Initial settings of the SH7641 external bus and pins.
HIFRAM consecutive write routine	write_HIFRAM	Writes a boot program to consecutive HIFRAM addresses.
SH7618 startup routine	hif_boot	Starts up the SH7618.

(2) Internal registers used

Table 6 is a list of SH7618 internal registers used in this sample task.

Table 6 SH7618 Internal Registers Used

Register Name			Setting	
	Bit	Bit Name	Value	Function
HIFADR				HIF Address Register
	31 to 10	—	All 0	Reserved
	9 to 2	A9 to A2	All 0	Specifies the target HIFRAM address. These bits specify the address, aligned on a 32-bit boundary, in HIFRAM to be accessed by an external device.
	1, 0	—	All 0	Reserved
HIFDATA				HIF Data Register
	31 to 0	D31 to D0	—	32-bit data. These bits are used for an access to HIFRAM from an external device.
HIFBCR				HIF Boot Control Register
	31 to 8	—	All 0	Reserved
	7 to 1	—	All 0	AC bit writing assistance These bits are used to write the bit pattern (H'A5) needed to set the AC bit to 1.
	0	AC	—	HIFRAM exclusive access control When the SH7618 is started up in the HIF boot mode, this bit is set to 1. After a program is written to HIFRAM, the SH7641 clears this bit to 0, and the SH7618 is activated.

Register Name		Setting	
Bit	Bit Name	Value	Function
HIFIDX			HIF Index Register
31 to 8	—	All 0	Reserved
7 to 2	REG5 to REG0	—	Select the HIF internal registers. These bits are used to select the HIF registers that will be accessed by the external device.
1 0	BYTE1 BYTE0	— —	Select a byte in an HIF internal register. These bits are used to specify the word position when an external device accesses an HIF Internal Register.
HIFMCR			HIF Memory Control Register
31 to 8	—	All 0	Reserved
7	LOCK	1	Lock bit This bit is used when the external device performs consecutive access to HIFRAM. Consecutive write to HIFRAM is enabled when both the LOCK bit and WT bit are set to 1.
6	—	0	Reserved
5	WT	1	Write bit When this bit is set to 1, the HIFDATA value is written to the HIFRAM location corresponding to HIFADR.
4	—	0	Reserved
3	RD	0	Read bit When this bit is set to 1, the HIFRAM data corresponding to HIFADR is fetched into HIFDATA.
2, 1	—	All 0	Reserved
0	AI/AD	0	Address auto-increment/auto-decrement When LOCK=1 and AI/AD=0, each time the SH7641 accesses HIFDATA, the HIFADR value is incremented (+4) so that data can be written to, or read from, consecutive HIFRAM addresses.

Table 7 is a list of SH7641 internal registers used in this sample task.

Table 7 SH7641 Internal Registers Used

Register Name		Setting	
Bit	**Bit Name**	**Value**	**Function**
CS5ABCR			CS5A Space Bus Control Register
10	BSZ1	1	Specify the data bus width for accessing the CS5A space.
9	BSZ0	0	When BSZ[1,0] = B'10, the data bus width is set to 16 bits.
CS5AWCR			CS5A Space Wait Control Register
31 to 19	—	All 0	Reserved
18	WW2	0	Number of write access wait cycles
17	WW1	0	These bits specify the number of wait cycles to be inserted during write access.
16	WW0	0	When WW[2-0] = B'000, the same number of wait cycles as the number of read access wait cycles specified by the WR bits are inserted.
15 to 13	—	All 0	Reserved
12	SW1	0	Number of delay cycles from address, $\overline{\text{CS}}$ assertion to $\overline{\text{RD}}$, $\overline{\text{WEn}}$ (BEn) assertion
11	SW0	1	While SW[1,0] = B'01, 1.5 wait cycles are inserted from address and $\overline{\text{CS}}$ assertion to $\overline{\text{RD}}$ and $\overline{\text{WEn}}$ assertion.
10	WR3	1	Number of read access wait cycles
9	WR2	0	These bits specify the number of wait cycles to be inserted during read access. In this sample task, the number of wait cycles specified by these bits are inserted during read and write accesses.
8	WR1	0	
7	WR0	0	When WR[3-0] = B'1000, 10 wait cycles are inserted during read or write accesses.
6	WM	0	Specifies the external wait mask. These bits specify whether to enable or disable external wait input. When WM = 0, external wait input is enabled.
5 to 2	—	All 0	Reserved
1	HW1	0	Number of wait cycles to be inserted from $\overline{\text{RD}}$, $\overline{\text{WEn}}$ negation to address, $\overline{\text{CS}}$ negation
0	HW0	1	While HW[1,0] = B'01, 1.5 wait cycles are inserted from $\overline{\text{RD}}$ and $\overline{\text{WEn}}$ negation to address and $\overline{\text{CS}}$ negation.
PCCR			Port C Control Register
3	PC1MD2	1	PC1 modes 2 and 1
2	PC1MD1	1	When PC1MD[2,1] = B'11, the PTC1 pin is set to the CS5A function.

(3) Variables

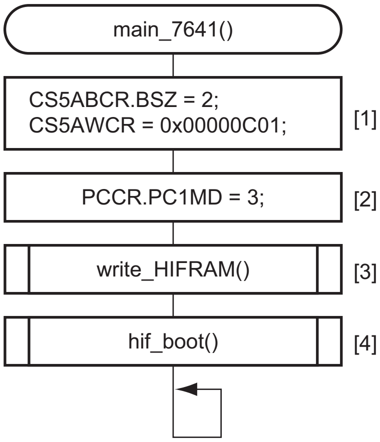
Table 8 is a list of variables used in this sample task.

Table 8 Variables

Variable	Description	Data Length	Initial Value	Used In
*trans_src_addr	Pointer that indicates the transfer source (boot program storage) address	2 bytes	—	HIFRAM consecutive write routine
hif_addr	HIF address to start writing from	2 bytes	H'0000	HIFRAM consecutive write routine
t_size	Size of the program to be transferred	2 bytes	H'300	HIFRAM consecutive write routine

2.1.4 Flowcharts

(1) SH7641 main routine



[1] CS5A space bus settings

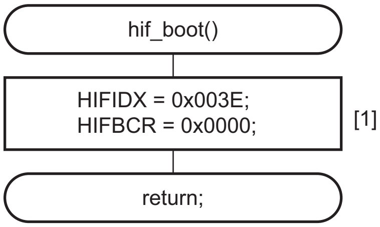
- The data bus width is set to 16 bits.
- 1.5 wait cycles are inserted from Address, $\overline{\text{CS5A}}$ assertion to $\overline{\text{RD}}$, $\overline{\text{WEn}}$ assertion.
- 1.5 wait cycles are inserted from $\overline{\text{RD}}$, $\overline{\text{WEn}}$ negation to address, $\overline{\text{CS5A}}$ negation.
- 10 wait cycles are inserted during read/write access.

[2] The pin function is set to $\overline{\text{CS5A}}$.

[3] The HIFRAM consecutive write routine is called.

[4] The SH7618 startup routine is called.

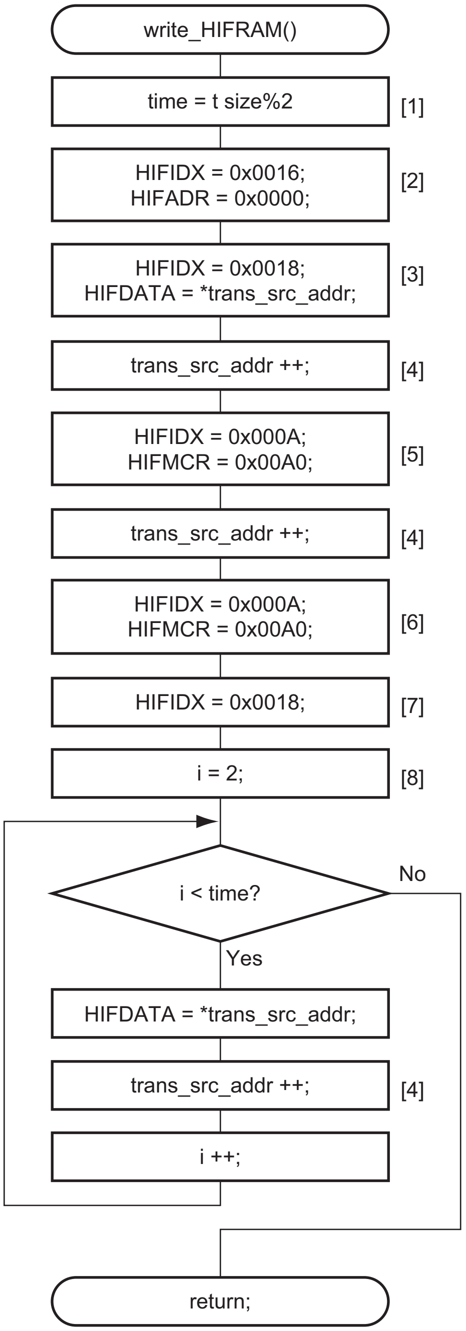
(2) SH7618 startup routine



[1] The HIFDX register is used to specify the lower 16 bits of the HIFBCR register, and the AC bit of the HIFBCR register is cleared to 0.
(The SH7618 reads the data written at the start address of HIFRAM as a reset vector, and starts.)

(3) HIFRAM consecutive write routine

Arguments:
*trans_src_addr: Pointer to the transfer-source address
hif_addr: Start address for writing to HIFRAM
t_size: Size of the program to be transferred



[1] The number of writes to HIFRAM is set.
time: Number of transfers

[2] The HIFIDX register is used to specify the lower 16 bits of the HIFADR register, and the HIFRAM start address (0x0000) is set in the HIFADR register.

[3] The HIFIDX register is used to specify the upper 16 bits of the HIFDATA register, and the write data is set in bits [31:16] of the HIFDATA register.

[4] The transfer-source address is incremented (+2).

[5] The write data is set in bits [15:0] of the HIFDATA register.

[6] The HIFIDX register is used to specify the lower 16 bits of the HIFMCR register, and HIFRAM consecutive write mode is set.
(The HIFADR register value is incremented automatically each time the HIFDATA register is accessed.)

[7] The HIFIDX register is used to specify bits [31:16] of the HIFDATA register.

[8] Data write to the HIFDATA register continues until the writing of the boot program to HIFRAM is completed.
(The first 4 bytes have already been written, so loop variable "i" starts at 2.)

2.1.5 Program Listing

```
/*****************************************************************/
// SH7618 HIF boot mode application note
//      A boot program is written to HIFRAM and SH7618 is booted
//               CPU     : SH7641,SH-3 DSP,Big Endian
//               Clock   : External input = 12.5MHz
//                         CPU clock = 100MHz
//                         External BUS clock = 50MHz
//                         Peripheral clock = 25MHz
//               Written : '04/4 Rev.2.0
/*****************************************************************/

#include "7641.h"

/**************************************************/
/*      Protocol declaration of the function      */
/**************************************************/
/*------ Symbol Definition ------------------------------*/
#define BOOT_STRAGE_ADDR   0xA5600000             // storing address of a boot program
#define HIFRAM_START       0x0000                 // write address of HIFRAM
#define BOOT_P_SIZE        0x300                  // boot program size(Byte)

        //------The value when specifying the register of HIF
#define SEL_HIFMCR_LO      0x000A                 // HIFMCR[15:0]
#define SEL_HIFBCR_LO      0x003E                 // HIFBCR[15:0]
#define SEL_HIFADR_LO      0x0016                 // HIFADR[15:0]
#define SEL_HIFDATA_UP     0x0018                 // HIFDATA[31:16]

/*------- Function Definition ---------------------------*/
void main(void);

void write_HIFRAM(unsigned short *, unsigned short , unsigned short);
void hif_boot(void);

/*--------------------------------------------------*/
        //------The address when accessing the register of HIF
// select of the register of HIF
#define HIF_REG_SEL       (*(volatile unsigned short *)0xB4001000)
// data write to the register of HIF
#define HIF_DATA_WR       (*(volatile unsigned short *)0xB4000000)

/**************************************************/
/*      Main routine                              */
/**************************************************/
void main(void)
{
      //----------set of bus interface
      BSC.CS5ABCR.BIT.BSZ = 2;
      BSC.CS5AWCR = 0x00000C01;

      //----------set as a  CS5A function
      PFC.PCCR.BIT.PC1MD = 3;          /* bit[2-3]-PC1MD=b'11 : PC1=>CS5A           */
      //----------boot program is written to HIFRAM
      write_HIFRAM((unsigned short *)BOOT_STRAGE_ADDR , HIFRAM_START , BOOT_P_SIZE);

      //----------SH7618 is booted
      hif_boot();                      /* HIF boot                                  */

      //----------Loop
      while(1);                        /* Loop                                      */
}

/************************************************************************************/
//    function  :write_HIFRAM
//    operation :boot program writing to HIFRAM
//    argument  :trans_src_addr ;storing address of a boot program
//               hif_addr       ;head address of HIFRAM which transmits a boot program
//               t_size         ;transmit program size
//    return    :non-return
/************************************************************************************/
void write_HIFRAM(unsigned short *trans_src_addr , unsigned short hif_addr , unsigned
short t_size)
{
      volatile unsigned short time , i ;

      time = t_size/2 + t_size%2;      /* calculate times of transmission           */

      HIF_REG_SEL = SEL_HIFADR_LO;     /* select HIFADR register                    */
      HIF_DATA_WR = hif_addr;          /* set of HIFRAM address                     */

      HIF_REG_SEL = SEL_HIFDATA_UP;    /* select HIFDATA register[31:16]            */
      HIF_DATA_WR = (*trans_src_addr); /* set to a HIFDATA register[31:16]          */

      trans_src_addr ++;               /* storing address is increment(+2)          */

      HIF_DATA_WR = (*trans_src_addr); /* set to a HIFDATA register[15:0]           */

      trans_src_addr ++;               /* storing address is increment(+2)          */

      HIF_REG_SEL = SEL_HIFMCR_LO;     /* select HIFMCR register[15:0]              */
      HIF_DATA_WR = 0x00A0;            /* set as continuation write mode            */

      HIF_REG_SEL = SEL_HIFDATA_UP;    /* select HIFDATA register[31:16]            */

      for( i=2 ; i<time ; i++){
            HIF_DATA_WR = (*trans_src_addr);  /* write to HIFDATA register(HIFRAM)  */

            trans_src_addr ++;                /* storing address is increment(+2)   */
      }
}

/***************************************************/
//    function      : hif_boot
//    operation     : SH7618 is booted

//    argument      : non-argument
//    return        : non-return
/**************************************************/
void hif_boot(void)
{
      HIF_REG_SEL = SEL_HIFBCR_LO;     /* select HIFBCR register[15:0]             */
      HIF_DATA_WR = 0x0000;            /* clear AC bit of HIFBCR register          */
}
```

2.2 Program Transfer to Internal RAM in HIF Boot Mode

2.2.1 Overview

This section describes a sample task that transfers a program to the internal RAM via the HIFRAM and executes the program after the SH7618 is started up in the HIF boot mode.

By storing the SH7618 boot program in the same ROM as external device programs, the ROM exclusively for activating the SH7618 can be eliminated.

2.2.2 Specifications

(1) While the program is being executed in the HIFRAM, the program is transferred using the remaining available area in the HIFRAM.

(2) The SH7641 is connected to the HIF, and the SH7618 is started up in the HIF boot mode.

(3) The program that the SH7641 transferred to the HIFRAM buffer area is transferred to the internal RAM by the SH7618, using the available area in the HIFRAM as a buffer.

(4) The program is divided into parts that are transferred separately to SH7618 internal RAM.

(5) The HIF General Status Register (HIFGSR) is used to synchronize the program transfer sessions between the SH7641 and SH7618 to prevent conflicts between transfers from the SH7641 to HIFRAM and transfers from HIFRAM to the internal RAM.

(6) The SH7641 writes the program, including a header that indicates the transfer size, to the HIFRAM buffer.

(7) The SH7618 transfers the amount of data specified in the header from the HIFRAM buffer to the internal RAM.

(8) After the program is transferred to the internal RAM, the SH7618 executes the program.

(9) In this sample task, the SH7641 operates with a 100-MHz CPU clock, a 50-MHz external bus clock, and a 25-MHz peripheral clock. Since each register of the SH7618 retains its initial value until the SH7618 is activated, the operating frequency of the SH7618 is set after startup. After startup, the SH7618 operates with a 100-MHz internal clock, a 50-MHz external bus clock, and a 50-MHz peripheral clock.

Figure 11 shows an example of connection between the SH7641 and the SH7618.

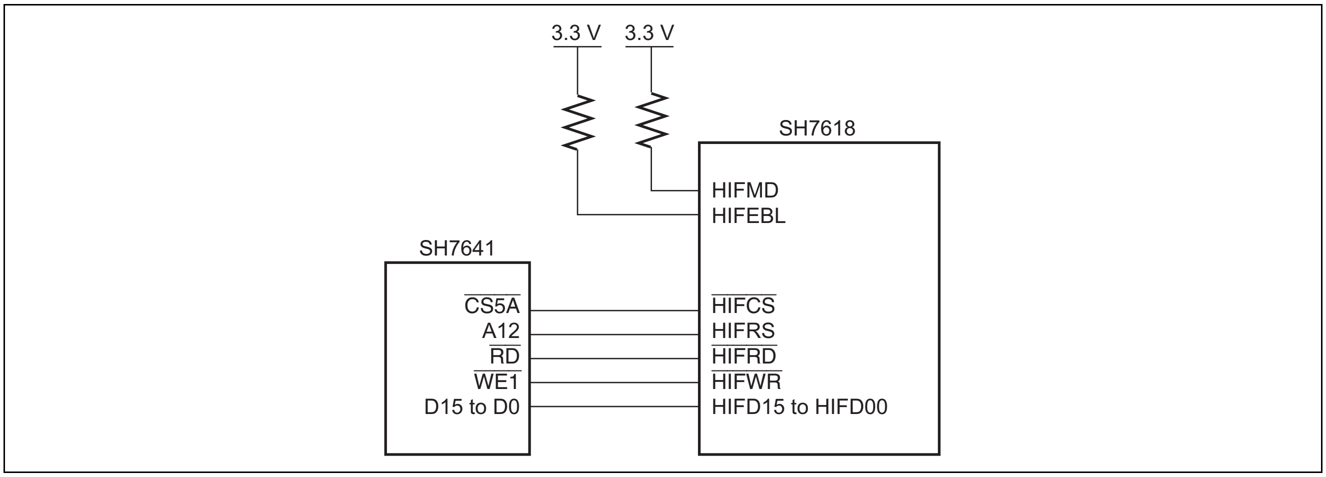


Figure 11 Connection Example

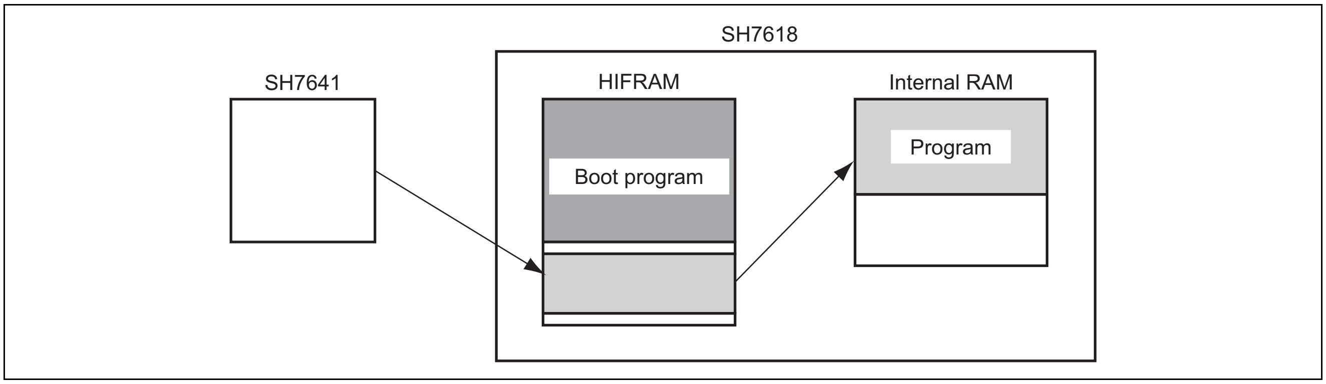
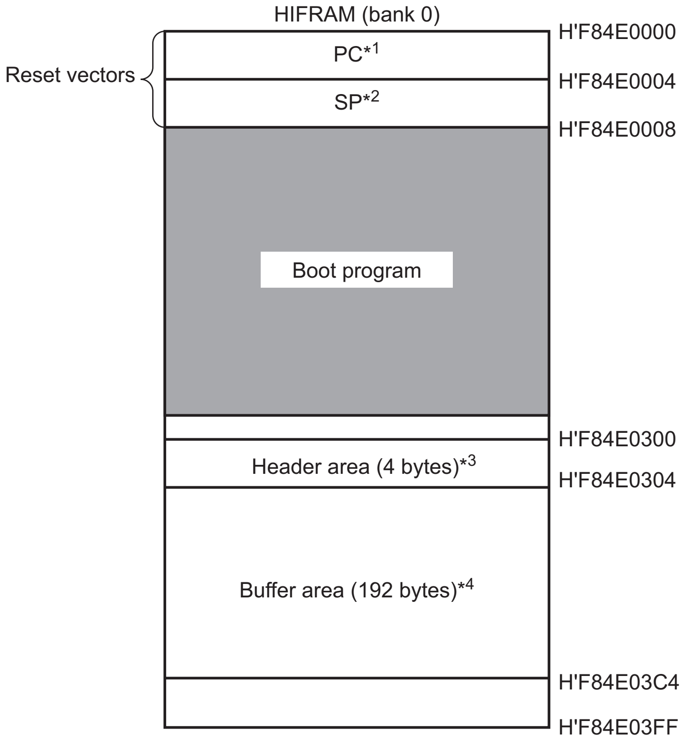


Figure 12 Program Transfer to Internal RAM

Figure 13 shows the memory map of the HIFRAM for transferring a program to the internal RAM.



Notes: *1. PC: Program counter

*2. SP: Stack pointer

*3. The header area stores the transfer size.

*4. In this sample task, the buffer area size of 192 bytes is determined based on the size of the boot program.

Figure 13 HIFRAM Memory Map

The user can define the HIFGSR register bits. In this sample task, bit 1 is defined as TEND and bit 0 is defined as HIFS to synchronize transfer sessions between the SH7641 and the SH7618. Table 9 is a list of HIFGSR register definitions in this sample task.

Table 9 HIFGSR Register Definitions

Bit	Bit Name	Setting Value	Function
31 to 16	—	All 0	Reserved
15 to 2	STATUS15 to STATUS2	All 0	General status bits These bits are not used in this sample task.
1	TEND	—	Transmit end This bit indicates whether program transfer has ended. The SH7618 references this bit to verify that program transfer has ended, and executes the program when it has been transferred to internal RAM. The SH7641 sets this bit to 1 after all program transfer sessions are complete. 1: All program transfer sessions have ended. The SH7618 executes the program transferred to internal RAM. 0: Program transfer is in progress. Part of the program being transferred to internal RAM remains on the SH7641 side.
0	HIFS	—	HIFRAM status This bit indicates the status of the HIFRAM. The SH7618 and the SH7641 reference this bit to perform a transfer. The SH7618 can set this bit to 1 only when a transfer from the SH7641 to HIFRAM is possible. The SH7641 can set this bit to 0 only when a program transfer to the HIFRAM buffer has ended. 1: The SH7641 can perform a write to HIFRAM. The SH7618 is in the standby state. 0: The SH7641 cannot perform a write to HIFRAM. The SH7618 transfers a program from the HIFRAM buffer to internal RAM.

2.2.3 Operation

Figure 14 shows the HIFGSR values and indicates the SH7618 and SH7641 operations for those HIFGSR values. Table 10 describes the operations in detail. Figure 15 shows how operation that differs depending on the value of the HIFS bit of the HIFGSR register.

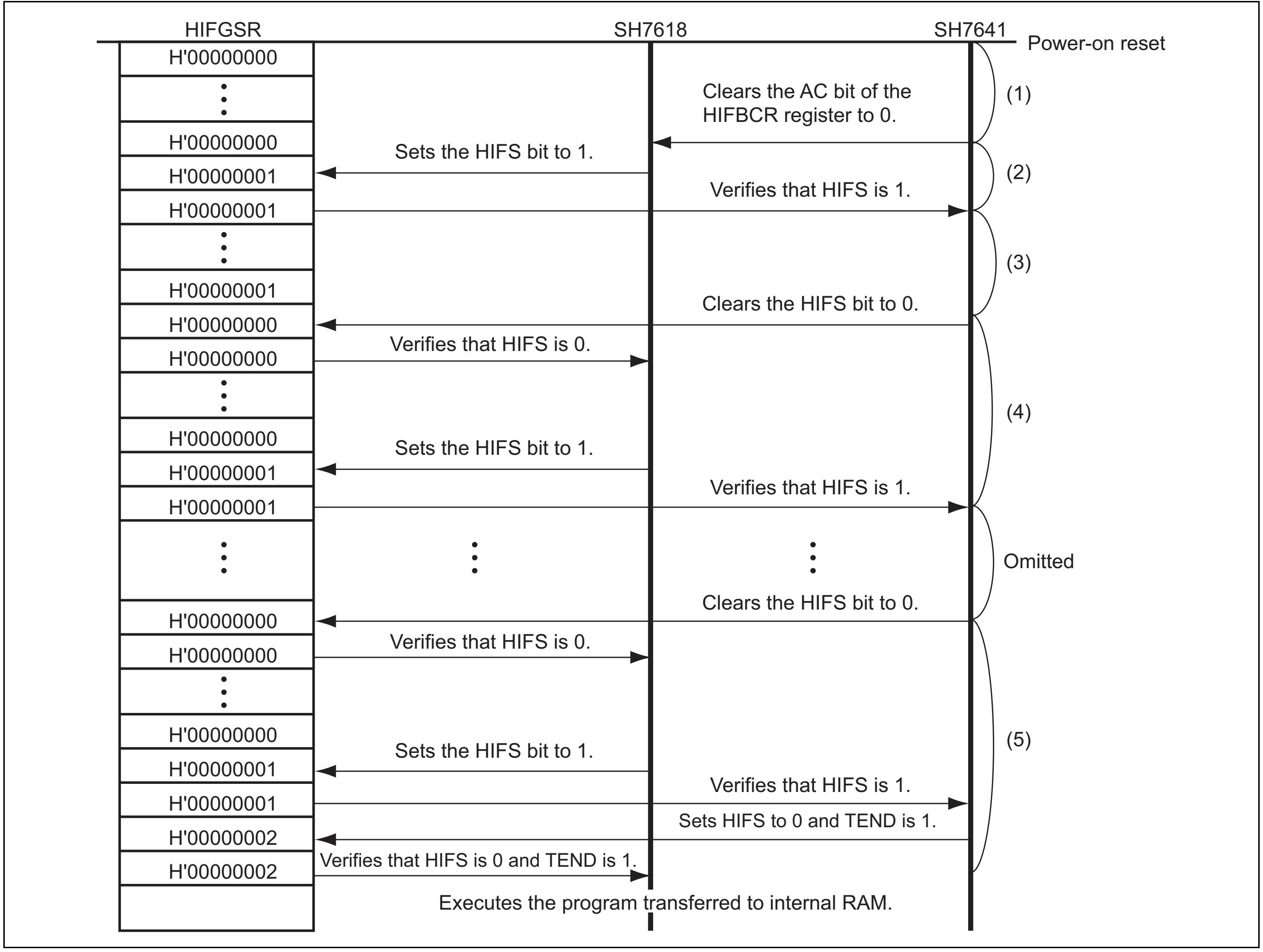
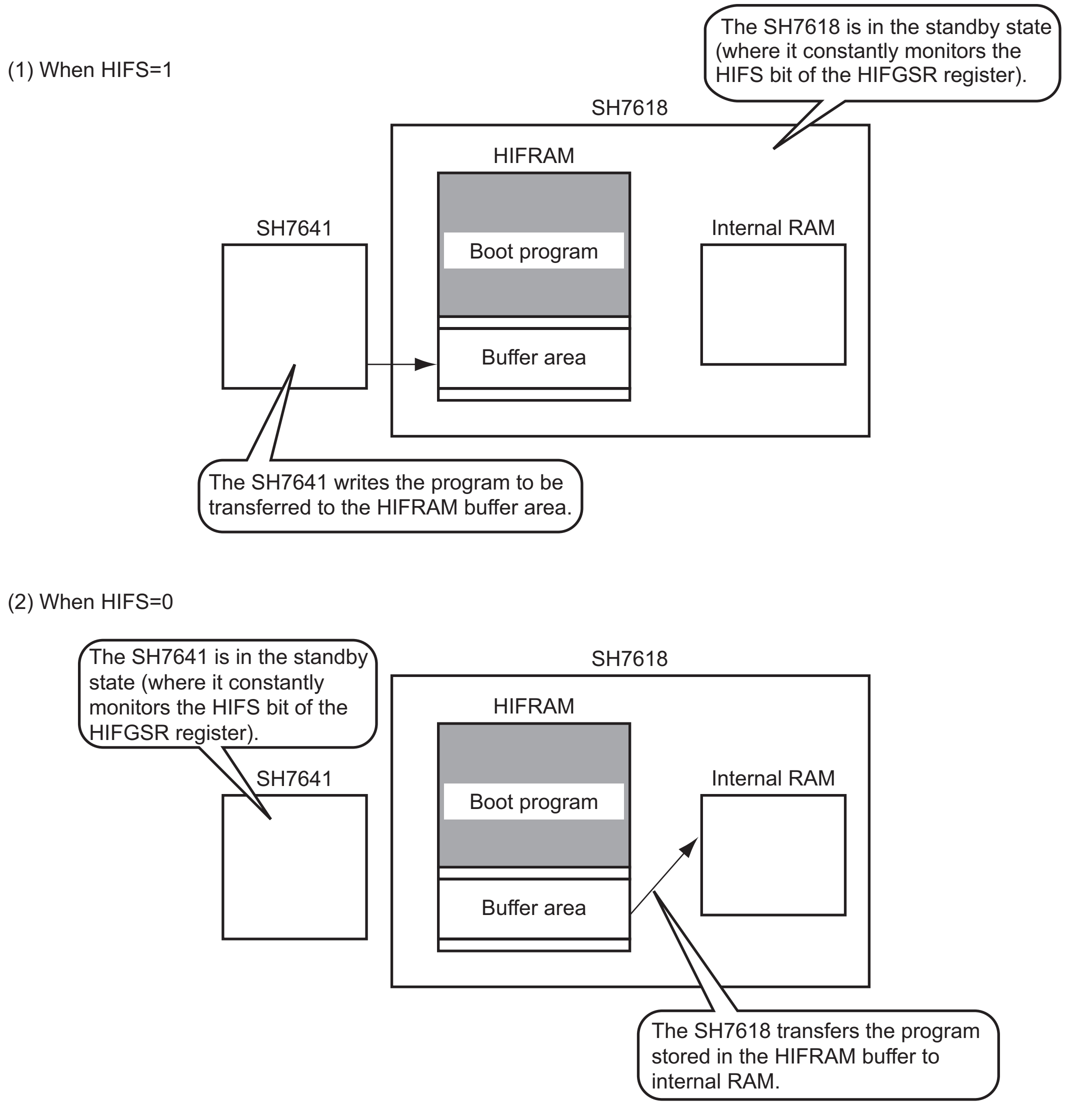


Figure 14 HIFGSR Values and Operations

Table 10 Description of SH7618 and SH7641 Operations

	SH7618 Operation	SH7641 Operation
(1)	• A power-on reset is canceled while a high level is input to the HIFMD. The SH7618 is placed in the HIF boot mode.* • The SH7618 waits until the AC bit of the HIFBCR register is cleared to 0.*	• After the reset state is released, the SH7641 writes the boot program to HIFRAM. • After the boot program is written, the SH7641 clears the AC bit of the HIFBCR register to 0.
(2)	• The AC bit of the HIFBCR register is cleared to 0, and the program written to HIFRAM is executed.* • Initial setting of the SH7618 is performed and HIFRAM is initialized. • The SH7618 sets the HIFS bit of the HIFGSR register to 1.	• The SH7641 waits while the HIFS bit of the HIFGSR register is 0.
(3)	• The SH7618 waits while the HIFS bit of the HIFGSR register is 1.	• The SH7641 verifies that the HIFS bit of the HIFGSR register has been set to 1. • The SH7641 writes to the HIFRAM buffer the program to be transferred. • After the program is written, the SH7641 clears the HIFS bit of the HIFGSR register to 0.
(4)	• The SH7618 verifies that the HIFS bit of the HIFGSR register is cleared to 0. • The SH7618 transfers the program that was written by the SH7641 from the HIFRAM buffer to internal RAM. • After the program is transferred, the SH7618 sets the HIFS bit of the HIFGSR register to 1.	• The SH7641 waits while the HIFS bit of the HIFGSR register is 0.
(5)	• The SH7618 verifies that the HIFS bit of the HIFGSR register is cleared to 0. • The SH7618 transfers the program that was written by the SH7641 from the HIFRAM buffer to internal RAM. • After the program is transferred, the SH7618 sets the HIFS bit of the HIFGSR register to 1. • The SH7618 verifies that the HIFS bit of the HIFGSR register is 0 and that the TEND bit of the HIFGSR register is 1. It then starts processing of the program transferred to internal RAM.	• When all program write sessions have been completed, the SH7641 clears the HIFS bit of the HIFGSR register to 0. • The SH7641 waits while the HIFS bit of the HIFGSR register is 0. • The SH7641 verifies that the HIFS bit of the HIFGSR register has been set to 1. • The SH7641 writes HIFS = 0 and TEND = 1 in the HIFGSR register.

Note: * This processing is performed by hardware; other processing is performed by software.


(1) When HIFS=1
The SH7618 is in the standby state (where it constantly monitors the HIFS bit of the HIFGSR register).
SH7618
HIFRAM
SH7641
Boot program
Internal RAM
Buffer area
The SH7641 writes the program to be transferred to the HIFRAM buffer area.
(2) When HIFS=0
The SH7641 is in the standby state (where it constantly monitors the HIFS bit of the HIFGSR register).
SH7618
HIFRAM
SH7641
Boot program
Internal RAM
Buffer area
The SH7618 transfers the program stored in the HIFRAM buffer to internal RAM.

Note: Bank 0 of HIFRAM is used to execute the boot program and to transfer the program to internal RAM. Bank 1 is not used.

Figure 15 HIFS Bit Settings and Operations

2.2.4 Description of Software

(1) Modules

Table 11 is a list of SH7618 modules used in this sample task. Table 12 is a list of SH7641 modules used in this sample task.

Table 11 SH7618 Modules

Module Name	Label Name	Description
HIFRAM main routine	hifram_main	Performs initial setting of the SH7618 and transfers the program stored in the HIFRAM buffer to internal RAM.
Transfer program execution routine	jump_uram	Moves the program counter to internal RAM, and executes the transferred program. Note: This module is written in assembly language.

Table 12 SH7641 Modules

Module Name	Label Name	Description
SH7641 main routine	main_7641	Sets the external bus and pins, and calls the individual modules.
HIFRAM consecutive write routine	write_HIFRAM	Writes a boot program to HIFRAM.
SH7618 startup routine	hif_boot	Starts up the SH7618 in the HIF boot mode.
HIFRAM buffer write routine	sync_7618	Writes the boot program to the HIFRAM buffer in synchronization with the SH7618.

(2) Internal registers used

Table 13 lists the SH7618 internal registers used in this sample task.

Table 13 SH7618 Internal Registers Used

Register Name			Setting	
	Bit	Bit Name	Value	Function
HIFADR				HIF Address Register
	31 to 10	—	All 0	Reserved
	9 to 2	A9 to A2	—	Specify the target HIFRAM address. These bits specify, on a 32-bit boundary, the address in HIFRAM to be accessed by the external device.
	1	—	0	Reserved
	0	—	0	
HIFDATA				HIF Data Register
	31 to 0	D31 to D0	—	32-bit data. These bits are used for an access to HIFRAM from the external device.
HIFBCR				HIF Boot Control Register
	31 to 8	—	All 0	Reserved
	7 to 1	—	All 0	AC bit writing assistance These bits are used to write the bit pattern (H'A5) needed to set the AC bit to 1.
	0	AC	—	HIFRAM exclusive access control When the SH7618 is started up in the HIF boot mode, this bit is set to 1. After a program is written to the HIFRAM, the SH7641 clears this bit to 0, and the SH7618 is activated.

Register Name		Setting	
Bit	Bit Name	Value	Function
HIFIDX			HIF Index Register
31 to 8	—	All 0	Reserved
7 to 2	REG5 to REG0	—	Select the HIF registers. These bits are used to select the HIF registers that will be accessed by the external device.
1	BYTE1	—	Select a byte in an HIF register.
0	BYTE0	—	These bits are used to specify the word position when an external device accesses an HIF internal register.
HIFMCR			HIF Memory Control Register
31 to 8	—	All 0	Reserved
7	LOCK	—	Lock bit This bit is used when an external device performs consecutive access to HIFRAM.
6	—	0	Reserved
5	WT	—	Write bit When this bit is set to 1, the HIFDATA value is written to the HIFRAM location corresponding to HIFADR.
4	—	0	Reserved
3	RD	—	Read bit When this bit is set to 1, the HIFRAM data corresponding to HIFADR is fetched into HIFDATA.
2, 1	—	All 0	Reserved
0	AI/AD	0	Address auto-increment/decrement When LOCK=1 and AI/AD=0, each time the SH7641 accesses HIFDATA, the HIFADR value is incremented (+4) so that data can be written to, or read from, consecutive HIFRAM addresses.
HIFGSR			HIF General Status Register
31 to 16	—	All 0	Reserved
15 to 2	STATUS15 to STATUS2	All 0	General status bits These bits are not used in this sample task.
1	TEND	—	Transmit end This bit indicates the program transfer status. TEND=1: The SH7618 executes the program transferred to internal RAM. TEND=0: The SH7618 transfers the program to internal RAM.
0	HIFS	—	HIFRAM status This bit indicates the status of HIFRAM. HIFS=0: The SH7641 can perform a write to HIFRAM. HIFS=1: The SH7641 cannot transfer a program to HIFRAM.

Table 14 is a list of SH7641 internal registers used in this sample task.

Table 14 SH7641 Internal Registers Used

Register Name		Setting	
Bit	Bit Name	Value	Function
CS5ABCR			CS5A Space Bus Control Register
10	BSZ1	1	Specify the data bus width for accessing the CS5A space.
9	BSZ0	0	When BSZ[1,0] = B'10, the data bus width is set to 16 bits.
CS5AWCR			CS5A Space Wait Control Register
31 to 19	—	All 0	Reserved
18	WW2	0	Number of write access wait cycles
17	WW1	0	These bits specify the number of wait cycles to be inserted during write access.
16	WW0	0	When WW[2-0] = B'000, the same number of wait cycles as the number of read access wait cycles specified by the WR bits are inserted.
15 to 13	—	All 0	Reserved
12	SW1	0	Number of delay cycles from address, $\overline{\text{CS}}$ assertion to the $\overline{\text{RD}}$, $\overline{\text{WEn}}$ (BEn) assertion
11	SW0	1	While SW[1,0] = B'01, 1.5 wait cycles are inserted from address and $\overline{\text{CS}}$ assertion to $\overline{\text{RD}}$ and $\overline{\text{WEn}}$ assertion.
10	WR3	—	Number of read access wait cycles
9	WR2	—	These bits specify the number of wait cycles to be inserted during read access.
8	WR1	—	In this sample task, the number of wait cycles to be inserted changes after the SH7618 is started up in the HIF boot mode.
7	WR0	—	Before startup: WR[3-0] = B'1000 (10 wait cycles are inserted) After startup: WR[3-0] = B'0010 (2 wait cycles are inserted)
6	WM	0	Specifies the external wait mask. These bits specify whether to enable or disable external wait input. When WM=0, external wait input is enabled.
5 to 2	—	All 0	Reserved
1	HW1	0	Number of wait cycles to be inserted from $\overline{\text{RD}}$, $\overline{\text{WEn}}$ negation to address, $\overline{\text{CS}}$ negation
0	HW0	1	While HW[1,0] = B'01, 1.5 wait cycles are inserted from $\overline{\text{RD}}$ and $\overline{\text{WEn}}$ negation to address and $\overline{\text{CS}}$ negation.
PCCR			Port C Control Register
3	PC1MD2	1	PC1 modes 2 and 1
2	PC1MD1	1	When PC1MD[2,1] = B'11, the PTC1 pin is set to the CS5A function.

(3) Variables

Table 15 is a list of variables used in the SH7618 program in this sample task. Table 16 is a list of variables used in the SH7641 program in this sample task.

Table 15 Variables Used in the SH7618 Program

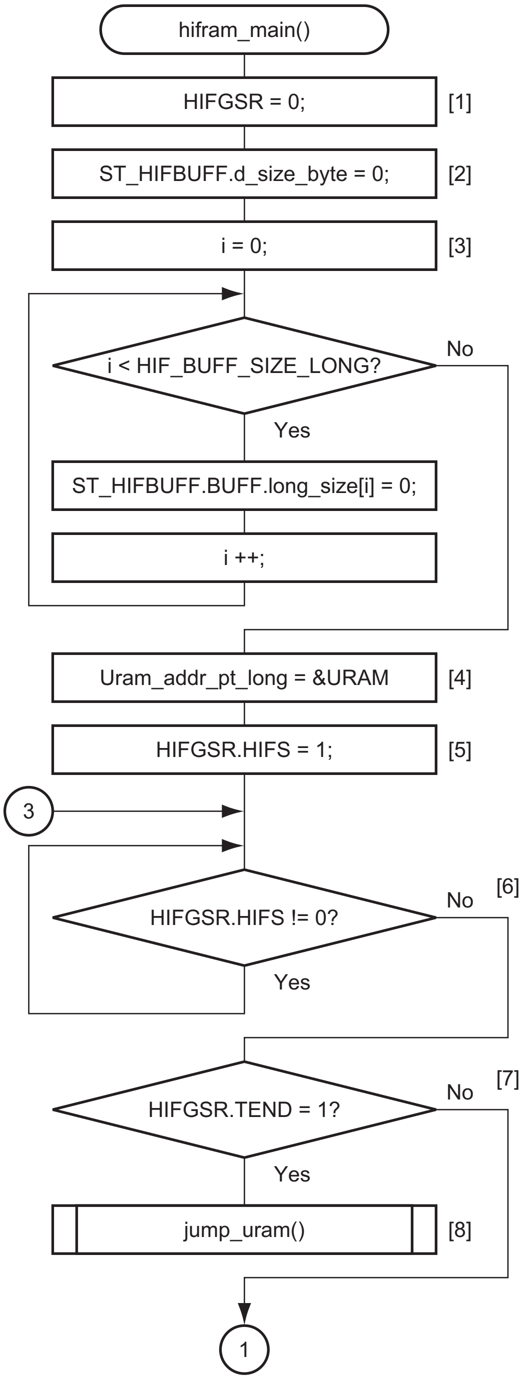
Variable	Description	Data Length	Initial Value	Used In
t_count	Number of transfer sessions for a transfer in longword units	4 bytes	—	HIFRAM main routine
*Uram_address_pt_byte	Pointer that indicates the program transfer destination (internal RAM) address	1 byte	—	HIFRAM main routine
*Uram_address_pt_long	Pointer that indicates the program transfer destination (internal RAM) address	4 bytes	—	HIFRAM main routine

Table 16 Variables Used in the SH7641 Program

Variable	Description	Data Length	Initial Value	Used In
*trans_src_addr	Pointer that indicates the transfer source (boot program storage) address	2 bytes	—	HIFRAM consecutive write routine
hif_addr	Start address where writing starts	2 bytes	H'0000	HIFRAM consecutive write routine
t_size	Size of the program to be transferred	2 bytes	H'300	HIFRAM consecutive write routine
*s_addr	Pointer that indicates the transfer source (boot program storage) address	2 bytes	—	HIFRAM buffer write routine
h_addr	HIFRAM header address	2 bytes	H'0300	HIFRAM buffer write routine
b_addr	HIFRAM buffer address	2 bytes	H'0304	HIFRAM buffer write routine
t_size	Total size of the program to be transferred	4 bytes	H'0300	HIFRAM buffer write routine
b_size	HIFRAM buffer size (size of data transferred in one transfer session)	2 bytes	H'C0	HIFRAM buffer write routine

2.2.5 Flowcharts

(1) HIFRAM main routine



[1] The HIFGSR register is initialized.

[2] The HIFRAM header is initialized.

[3] The HIFRAM buffer is initialized.

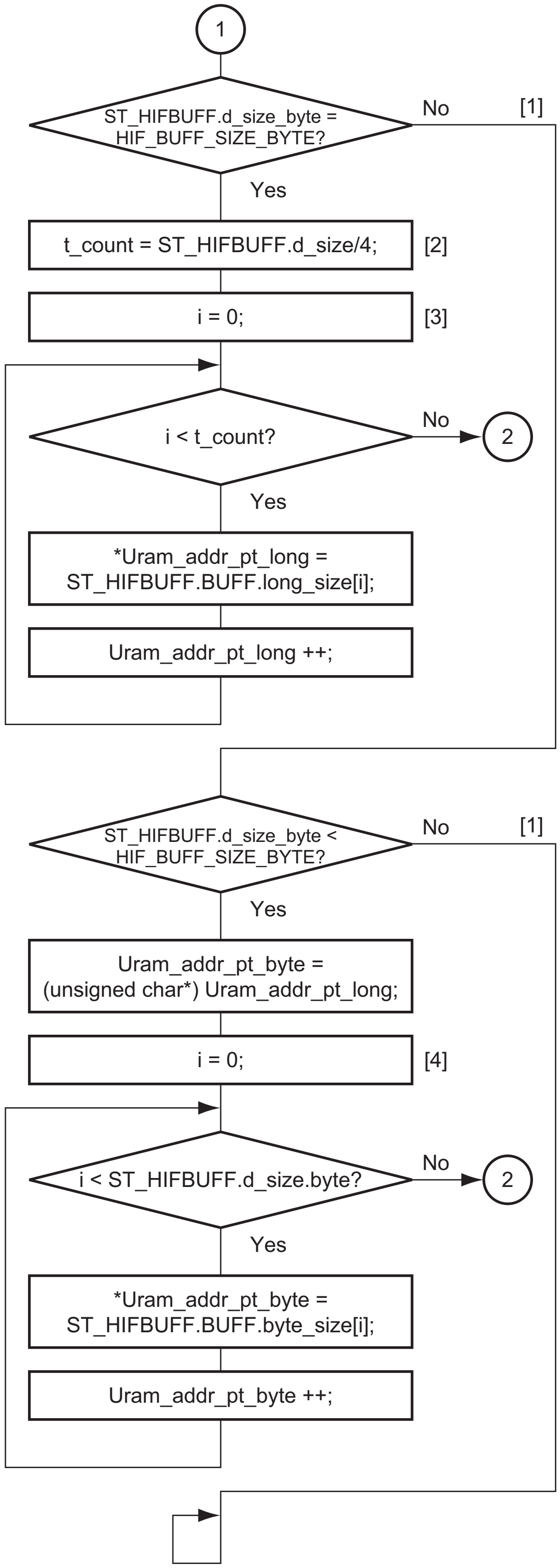
[4] The program transfer destination address is set.

[5] The HIFS bit of the HIFGSR register is set to 1 to enable the external device to write to HIFRAM.

[6] The program waits until the write from the external device to HIFRAM is completed.

[7] Whether the entire program has been transferred is decided.
If TEND=1, the transfer has been completed.

[8] The routine that executes the transferred program is called.



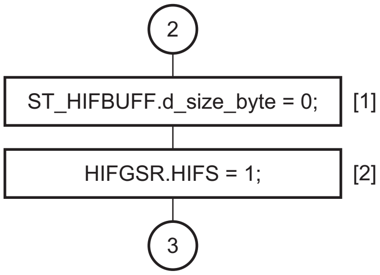
[1] Whether data should be transferred in longword or byte units is decided.

- If transfer-size = buffer-size, then data is transferred in longword units.
- If transfer-size < buffer-size, then data is transferred in byte units.
- In other cases, an error (endless loop) occurs.

[2] The transfer count for a transfer in longword units is set.

[3] Data is transferred in longword units from the HIFRAM buffer to internal RAM.

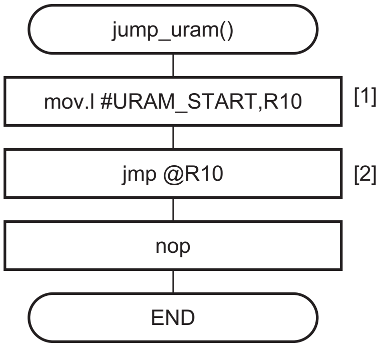
[4] Data is transferred in byte units of from the HIFRAM buffer to internal RAM.



[1] The HIFRAM header is initialized.

[2] The HIFS bit of the HIFGSR register is set to 1 to enable the external device to write to HIFRAM.

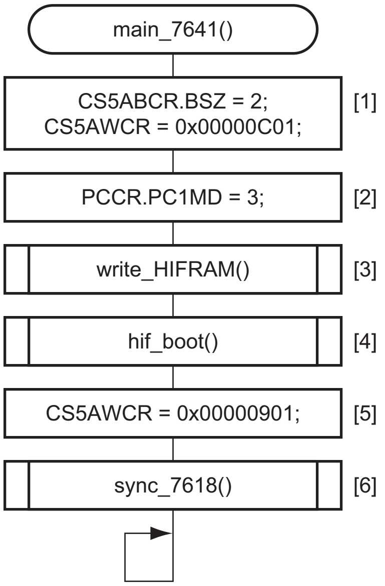
(2) Transfer program execution routine



[1] The execution start address of the transferred program is set.
URAM_START: Start address of internal RAM

[2] Control branches to the set address.

(3) SH7641 main routine



[1] CS5A space bus settings
The data bus width is set to 16 bits.

- 1.5 wait cycles are inserted from address and CS5A assertion to RD and WEn assertion.
- 1.5 wait cycles are inserted from RD and WEn negation to address and CS5A negation.
- 10 wait cycles are inserted during read/write access.

[2] The pin function is set to CS5A.

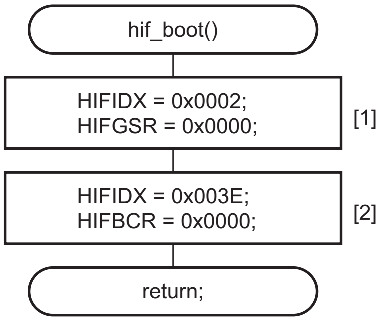
[3] The HIFRAM consecutive write routine is called.

[4] The SH7618 startup routine is called.

[5] The number of wait cycles to be inserted during read/write access for the CS5A area is changed to 2. (The change is made because faster access is possible after the SH7618 is started and the operating frequency is increased.)

[6] The HIFRAM buffer write routine is called.

(4) SH7618 startup routine



[1] The HIFDX register is used to specify the lower 16 bits of the HIFGSR register, and the HIFGSR register is initialized. (H'0000 is written.)

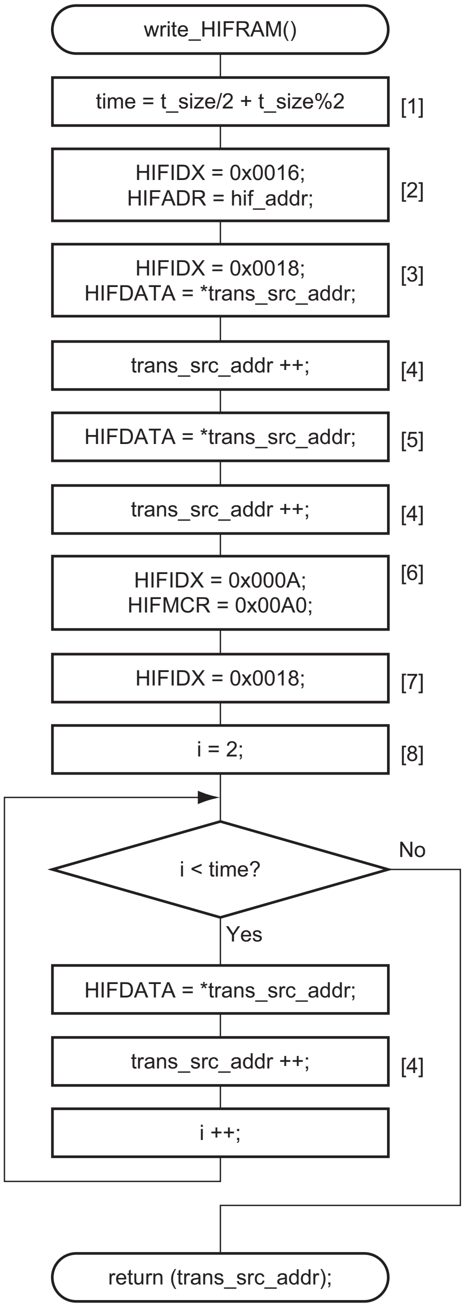
[2] The HIFDX register is used to specify the lower 16 bits of the HIFBCR register, and the AC bit of the HIFBCR register is cleared to 0.
(The SH7618 reads the data written at the start address of HIFRAM as a reset vector, and starts up.)

(5) HIFRAM consecutive write routine

Arguments:

*trans_src_addr:	Pointer to the transfer-source address
hif_addr:	HIFRAM address to start writing from
t_size:	Size of the program to be transferred

Return value:
trans_src_addr: Transfer-source address



[1] The number of writes to HIFRAM is set.
time: Number of transfers

[2] The HIFIDX register is used to specify the lower 16 bits of the HIFADR register, and the HIFRAM address to start writing from is set in the HIFADR register.

[3] The HIFIDX register is used to specify the upper 16 bits of the HIFDATA register, and the write data is set in bits [31:16] of the HIFDATA register.

[4] The transfer-source address is incremented (+2).

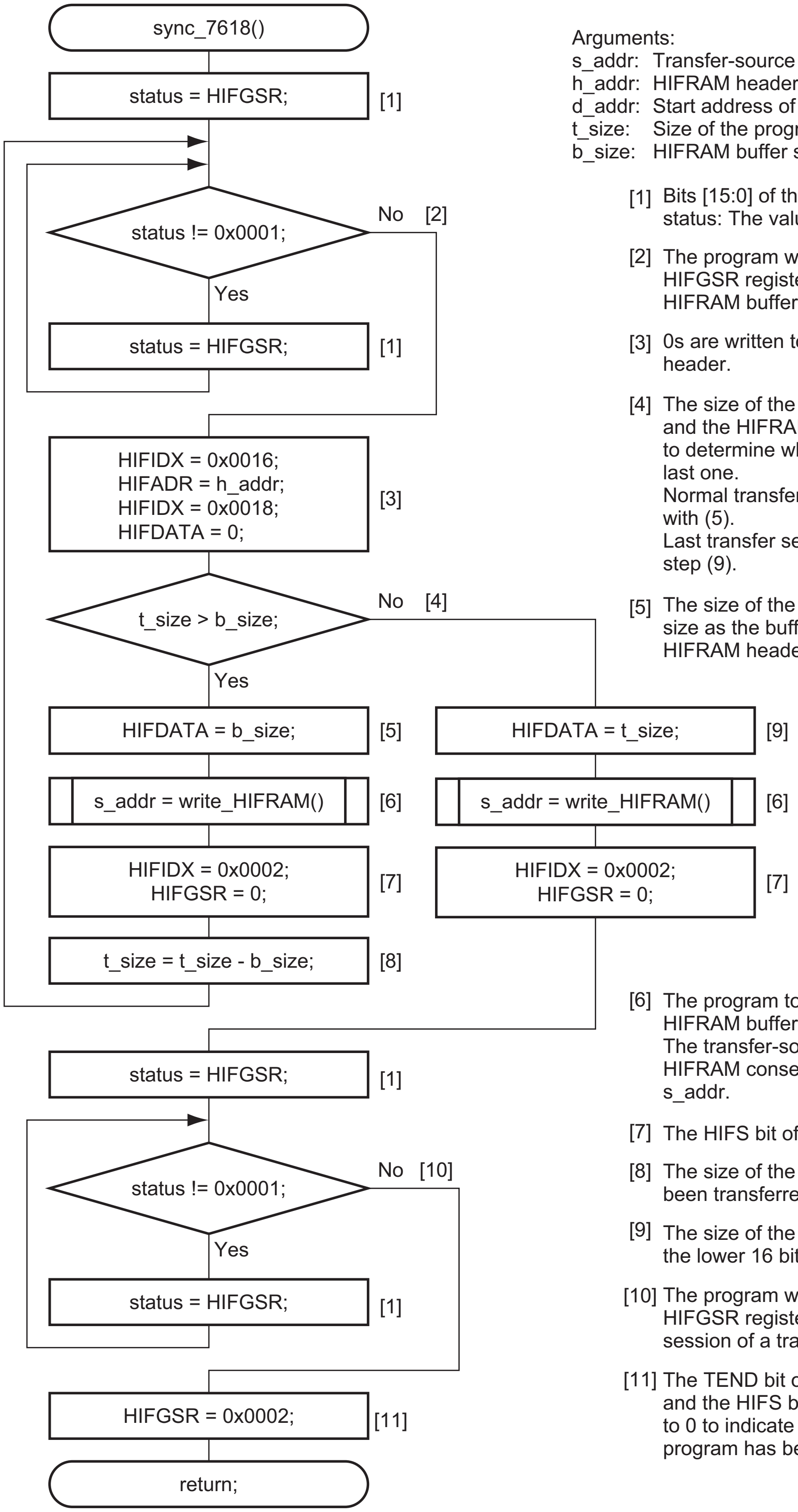
[5] The write data is set in bits [15:0] of the HIFDATA register.

[6] The HIFIDX register is used to specify the lower 16 bits of the HIFMCR register, and HIFRAM consecutive write mode is set.
(The HIFADR register value is incremented automatically each time the HIFDATA register is accessed.)

[7] The HIFIDX register is used to specify bits [31:16] of the HIFDATA register.

[8] Writing of data to the HIFDATA register continues until the writing of the boot program to HIFRAM is completed.
(The first 4 bytes have already been written, so loop variable "i" starts at 2.)

(6) HIFRAM buffer write routine



Arguments:

s_addr:	Transfer-source address
h_addr:	HIFRAM header address
d_addr:	Start address of the HIFRAM buffer
t_size:	Size of the program to be transferred
b_size:	HIFRAM buffer size

[1] Bits [15:0] of the HIFGSR register are read.
status: The value of HIFGSR is stored.

[2] The program waits until the HIFS bit of the HIFGSR register is set to 1 (write access to the HIFRAM buffer is possible).

[3] 0s are written to the upper 16 bits of the HIFRAM header.

[4] The size of the program to be transferred (t_size) and the HIFRAM buffer size (b_size) are compared to determine whether the transfer session was the last one.
Normal transfer sessions: The program continues with (5).
Last transfer sessions: The program branches to step (9).

[5] The size of the program to be transferred (same size as the buffer) is set in the lower 16 bits of the HIFRAM header.

[6] The program to be transferred is written to the HIFRAM buffer.
The transfer-source address used within the HIFRAM consecutive write routine is stored in s_addr.

[7] The HIFS bit of the HIFGSR register is cleared to 0.

[8] The size of the part of the program that has not been transferred yet is calculated.

[9] The size of the program to be transferred is set in the lower 16 bits of the HIFRAM header.

[10] The program waits until the HIFS bit of the HIFGSR register is set to 1 (indicates that the last session of a transfer from HIFRAM has ended).

[11] The TEND bit of the HIFGSR register is set to 1 and the HIFS bit of the HIFGSR register is cleared to 0 to indicate that the transfer of the entire program has been completed.

2.2.6 Program Listing

(1) SH7618 program 1

```
/********************************************************************/
// SH7618 HIF boot mode application note
//      A program is transmitted to U memory from HIFRAM buffer
//              CPU     : SH7618,SH-2,Big Endian
//              Clock   : External input = 25MHz
//                        CPU clock = 100MHz
//                        External BUS clock = 50MHz
//                        Peripheral clock = 50MHz
//              Written : '04/4 Rev.2.0
/********************************************************************/
//------ Symbol Definition ------------------------------
#define HIF_BUFF_SIZE_BYTE   0xC0   /* HIFRAM buffer size(Byte)                     */
#define HIF_BUFF_SIZE_LONG   (HIF_BUFF_SIZE_BYTE/4)
                                    /* HIFRAM buffer size(long word access)         */

struct st_hifbuff{                  /* definition of HIFRAM buffer                  */
     unsigned long d_size_byte;     /* HIFRAM header(transmission data size)        */
     union{
          unsigned long long_size[HIF_BUFF_SIZE_LONG];
                                    /* HIFRAM buffer(long word access)              */
          unsigned char byte_size[HIF_BUFF_SIZE_BYTE];
                                    /* HIFRAM buffer(byte access)                   */
     } BUFF;
};

//------ Definition of HIF Register ------------------------------
union st_hifgsr{                  /* definition of HIFGSR                           */
     unsigned long LONG;          // Long Access
     struct {                     // Bit Access
          unsigned long :24;      // reserve
          unsigned long :6;       // no use
          unsigned long TEND:1;   // TEND
          unsigned long HIFS:1;   // HIFS
     } BIT;
};

//------ Function Definition ------------------------------
void hifram_main(void);

extern void jump_uram(void);

//---------------------------------------------------------
#pragma section _HIF_BUFF
volatile struct st_hifbuff ST_HIFBUFF;
#pragma section

#define ST_HIFGSR (*(volatile union st_hifgsr*)0xF84D0004)
                                    /* SH7618 HIFGSR register address               */

#define URAM (*(volatile unsigned long *)0xE55FF000)
                                    /* U memory top address, Non Cache area         */
```

```
unsigned char* Uram_addr_pt_byte;   /* pointer of U memory address(byte access)     */
unsigned long* Uram_addr_pt_long;   /* pointer of U memory address(long word access)*/

/**********************************************************/
/*          Main routine                                  */
/**********************************************************/
void hifram_main( void )
{
     unsigned long i;
     unsigned long t_count;         /* times of transmission when long word access  */

     //----------initialize HIFGSR
     ST_HIFGSR.LONG = 0;

     //----------clear in HIFRAM buffer
     ST_HIFBUFF.d_size_byte = 0;
     for(  i=0 ;  i<HIF_BUFF_SIZE_LONG ; i++ )
     {
          ST_HIFBUFF.BUFF.long_size[i] = 0;
     }

     Uram_addr_pt_long = &URAM;     /* transmission destination address             */
     ST_HIFGSR.BIT.HIFS = 1;        /* set HIFS=1(The writing to HIFRAM is possible)*/

     while(1)
     {
          while(ST_HIFGSR.BIT.HIFS != 0);
                             /* wait until write end to HIFRAM from external device */

          if( ST_HIFGSR.BIT.TEND == 1 )             /* transmission end (TEND=1)    */
          {
               break;                               /* escape from loop             */
          }
          else
          {
               //----------transmit to U memory from HIFRAM buffer
               //----------transmit by long-word size
               if( ST_HIFBUFF.d_size_byte == HIF_BUFF_SIZE_BYTE )
               {
                    t_count = ST_HIFBUFF.d_size_byte/4;    /* times of transmission */

                    for( i=0 ; i<t_count ; i++ )
                    {
                         *Uram_addr_pt_long = ST_HIFBUFF.BUFF.long_size[i];
                         Uram_addr_pt_long++;
                    }
               }
               //----------transmit by byte size
               else if( ST_HIFBUFF.d_size_byte < HIF_BUFF_SIZE_BYTE )
               {
                    Uram_addr_pt_byte = (unsigned char*)Uram_addr_pt_long;

                    for( i=0 ; i<ST_HIFBUFF.d_size_byte ; i++ )
                    {
                          (*Uram_addr_pt_byte) = ST_HIFBUFF.BUFF.byte_size[i];
                         Uram_addr_pt_byte++;
                    }
               }
               else
               {
                    while(1);      /* error                                         */
               }

               ST_HIFBUFF.d_size_byte = 0;
                                   /* clear the HIFRAM header                       */
               ST_HIFGSR.BIT.HIFS = 1;
                                   /* set HIFS=1(The writing to HIFRAM is possible) */
          }
     }
     //----------Execution of the transmitted program
     jump_uram();
}
```

(2) SH7618 program 2

```
;********************************************************************
;     function      : jump_uram
;     operation     : Execution of the transmitted program
;     CUP           : SH7618
;     date          : 2004.4
;********************************************************************
               .EXPORT     _jump_uram
;
; MS7618 U memory address,user program start address
URAM_START:    .equ        H'E55FF000
               .SECTION    HIF_P, CODE, ALIGN=4
;***************************************************************
;
_jump_uram:
;
     mov.l   #URAM_START,R10                              ;Program Start
     jmp     @R10
     nop
;
     .END
;
```

(3) SH7641 program

```
/*************************************************************************************/
// SH7618 HIF boot mode application note
//     SH7618 is booted in HIF boot mode, and a program is written to a HIFRAM buffer
//          CPU     : SH7641,SH-3 DSP,Big Endian
//          Clock   : External input = 12.5MHz
//                    CPU clock = 100MHz
//                    External BUS clock = 50MHz
//                    Peripheral clock = 25MHz
//          Written : '04/4 Rev.2.0
/*************************************************************************************/

#include "7641.h"

/**********************************************************/
/*          Protocol declaration of the function          */
/**********************************************************/
/*------ Symbol Definition -------------------------------------*/
#define BOOT_STRAGE_ADDR   0xA5600000     // storing address of a boot program
#define HIFRAM_START       0x0000         // write address of HIFRAM
#define BOOT_P_SIZE        0x300          // boot program size(Byte)

#define URAM_STRAGE_ADDR   0xA5601000     // storing address of a uram program
#define URAM_P_SIZE        0x300          // uram program size(Byte)

#define HIF_BUFF_SIZE      0xC0           // HIFRAM buffer size(192Byte)
#define HIF_HEAD_ADDR      0x0300         // HIFRAM header address
#define HIF_BUFF_ADDR      0x0304         // HIFRAM buffer address

     //------The value when specifying the register of HIF
#define SEL_HIFMCR_LO      0x000A         // HIFMCR[15:0]
#define SEL_HIFBCR_LO      0x003E         // HIFBCR[15:0]
#define SEL_HIFADR_LO      0x0016         // HIFADR[15:0]
#define SEL_HIFDATA_UP     0x0018         // HIFDATA[31:16]
#define SEL_HIFGSR_LO      0x0002         // HIFGSR[15:0]

/*------ Function Definition -----------------------------------*/
void main_7641(void);

unsigned short* write_HIFRAM(unsigned short* , unsigned short , unsigned short);
void hif_boot(void);

void sync_7618(unsigned short* , unsigned short , unsigned short , unsigned long ,
unsigned short );

/*--------------------------------------------------------------*/
     //------The address when accessing the register of HIF
// select of the register of HIF
#define HIF_REG_SEL                (*(volatile unsigned short *)0xB4001000)
// data write to the register of HIF
#define HIF_DATA_WR                (*(volatile unsigned short *)0xB4000000)
// data read from the HIFGSR register
#define HIF_GSR_RD                 (*(volatile unsigned short *)0xB4001004)
```

```
/**********************************************************/
/*          Main routine                                  */
/**********************************************************/
void main_7641(void)
{
     //------set of bus interface
     BSC.CS5ABCR.BIT.BSZ = 2;
     BSC.CS5AWCR = 0x00000C01;

     //------set as a  CS5A function
     PFC.PCCR.BIT.PC1MD = 3;             /* bit[2-3]-PC1MD=b'11 : PC1=>CS5A         */

     //------boot program is written to HIFRAM
     write_HIFRAM((unsigned short *)BOOT_STRAGE_ADDR,HIFRAM_START,BOOT_P_SIZE);

     //------SH7618 is booted
     hif_boot();                         /* HIF boot                                */

     //------change of bus wait cycle
     BSC.CS5AWCR = 0x00000901;

     //------synchronization is taken SH7618, and a program is written to HIFRAM
     sync_7618((unsigned short*)URAM_STRAGE_ADDR , HIF_HEAD_ADDR ,
                                         HIF_BUFF_ADDR , URAM_P_SIZE , HIF_BUFF_SIZE);

     while(1);                           /* Loop                                    */
}

/*************************************************************************************/
//     function  : write_HIFRAM
//     operation     : boot program writing to HIFRAM
//     argument      : trans_src_addr ; storing address of a boot program
//                     hif_addr       ; head address of HIFRAM which transmits a boot
                                        program
//                     t_size         ; transmit program size
//     return        : trans_src_addr ; storing address of a boot program
/*************************************************************************************/
unsigned short* write_HIFRAM(unsigned short *trans_src_addr ,
                                      unsigned short hif_addr , unsigned short t_size)
{
     volatile unsigned short time , i ;

     time = t_size/2 + t_size%2;        /* calculate times of transmission          */

     HIF_REG_SEL = SEL_HIFADR_LO;       /* select HIFADR register                   */
     HIF_DATA_WR = hif_addr;            /* set of HIFRAM address                    */

     HIF_REG_SEL = SEL_HIFDATA_UP;      /* select HIFDATA register[31:16]           */
     HIF_DATA_WR = (*trans_src_addr);   /* set to HIFDATA register[31:16]           */

     trans_src_addr ++;                 /* storing address is increment(+2)         */

     HIF_DATA_WR = (*trans_src_addr);   /* set to HIFDATA register[15:0]            */
     trans_src_addr ++;                 /* storing address is increment(+2)         */

     HIF_REG_SEL = SEL_HIFMCR_LO;       /* select HIFMCR register[15:0]             */
     HIF_DATA_WR = 0x00A0;              /* set as continuation write mode           */

     HIF_REG_SEL = SEL_HIFDATA_UP;      /* select HIFDATA register[31:16]           */

     for( i=2 ; i<time ; i++){
          HIF_DATA_WR = (*trans_src_addr);   /* write to HIFDATA register(HIFRAM)   */

          trans_src_addr ++;                 /* storing address is increment(+2)    */
     }

     return(trans_src_addr);
}

/***********************************************************/
//     function : hif_boot
//     operation     : SH7618 is booted
//     argument      : non-argument
//     return        : non-return
/***********************************************************/
void hif_boot(void)
{
     //------initialize HIFGSR
     HIF_REG_SEL = SEL_HIFGSR_LO;           /* select HIFGSR register[15:0]         */
     HIF_DATA_WR = 0x0000;                  /* H'0000 is written to HIFGSR register */

     //------boot SH7618
     HIF_REG_SEL = SEL_HIFBCR_LO;           /* select HIFBCR                        */
     HIF_DATA_WR = 0x0000;                  /* clear AC bit of HIFBCR register      */
}

/**************************************************************************/
//     function : sync_7618
//     operation     : synchronization is taken SH7618, and a program is
                       written to HIFRAM
//     argument      : *s_addr ; pointer of source address
//                      h_addr ; HIFRAM header address
//                      b_addr ; HIFRAM buffer address
//                      t_size ; transmission data size
//                      b_size ; HIFRAM buffer size
//     argument      : non-argument
/**************************************************************************/

void sync_7618(unsigned short* s_addr , unsigned short h_addr , unsigned short b_addr ,
unsigned long t_size , unsigned short b_size)
{
     volatile unsigned short status;

     while(1){
          status = HIF_GSR_RD;               /* read from HIFGSR register[15:0]     */
          while(status != 0x0001){           /* wait until HIFGSR.HIFS=1            */
               status = HIF_GSR_RD;          /* read from HIFGSR register[15:0]     */
          }

     //------preparation write to header
          HIF_REG_SEL = SEL_HIFADR_LO;       /* select HIFADR register[15:0]        */
          HIF_DATA_WR = h_addr;              /* set of HIFRAM header address        */
          HIF_REG_SEL = SEL_HIFDATA_UP;      /* select HIFDATA register[31:16]      */
          HIF_DATA_WR = 0;                   /* set to HIFDATA register[31:16]      */

     //------usual transmission
          if(t_size > b_size)
          {
     //------transmission data size is written to header
               HIF_DATA_WR = b_size;         /* write to HIFDATA register[15:0]     */

     //------write to HIFRAM buffer
               s_addr = write_HIFRAM(s_addr,b_addr,b_size);

     //------transmission end process
               HIF_REG_SEL = SEL_HIFGSR_LO;          /* select HIFGSR register[15:0] */
               HIF_DATA_WR = 0x0000;                 /* set to HIFGSR register[15:0] */
                                                    // TEND=0 , HIFS=0
          }

     //------last transmission
          else
          {
     //------transmission data size is written to header
               HIF_DATA_WR = (unsigned short)t_size;
                                              /* write to HIFDATA register [15:0]   */

     //------write to HIFRAM buffer
               s_addr = write_HIFRAM(s_addr,b_addr,b_size);

     //------transmission end process
               HIF_REG_SEL = SEL_HIFGSR_LO;          /* select HIFGSR register[15:0] */
               HIF_DATA_WR = 0x0000;                 /* set to HIFGSR register[15:0] */
                                                    // TEND=0 , HIFS=0

               break;                                /* escape from loop             */
           }
```

```
        //------calculate transmission data size
             t_size = t_size - b_size;
        }

        status = HIF_GSR_RD;                        /* read from HIFGSR register[15:0]     */

        while( status==0x0000 )                     /* wait until HIFGSR.HIFS=1            */
        {
             status = HIF_GSR_RD;                   /* read from HIFGSR register[15:0]     */
        }

        HIF_DATA_WR = 0x0002;                       /* set to HIFGSR register[15:0]        */
                                                    // TEND=1 , HIFS=0
   }
```

2.3 Program Transfer Using Two Banks of HIFRAM

2.3.1 Overview

This section provides a sample task that uses two banks of HIFRAM for high-speed transfer of a large program.

When two banks of HIFRAM are used, writing a program to HIFRAM and transferring a program from HIFRAM to SDRAM can be performed concurrently, speeding up program transfer.

2.3.2 Specifications

(1) The SH7641 is connected to the HIF, and the SH7618 is started up in the HIF boot mode.
(2) After the SH7618 is activated, it transfers a transfer program to the internal RAM via the HIFRAM, and executes it.
(3) Using two banks of HIFRAM, the transfer program transfers an application program to SDRAM.
(4) During the transfer, the SH7618 and the SH7641 access different banks of HIFRAM.
(5) The HIF General Status Register (HIFGSR) is used to synchronously switch the banks accessed by the SH7618 and SH7641.
(6) The SH7641 writes the application program, including a header that indicates the size of data to be transferred, to HIFRAM.
(7) The SH7618 transfers the amount of data specified by the header from HIFRAM to SDRAM.
(8) After the application program is transferred to SDRAM, the SH7618 executes the application program in SDRAM.
(9) In this sample task, the SH7641 operates with a 100-MHz CPU clock, a 50-MHz external bus clock, and a 25-MHz peripheral clock. Also, the SH7618 operates with a 100-MHz internal clock, a 50-MHz external bus clock, and a 50-MHz peripheral clock.

Note: This sample task describes only program transfer using two banks of HIFRAM. For details of the sequence from startup in the HIF boot mode to execution of the program transferred to the internal RAM, see section 2.2, Program Transfer to Internal RAM in HIF Boot Mode.

Figure 16 shows an example of connection between the SH7618, the SH7641, and SDRAM.

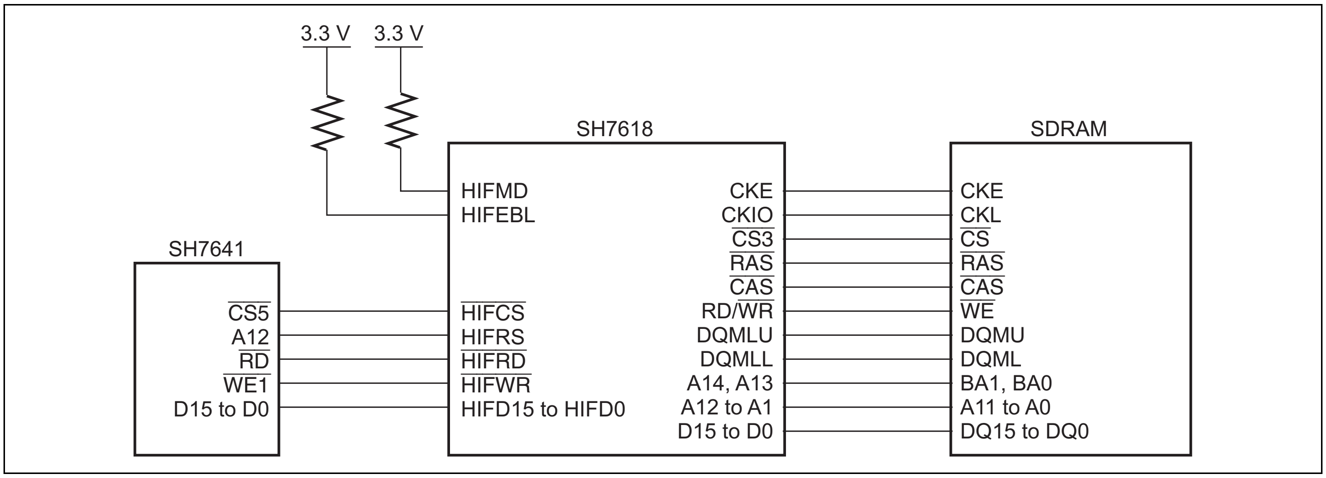
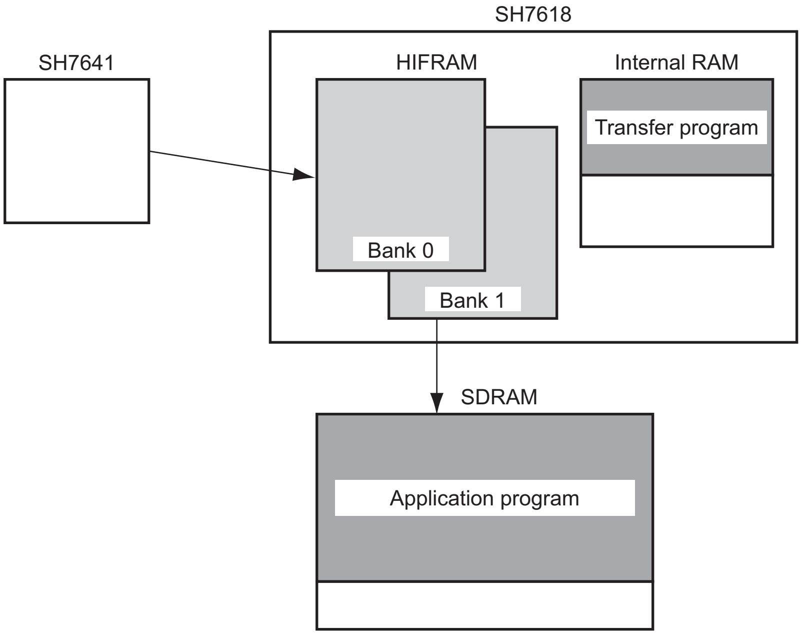


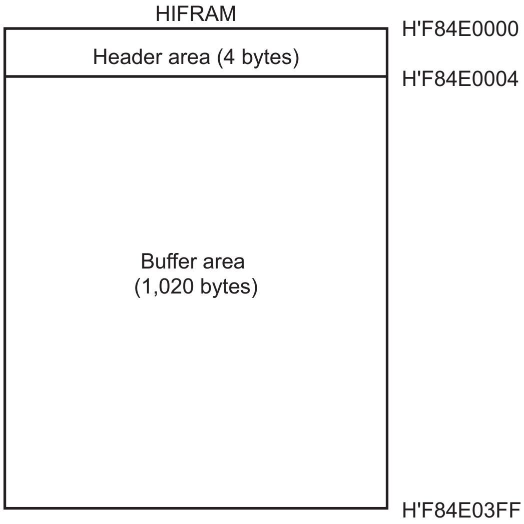
Figure 16 Connection Example



Note: The program in the internal RAM is executed, and the program is transferred to SDRAM.

Figure 17 Program Transfer Using Two Banks of HIFRAM

Figure 18 shows the HIFRAM memory map.



Note: The header area stores the size of the data to be transferred in one transfer session.
This memory map is common to banks 0 and 1.

Figure 18 HIFRAM Memory Map

Figure 19 shows the relationship between the buffer size and the size of a program transferred from HIFRAM to SDRAM.

In this sample task, the SH7641 writes the size of the program to the header area. The SH7618 references the data in the header area and transfers the program from the HIFRAM buffer to SDRAM.

To transfer the application program faster, the SH7618 transfers data in longword units. The data size during a transfer is determined by the size of the program being transferred.

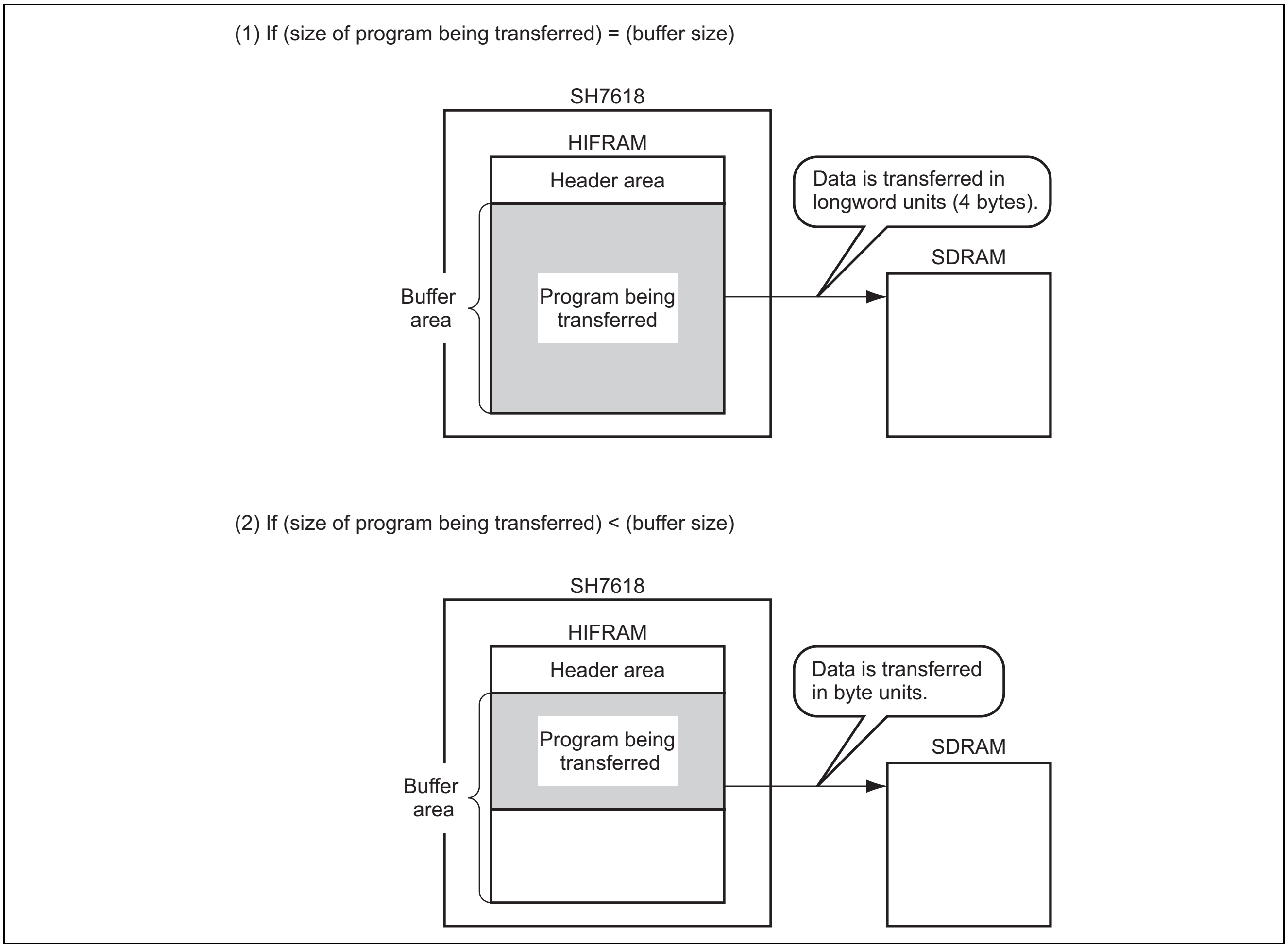

(1) If (size of program being transferred) = (buffer size)
SH7618
HIFRAM
Header area
Buffer area
Program being transferred
Data is transferred in longword units (4 bytes).
SDRAM
(2) If (size of program being transferred) < (buffer size)
SH7618
HIFRAM
Header area
Buffer area
Program being transferred
Data is transferred in byte units.
SDRAM

Figure 19 Transfer from HIFRAM to SDRAM

The user can define the HIFGSR register bits. In this sample task, bit 1 is defined as TEND and bit 0 is defined as HIFS to synchronize transfer sessions between the SH7641 and the SH7618. Table 17 is a list of HIFGSR register definitions in the sample task.

Table 17 HIFGSR Register Definitions

Bit	Bit Name	Setting Value	Function
31 to 16	—	All 0	Reserved
15 to 2	STATUS15 to STATUS2	All 0	General status bits These bits are not used in this sample task.
1	TEND	0	Transmit end This bit indicates whether program transfer has ended. The SH7618 references this bit to verify that program transfer has ended, and executes the program transferred to internal RAM. The SH7641 sets this bit to 1 after all program transfer sessions are complete. 1: All program transfer sessions have ended. The SH7618 executes the program transferred to internal RAM. 0: Program transfer is in progress. Part of the program being transferred to the internal RAM remains on the SH7641 side.
0	HIFS	0	HIFRAM status This bit indicates the status of HIFRAM. The SH7618 and the SH7641 reference this bit to perform a transfer. The SH7618 can set this bit to 1 only when a transfer from the SH7641 to HIFRAM is possible. The SH7641 can clear this bit to 0 only when a program transfer to the HIFRAM buffer has ended. 1: The SH7641 can perform a write to HIFRAM. The SH7618 is in the standby state. 0: The SH7641 cannot perform a write to HIFRAM. The SH7618 transfers a program from the HIFRAM buffer to internal RAM.

2.3.3 Operation

Figure 20 shows the HIFGSR values, and indicates the SH7618 and SH7641 operations for those HIFGSR values. Table 18 describes the operations in detail. Figure 21 shows how the operation differs depending on the values of the BMD and BSEL bits of the HIFSCR register.

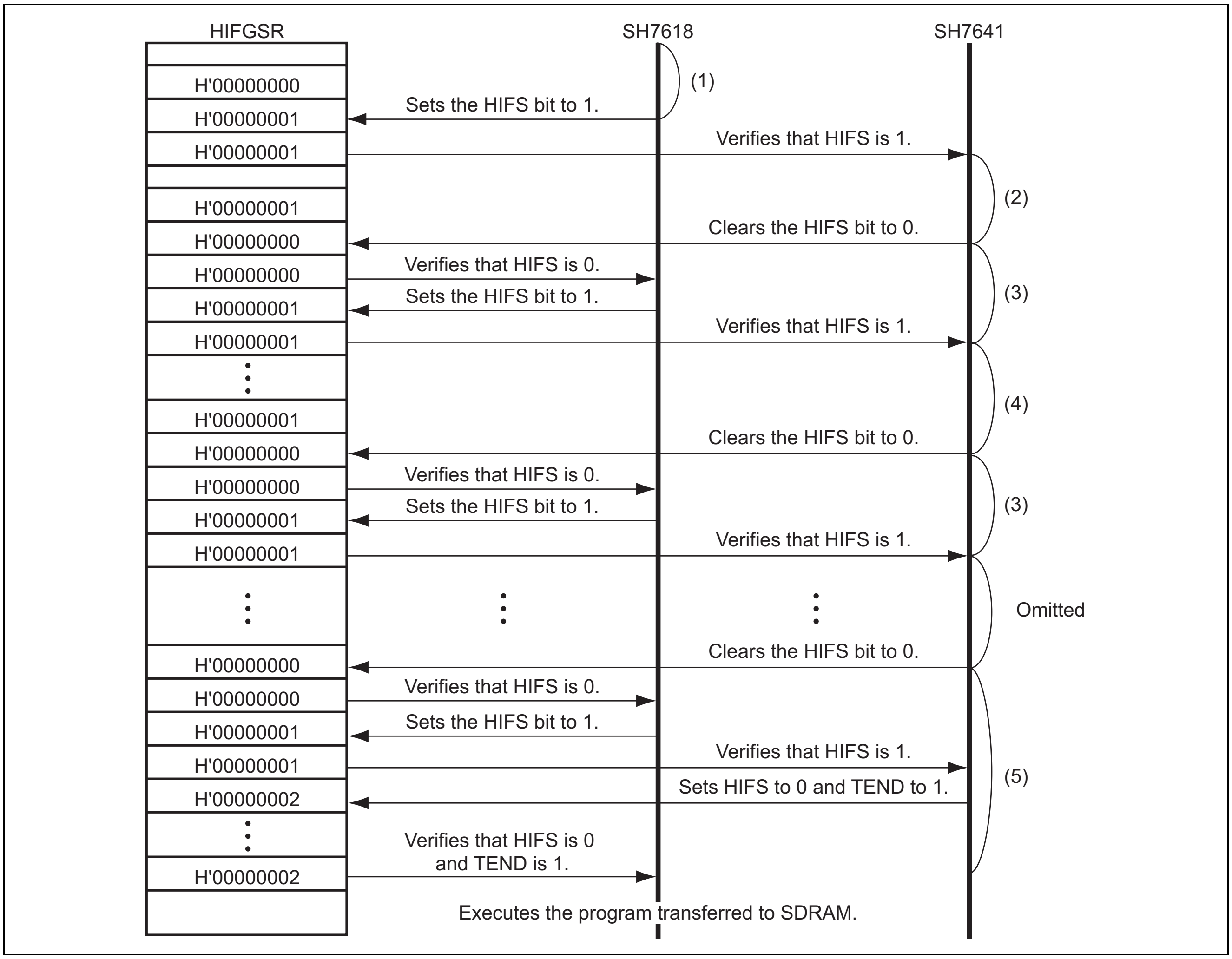
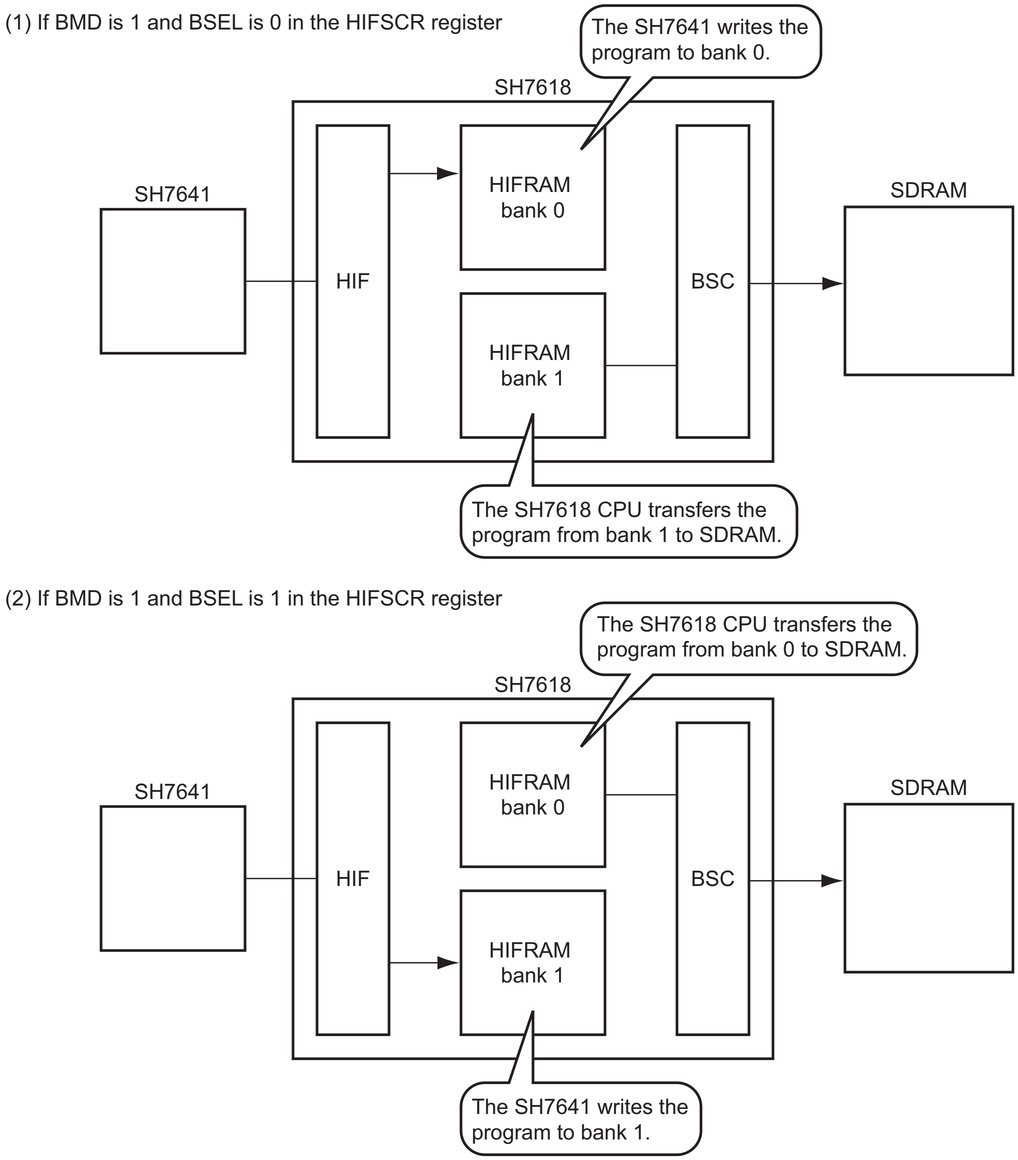


Figure 20 HIFGSR Values and Operations

Table 18 Description of SH7618 and SH7641 Operations

	SH7618 Operation	SH7641 Operation
(1)	• The SH7618 initializes both banks 0 and 1 of HIFRAM. • The SH7618 sets the HIFS bit of the HIFGSR register to 1.	• The SH7641 waits while the HIFS bit of the HIFGSR register is 0.
(2)	• The SH7618 waits while the HIFS bit of the HIFGSR register is 1.	• The SH7641 verifies that the HIFS bit of the HIFGSR register has been set to 1. • The SH7641 writes program to the HIFRAM buffer. • After the program is written, the SH7641 clears the HIFS bit of the HIFGSR register to 0.
(3)	• The SH7618 verifies that the HIFS bit of the HIFGSR register is cleared to 0. • The SH7618 inverts the BSEL bit of the HIFSCR register (to switch the banks accessed by the SH7618 and SH7641). • The SH7618 sets the HIFS bit of the HIFGSR register to 1. • The SH7618 transfers the program that was written by the SH7641 from the HIFRAM buffer to SDRAM.	• The SH7641 waits while the HIFS bit of the HIFGSR register is 0.
(4)	• After the program is transferred to SDRAM, the SH7618 clears the HIFRAM header. • The SH7618 waits while the HIFS bit of the HIFGSR register is 1.	• The SH7641 verifies that the HIFS bit of the HIFGSR register has been set to 1. • The SH7641 writes the program to the HIFRAM buffer. • After the program is written, the SH7641 clears the HIFS bit of the HIFGSR register to 0.
(5)	• The SH7618 verifies that the HIFS bit of the HIFGSR register is cleared to 0. • The SH7618 inverts the BSEL bit of the HIFSCR register (to switch banks accessed by the SH7618 and the SH7641). • The SH7618 sets the HIFS bit of the HIFGSR register to 1. • The SH7618 transfers the program that was written by the SH7641 from the HIFRAM buffer to SDRAM. • The SH7618 verifies that the HIFS bit of the HIFGSR register is 0 and that the TEND bit of the HIFGSR register is 1. It then starts processing of the program transferred to SDRAM.	• When all program write sessions have been completed, the SH7641 clears the HIFS bit of the HIFGSR register to 0. • The SH7641 waits while the HIFS bit of the HIFGSR register is 0. • The SH7641 verifies that the HIFS bit of the HIFGSR register has been set to 1. • The SH7641 writes HIFS = 0 and TEND = 1 in the HIFGSR register.


(1) If BMD is 1 and BSEL is 0 in the HIFSCR register
The SH7641 writes the program to bank 0.
SH7618
SH7641
HIF
HIFRAM bank 0
HIFRAM bank 1
BSC
SDRAM
The SH7618 CPU transfers the program from bank 1 to SDRAM.
(2) If BMD is 1 and BSEL is 1 in the HIFSCR register
The SH7618 CPU transfers the program from bank 0 to SDRAM.
SH7618
SH7641
HIF
HIFRAM bank 0
HIFRAM bank 1
BSC
SDRAM
The SH7641 writes the program to bank 1.

Note: Because the SH7618 CPU and the SH7641 can concurrently access different banks, writing to HIFRAM of the SH7641 and transfer to SDRAM by the SH7618 CPU are performed at the same time.

* BSC: Bus State Controller

Figure 21 HIFSCR Register Settings and Operations

2.3.4 Description of Software

(1) Modules

Table 19 is a list of SH7618 modules used in this sample task. Table 20 is a list of SH7641 modules used in this sample task.

Table 19 SH7618 Modules

Module Name	Label Name	Description
U memory main routine	uram_main	Performs initial setting of the SH7618 and transfers the program written by the SH7641 from the HIFRAM buffer to internal RAM. Note: This module is executed in internal RAM (U memory) of the SH7618.
Transfer program execution routine	jump_uram	Moves the program counter to internal RAM, and executes the transferred program. Note: This module is written in assembly language.

Table 20 SH7641 Modules

Module Name	Label Name	Description
SH7641 main routine	main_7641	Sets the external bus and pins, and calls modules.
HIFRAM consecutive write routine	write_HIFRAM	Writes a boot program to HIFRAM.
SH7618 startup routine	hif_boot	Starts up the SH7618 in the HIF boot mode.
HIFRAM buffer write routine	sync_7618	Writes the boot program to the HIFRAM buffer in synchronization with the SH7618.

(2) Internal registers used

Table 21 is the lists of SH7618 internal registers used in this sample task.

Table 21 SH7618 Internal Registers Used

Register Name			Setting	
	Bit	**Bit Name**	**Value**	**Function**
HIFADR				HIF Address Register
	31 to 10	—	All 0	Reserved
	9 to 2	A9 to A2	—	Specifies the target HIFRAM address. These bits specify, on a 32-bit boundary, the address in HIFRAM to be accessed by the external device.
	1	—	0	Reserved
	0	—	0	
HIFDATA				HIF Data Register
	31 to 0	D31 to D0	—	32-bit data. These bits are used for an access to HIFRAM from the external device.
HIFIDX				HIF Index Register
	31 to 8	—	All 0	Reserved
	7 to 2	REG5 to REG0	—	Select the HIF internal registers. These bits are used to select the HIF registers to be accessed by the external device.
	1	BYTE1	—	Select the byte in an HIF internal register.
	0	BYTE0	—	These bits are used to specify the word position when the external device accesses an HIF internal register.

Register Name		Setting	
Bit	Bit Name	Value	Function
HIFMCR			HIF Memory Control Register
31 to 8	—	All 0	Reserved
7	LOCK	—	Lock bit This bit is used when the external device performs consecutive access to HIFRAM.
6	—	0	Reserved
5	WT	—	Write bit When this bit is set to 1, the HIFDATA value is written to the HIFRAM location corresponding to the HIFADR register.
4	—	0	Reserved
3	RD	—	Read bit When this bit is set to 1, the HIFRAM data corresponding to HIFADR is fetched into HIFDATA.
2, 1	—	All 0	Reserved
0	AI/AD	0	Address auto-increment/decrement When LOCK = 1 and AI/AD = 0, each time the SH7641 accesses the HIFDATA register, the HIFADR register value is incremented (+4) so that data can be written to, or read from, consecutive HIFRAM addresses.
HIFGSR			HIF General Status Register
31 to 16	—	All 0	Reserved
15 to 2	STATUS15 to STATUS2	All 0	General status bits These bits are not used in this sample task.
1	TEND	—	Transmit end This bit indicates the program transfer status. TEND=1: The SH7618 executes the program transferred to internal RAM. TEND=0: The SH7618 transfers the program to internal RAM.
0	HIFS	—	HIFRAM status This bit indicates the status of HIFRAM. HIFS=0: The SH7641 can perform a write to HIFRAM. HIFS=1: The SH7641 cannot transfer a program to HIFRAM.

Register Name		Setting	
Bit	Bit Name	Value	Function
HIFSCR			HIF Status Control Register
31 to 12	—	0	Reserved
11	DMD	0	DREQ mode Controls the HIFDREQ pin's assertion mode in combination with the DPOL bit. This bit is not used in this sample task.
10	DPOL	0	DREQ polarity Controls the HIFDREQ pin's assertion mode in combination with the DMD bit. This bit is not used in this sample task.
9	BMD	—	HIFRAM bank mode This bit and the BSEL bit are used to determine the banks of HIFRAM to be accessed by the SH7618 CPU and the external device. If both the SH7618 CPU and the external device attempt to access the same bank at the same time, the external device has precedence. BMD = 0 and BSEL = 0: Both the SH7618 CPU and the external device access bank 0. BMD = 0 and BSEL = 1: Both the SH7618 CPU and the external device access bank 1. BMD = 1 and BSEL = 0: The SH7618 CPU accesses bank 1, and the external device accesses bank 0. BMD = 1 and BSEL = 1: The SH7618 CPU accesses bank 0, and the external device accesses bank 1.
8	BSEL	—	HIFRAM bank select This bit and the BMD bit are used to determine the banks of HIFRAM to be accessed by the SH7618 CPU and the external device.
7	—	0	Reserved
6	—	1	Reserved
5	MD1	1	HIF mode 1 Indicates whether the SH7618 has been started up in the HIF boot mode. If MD1 = 1, the SH7618 has been started up in the HIF boot mode.
4 to 2	—	0	Reserved
1	EDN	0	Endian for HIFRAM access Specifies the byte order when the SH7618 CPU accesses HIFRAM. 0: Big endian (MSB first)
0	BO	0	Byte order for access to all HIF registers, including HIFDATA Specifies the byte order when the external device accesses all HIF registers, including HIFDATA. 0: Big endian (MSB first)

Table 22 is a list of SH7641 internal registers used in this sample task.

Table 22 SH7641 Internal Registers Used

Register Name		Setting	
Bit	Bit Name	Value	Function
CS5ABCR			CS5A Space Bus Control Register
10	BSZ1	1	Specify the data bus width for accessing the CS5A space.
9	BSZ0	0	When BSZ[1,0] = B'10, the data bus width is set to 16 bits.
CS5AWCR			CS5A Space Wait Control Register
31 to 19	—	All 0	Reserved
18	WW2	0	Number of write access wait cycles
17	WW1	0	These bits specify the number of wait cycles to be inserted during a write access.
16	WW0	0	
			When WW[2-0] = B'000, the same number of wait cycles as the number of read access wait cycles specified by the WR bits are inserted.
15 to 13	—	All 0	Reserved
12	SW1	0	Number of delay cycles from address, $\overline{CS}$ assertion to $\overline{RD}$, $\overline{WEn}$ (BEn) assertion
11	SW0	1	
			While SW[1,0] = B'01, 1.5 wait cycles are inserted from address and $\overline{CS}$ assertion to $\overline{RD}$ and $\overline{WEn}$ assertion.
10	WR3	1	Number of read access wait cycles
9	WR2	0	These bits specify the number of wait cycles to be inserted during a read access. In this sample task, the number of wait cycles specified by these bits are inserted during read and write accesses.
8	WR1	0	
7	WR0	0	
			Before startup: WR[3-0] = B'1000 (10 wait cycles are inserted)
			After startup: WR[3-0] = B'0010 (2 wait cycles are inserted)
6	WM	0	Specifies the external wait mask. These bits specify whether to enable or disable external wait input. When WM = 0, the external wait input is enabled.
5 to 2	—	All 0	Reserved
1	HW1	0	Number of wait cycles to be inserted from $\overline{RD}$, $\overline{WEn}$ negation to address, $\overline{CS}$ negation
0	HW0	1	
			While HW[1,0] = B'01, 1.5 wait cycles are inserted from $\overline{RD}$ and $\overline{WEn}$ negation to address and $\overline{CS}$ negation.
PCCR			Port C Control Register
3	PC1MD2	1	PC1 modes 2 and 1
2	PC1MD1	1	When PC1MD[2,1] = B'11, the PTC1 pin is set to the CS5A function.

(3) Variables

Table 23 is a list of variables used in the SH7618 program in this sample task. Table 24 is a list of variables used in the SH7641 program in this sample task.

Table 23 Variables Used in the SH7618 Program

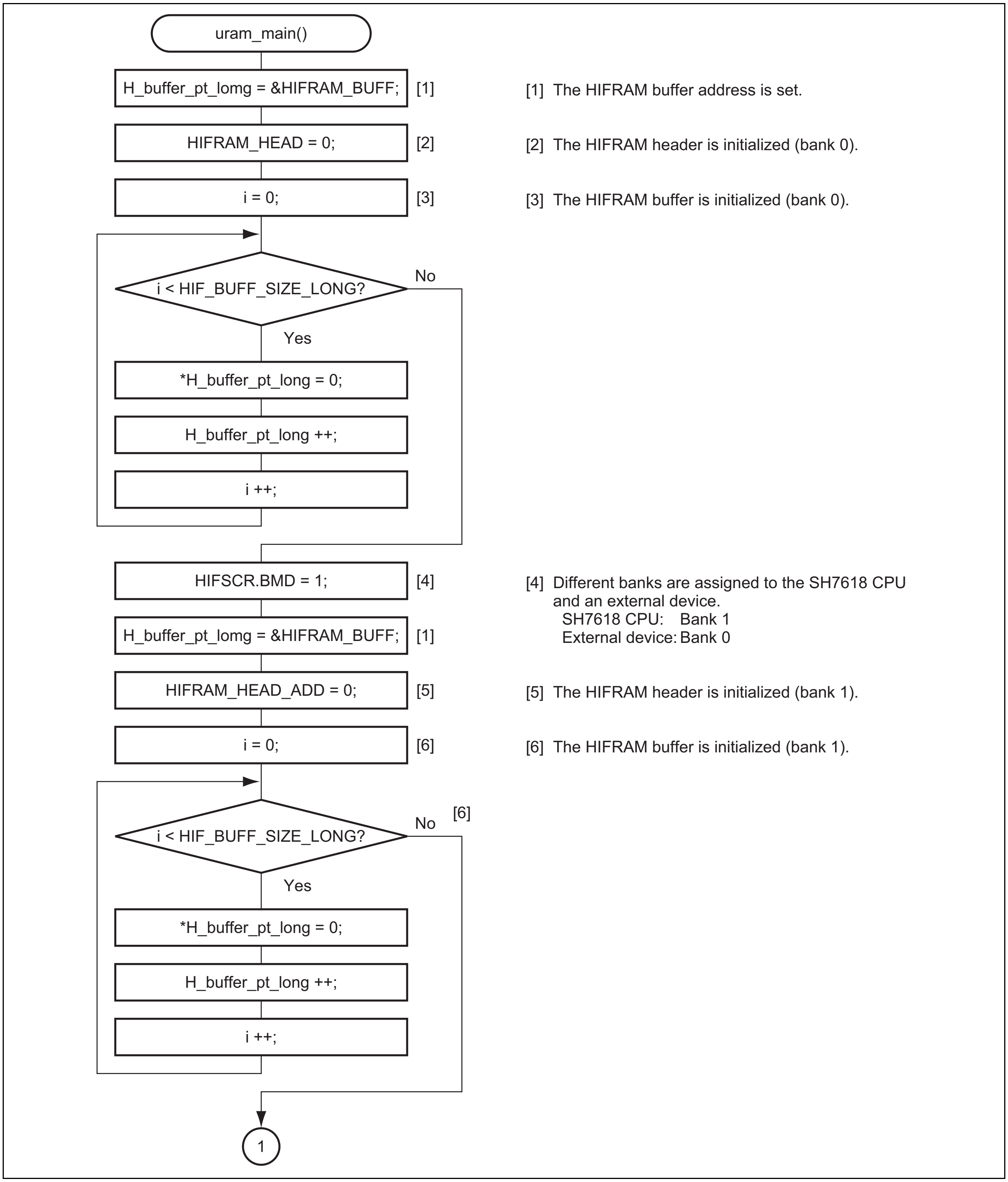
Variable	Description	Data Length	Initial Value	Used In
t_count	Number of transfer sessions for a transfer in longword units	4 bytes	—	U memory main routine
*Sdram_address_ pt_byte	Pointer that indicates the program transfer destination (internal RAM) address for a transfer in byte units	1 byte	—	U memory main routine
*Sdram_address_ pt_long	Pointer that indicates the program transfer destination (internal RAM) address for a transfer in longword units	4 bytes	—	U memory main routine
*h_buffer_pt_byte	Pointer that indicates the program transfer source (HIFRAM buffer) address for a transfer in byte units	1 byte	—	U memory main routine
*h_buffer_pt_long	Pointer that indicates the program transfer source (HIFRAM buffer) address for a transfer in longword units	4 bytes	—	U memory main routine

Table 24 Variables Used in the SH7641 Program

Variable	Description	Data Length	Initial Value	Used In
*trans_src_addr	Pointer that indicates the transfer source (boot program storage) address	2 bytes	—	HIFRAM consecutive write routine
hif_addr	HIF address to startwriting from	2 bytes	H'0000	HIFRAM consecutive write routine
t_size	Size of the program to be transferred	2 bytes	H'300	HIFRAM consecutive write routine
*s_addr	Pointer that indicates the transfer source (boot program storage) address	2 bytes	—	HIFRAM consecutive write routine
h_addr	HIFRAM header address	2 bytes	H'0000	HIFRAM buffer write routine
b_addr	HIFRAM buffer address	2 bytes	H'0004	HIFRAM buffer write routine
t_size	Total size of the program to be transferred	4 bytes	H'200000	HIFRAM buffer write routine
b_size	HIFRAM buffer size (size of data transferred in one transfer session)	2 bytes	H'3FC	HIFRAM buffer write routine

2.3.5 Flow Chart

(1) U memory main routine



[1] The HIFRAM buffer address is set.

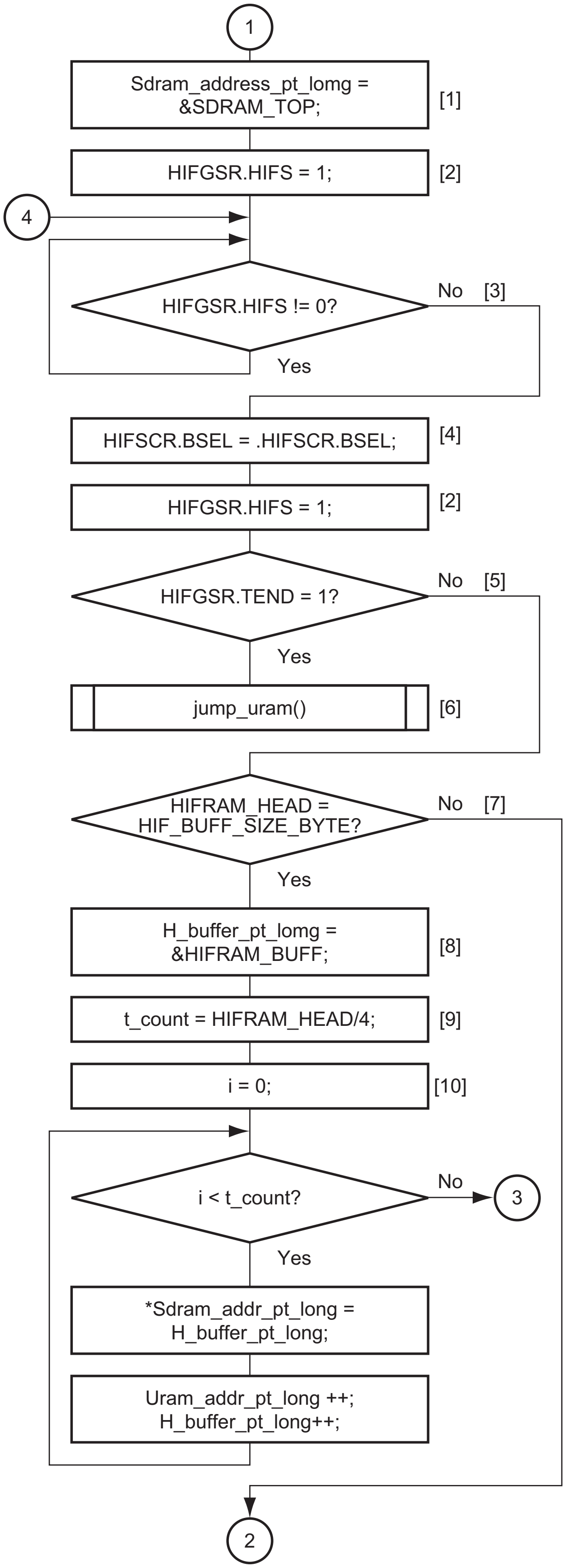
[2] The HIFRAM header is initialized (bank 0).

[3] The HIFRAM buffer is initialized (bank 0).

[4] Different banks are assigned to the SH7618 CPU and an external device.
SH7618 CPU: Bank 1
External device: Bank 0

[5] The HIFRAM header is initialized (bank 1).

[6] The HIFRAM buffer is initialized (bank 1).



[1] The transfer-destination SDRAM address is set.

[2] The HIFS bit of the HIFGSR register is set to 1 to permit the external device to write to HIFRAM.

[3] Whether the write by the external device to HIFRAM has ended is checked.

[5] Whether the entire program has been transferred is decided.
If TEND=1, the transfer has been completed.

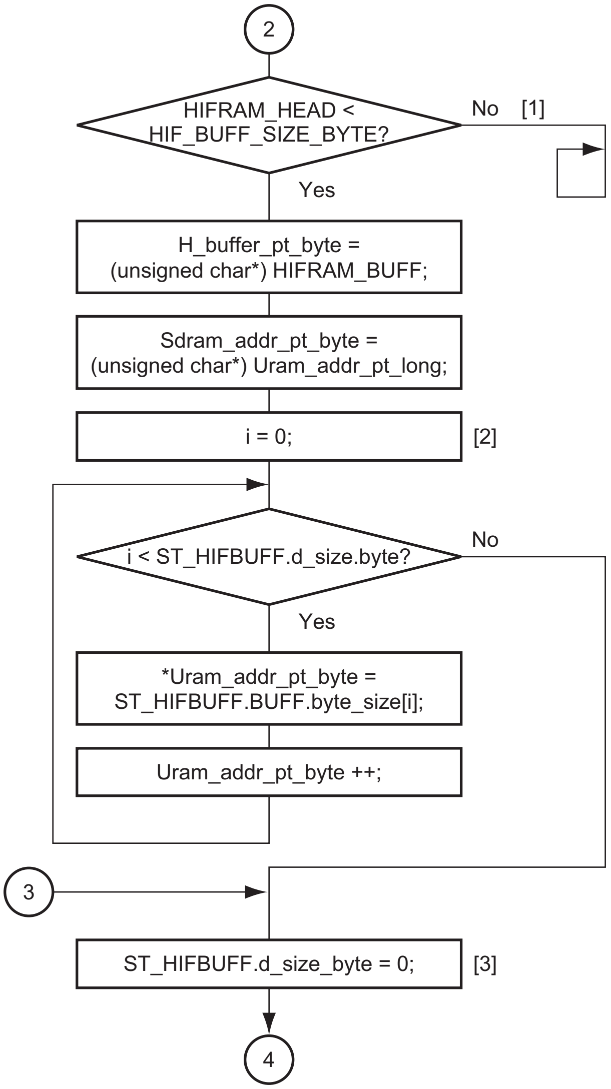
[6] The routine that executes the transferred program is called.

[7] Whether data should be transferred in longword or byte units is decided.
- If transfer-size = buffer-size, then data is transferred in longword units.
- If transfer-size < buffer-size, then data is transferred in byte units.
- In other cases, an error (endless loop) occurs.

[8] The HIFRAM buffer address is set.

[9] The transfer count for a transfer in longword units is set.

[10] Data is transferred in longword units from the HIFRAM buffer to SDRAM.


2
HIFRAM_HEAD <
HIF_BUFF_SIZE_BYTE?
No
[1]
Yes
H_buffer_pt_byte =
(unsigned char*) HIFRAM_BUFF;
Sdram_addr_pt_byte =
(unsigned char*) Uram_addr_pt_long;
i = 0;
[2]
i < ST_HIFBUFF.d_size.byte?
No
Yes
*Uram_addr_pt_byte =
ST_HIFBUFF.BUFF.byte_size[i];
Uram_addr_pt_byte ++;
3
ST_HIFBUFF.d_size_byte = 0;
[3]
4

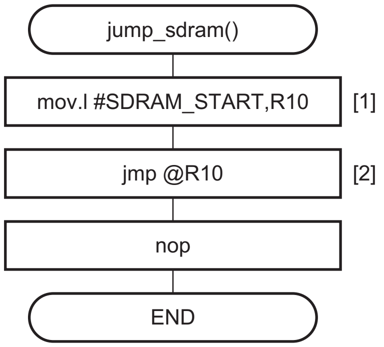
[1] Whether data should be transferred in longword or byte units is decided.

- If transfer-size = buffer-size, then data is transferred in longword unit.
- If transfer-size < buffer-size, then data is transferred in byte units.
- In other cases, an error (endless loop) occurs.

[2] Data is transferred in byte units from the HIFRAM buffer to SDRAM.

[3] The HIFRAM header is initialized.

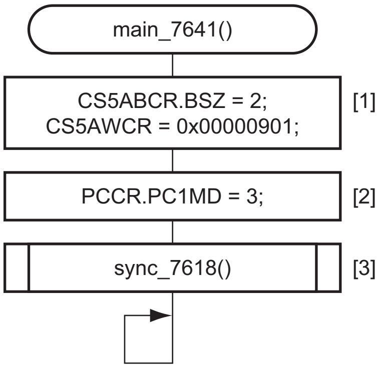
(2) Transfer program execution routine



[1] The execution start address of the transferred program is set.
SDRAM_START: Start address of SDRAM

[2] Control branches to the set address.

(3) SH7641 main routine

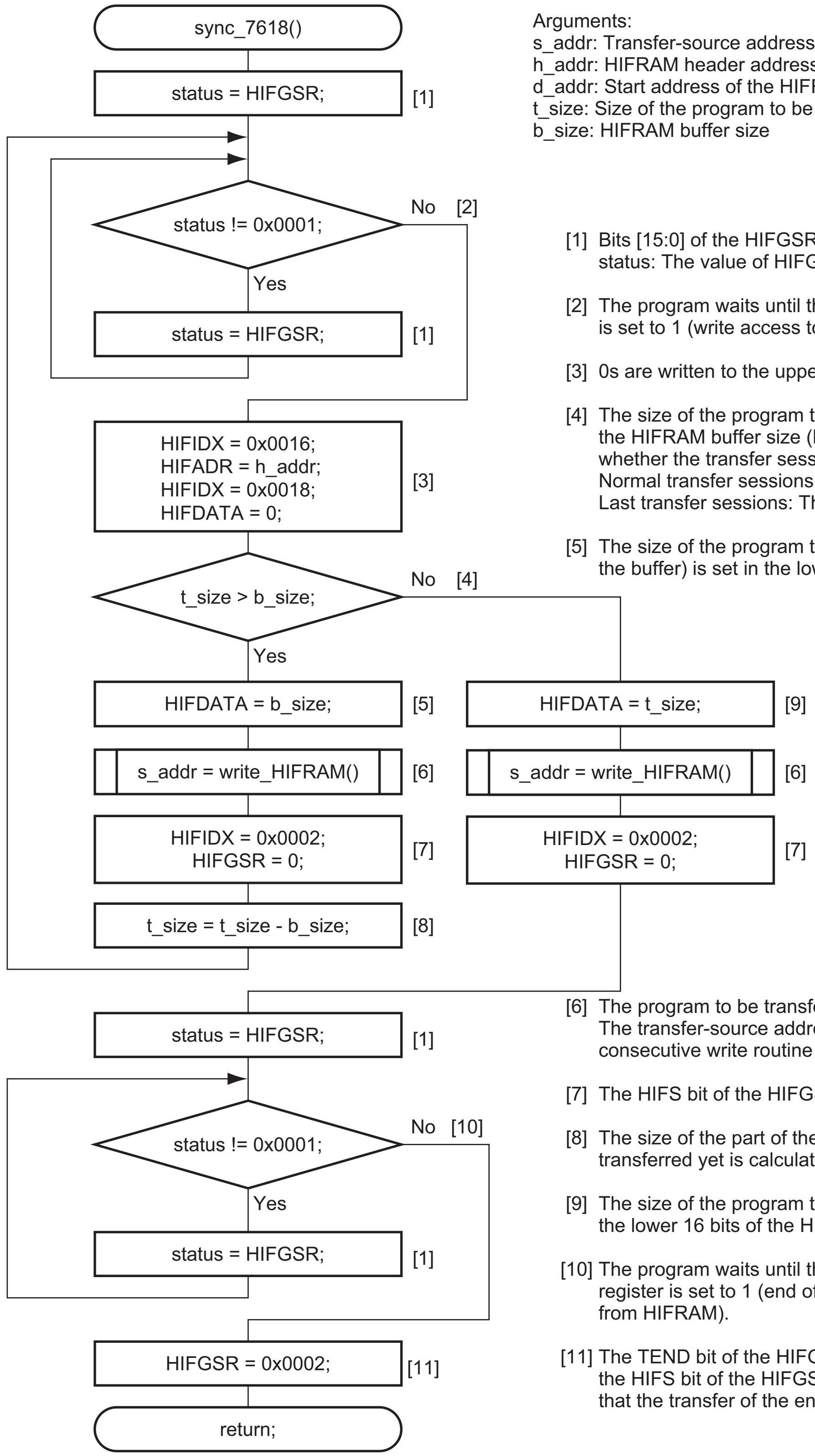


[1] CS5A space bus settings
The data bus width is set to 16 bits.
- 1.5 wait cycles are inserted from address and $\overline{\text{CS5A}}$ assertion to $\overline{\text{RD}}$ and $\overline{\text{WEn}}$ assertion.
- 1.5 wait cycles are inserted from $\overline{\text{RD}}$ and $\overline{\text{WEn}}$ negation to address and $\overline{\text{CS5A}}$ negation.
- 10 wait cycles are inserted during read/write access.

[2] The pin function is set to $\overline{\text{CS5A}}$.

[3] The TEND bit of the HIFGSR register is set to 1 and the HIFS bit of the HIFGSR register is set to 0 in order to indicate that the transfer of the entire program is completed.

(4) HIFRAM buffer write routine



Arguments:
s_addr: Transfer-source address
h_addr: HIFRAM header address
d_addr: Start address of the HIFRAM buffer
t_size: Size of the program to be transferred
b_size: HIFRAM buffer size

[1] Bits [15:0] of the HIFGSR register are read.
status: The value of HIFGSR is stored.

[2] The program waits until the HIFS bit of the HIFGSR register is set to 1 (write access to the HIFRAM buffer is possible).

[3] 0s are written to the upper 16 bits of the HIFRAM header.

[4] The size of the program to be transferred (t_size) and the HIFRAM buffer size (b_size) are compared to determine whether the transfer session was the last one.
Normal transfer sessions: The program continues with (5).
Last transfer sessions: The program branches to step (9).

[5] The size of the program to be transferred (same size as the buffer) is set in the lower 16 bits of the HIFRAM header.

[6] The program to be transferred is in to the HIFRAM buffer. The transfer-source address used in the HIFRAM consecutive write routine is stored in s_addr.

[7] The HIFS bit of the HIFGSR register is reset to 0.

[8] The size of the part of the program that has not been transferred yet is calculated.

[9] The size of the program to be transferred is set in the lower 16 bits of the HIFRAM header.

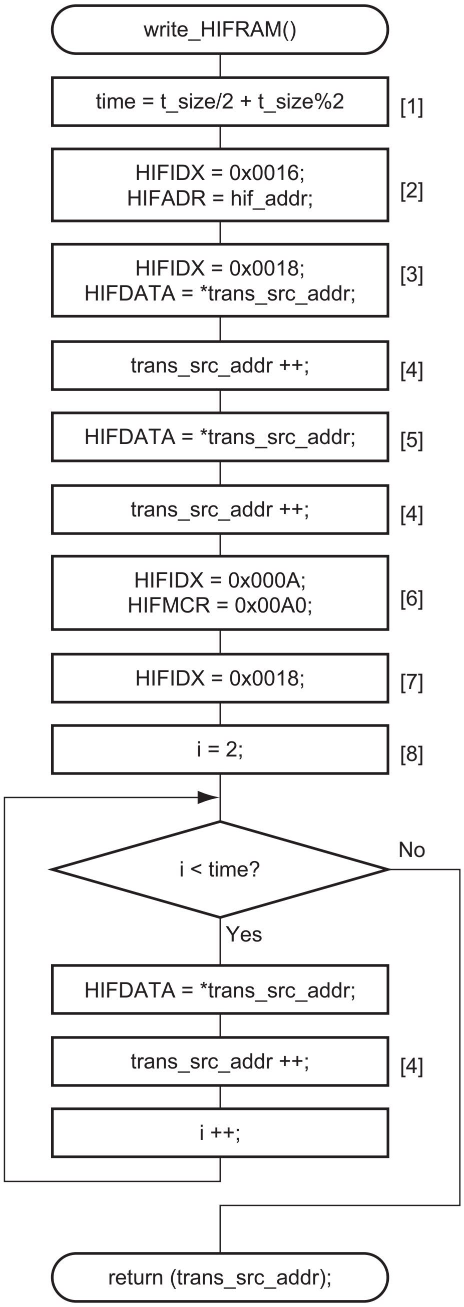
[10] The program waits until the HIFS bit of the HIFGSR register is set to 1 (end of the last session in a transfer from HIFRAM).

[11] The TEND bit of the HIFGSR register is set to 1 and the HIFS bit of the HIFGSR register is set to 0 to indicate that the transfer of the entire program has been completed.

(5) HIFRAM consecutive write routine

Arguments:
*trans_src_addr: Pointer to the transfer-source address
hif_addr: HIFRAM address where writing starts
t_size: Size of the program to be transferred

Return value:
trans_src_addr: Transfer-source address



[1] The number of writes to HIFRAM is set.
time: Number of transfers

[2] The HIFIDX register is used to specify the lower 16 bits of the HIFADR register, and the HIFRAM address where writing starts is set in the HIFADR register.

[3] The HIFIDX register is used to specify the upper 16 bits of the HIFDATA register, and the write data is set in bits [31:16] of the HIFDATA register.

[4] The transfer-source address is incremented (+2).

[5] The write data is set in bits [15:0] of the HIFDATA register.

[6] The HIFIDX register is used to specify the lower 16 bits of the HIFMCR register, and HIFRAM consecutive write mode is set. (The HIFADR register value is incremented automatically each time the HIFDATA register is accessed.)

[7] The HIFIDX register is used to specify bits [31:16] of the HIFDATA register.

[8] Writing of data to the HIFDATA register continues until the writing of the boot program to HIFRAM is completed.
(The first 4 bytes have already been written, so loop variable "i" starts at 2.)

2.3.6 Program Listing

(1) SH7618 program 1

```
/*************************************************************/
// SH7618 HIF boot mode application note
//     Program transmission which used two bank of HIFRAM
//          CPU      : SH7618,SH-2,Big Endian
//          Clock    : External input = 25MHz
//                     CPU clock = 100MHz
//                     External BUS clock = 50MHz
//                     Peripheral clock = 50MHz
//          Written  : '04/4 Rev.2.0
/*************************************************************/

//------ Symbol Definition ---------------------------
#define HIF_BUFF_SIZE_BYTE   0x3FC         /* HIFRAM buffer size(Byte)              */
#define HIF_BUFF_SIZE_LONG   (HIF_BUFF_SIZE_BYTE/4)
                                           /* HIFRAM buffer size(long word access)  */

//------ Definition of HIF Register ------------------
union st_hifgsr{             /* definition of HIFGSR                                */
     unsigned long LONG;          // Long Access
     struct {                     // Bit Access
          unsigned long :24;      // reserve
          unsigned long :5;       // no use
          unsigned long BS:1;     // BS
          unsigned long TEND:1;   // FIN
          unsigned long HIFS:1;   // HIFS
     } BIT;
};

union st_hifscr{             /* definition of HIFSCR                                */
     unsigned long LONG;          // Long Access
     struct {                     // Bit Access
          unsigned long :20;      // reserve
          unsigned long DMD:1;    // DREQ mode
          unsigned long DPOL:1;   // DREQ polarity
          unsigned long BMD:1;    // HIFRAM bunk mode
          unsigned long BSEL:1;   // HIFRAM bunk select
          unsigned long :2;       // reserve
          unsigned long MD1:1;    // HIF mode
          unsigned long :3;       // reserve
          unsigned long EDN:1;    // HIFRAM endian
          unsigned long BO:1;     // HIF byte order
     } BIT;
};

//------ Function Definition --------------------------
void uram_main(void);

extern void jump_sdram(void);

//------------------------------------------------
```

```
#define ST_HIFGSR   (*(volatile union st_hifgsr*)0xF84D0004)
                                       /* SH7618 HIFGSR register address            */
#define ST_HIFSCR   (*(volatile union st_hifscr*)0xF84D0008)
                                       /* SH7618 HIFSCR register address            */

#define SDRAM_TOP (*(volatile unsigned long *)0xAC000000)
                                       /* SDRAM top address,Non Cache area          */

#define HIFRAM_HEAD (*(volatile unsigned long *)0xF84E0000)
                                       /* HIFRAM header area address                */
#define HIFRAM_BUFF_LONG (*(volatile unsigned long *)0xF84E0004)
                                       /* HIFRAM buffer area when long-word access  */
#define HIFRAM_BUFF_BYTE (*(volatile unsigned char *)0xF84E0004)
                                       /* HIFRAM buffer area when byte access       */

unsigned char* Sdram_address_pt_byte;  /* pointer of SDRAM address(byte access)     */
unsigned long* Sdram_address_pt_long;  /* pointer of SDRAM address(long word access)*/

/**********************************************************/
/*          Main routine                                  */
/**********************************************************/
void uram_main( void )
{
     unsigned long i;
     unsigned long t_count;   /* times of transmission when long word access        */
     unsigned char *h_buffer_pt_byte;
                              /* pointer of HIFRAM buffer address(byte access)      */
     unsigned long *h_buffer_pt_long;
                              /* pointer of HIFRAM buffer address(long word access) */

     //------clear in HIFRAM buffer 0 (bank0)
     h_buffer_pt_long = &HIFRAM_BUFF_LONG;

     HIFRAM_HEAD = 0;
     for(  i=0 ;  i<HIF_BUFF_SIZE_LONG ; i++ )
     {
          *h_buffer_pt_long = 0;
          h_buffer_pt_long ++;
     }

     ST_HIFSCR.BIT.BMD = 1;
                 /* SH7618 and external device access the bank where HIFRAM differ  */

     //------clear in HIFRAM buffer 1 (bank1)
     h_buffer_pt_long = &HIFRAM_BUFF_LONG;

     HIFRAM_HEAD = 0;
     for(  i=0 ;  i<HIF_BUFF_SIZE_LONG ; i++ )
     {
          *h_buffer_pt_long = 0;
          h_buffer_pt_long ++;
     }

     Sdram_address_pt_long = &SDRAM_TOP;       /* transmission destination address  */
```

```
ST_HIFGSR.BIT.HIFS = 1; /* set HIFS=1(The writing to HIFRAM is possible)       */

while(1)
{
     while(ST_HIFGSR.BIT.HIFS != 0);
                        /* wait until write end to HIFRAM from external device */

     ST_HIFSCR.BIT.BSEL = ~ST_HIFSCR.BIT.BSEL;  /* access bank is change       */

     ST_HIFGSR.BIT.HIFS = 1;
                        /* set HIFS=1(The writing to HIFRAM is possible)       */

     if( ST_HIFGSR.BIT.TEND == 1 )              /* transmission end(TEND=1)    */
     {
          break;                                /* escape from loop            */
     }
     else
     {
          //------transmit to SDRAM from HIFRAM buffer
          //------transmit by long-word size
          if( HIFRAM_HEAD == HIF_BUFF_SIZE_BYTE )
          {
               h_buffer_pt_long = &HIFRAM_BUFF_LONG;
               t_count = HIFRAM_HEAD/4;         /* times of transmission       */

               for( i=0 ; i<t_count ; i++ )
               {
                    *Sdram_address_pt_long = *h_buffer_pt_long;

                    Sdram_address_pt_long++;
                    h_buffer_pt_long++;
               }
          }
```

```
               //------transmit by byte size
               else if( HIFRAM_HEAD < HIF_BUFF_SIZE_BYTE )
               {
                    h_buffer_pt_byte = &HIFRAM_BUFF_BYTE;
                    Sdram_address_pt_byte = (unsigned char*)Sdram_address_pt_long;

                    for( i=0 ; i<HIFRAM_HEAD ; i++ )
                    {
                         *Sdram_address_pt_byte = *h_buffer_pt_byte;

                         Sdram_address_pt_byte++;
                         h_buffer_pt_byte++;
                    }
               }
               else
               {
                    while(1);                        /* error                      */
               }

               HIFRAM_HEAD = 0;                      /* clear the HIFRAM header    */
          }
     }

     //------Execution of the transmitted program
     jump_sdram();
}
```

(2) SH7618 program 2

```
;**************************************************************************
;      function : jump_sdram
;      operation      : Execution of the transmitted program
;      CUP            : SH7618
;      date           : 2004.4
;***************************************************************
             .EXPORT     _jump_sdram
;
; MS7618 CS3 area SDRAM address,user program start address
SDRAM_START: .equu        H'AC000000
             .SECTION    UM_P, CODE, ALIGN=4
;********************************************************
;
_jump_sdram:
;
     mov.l  #SDRAM_START,R10               ;Program Start
     jmp    @R10
     nop
;
     .END
```

(3) SH7641 program

```
/**************************************************************/
// SH7618 HIF boot mode application note
//     Program transmission which used two bank of HIFRAM
//          CPU      : SH7641,SH-3 DSP,Big Endian
//          Clock    : External input = 12.5MHz
//                     CPU clock = 100MHz
//                     External BUS clock = 50MHz
//                     Peripheral clock = 25MHz
//          Written  : '04/4 Rev.2.0
/**************************************************************/

#include "7641.h"

/**********************************************************/
/*     Protocol declaration of the function               */
/**********************************************************/
/*------ Symbol Definition --------------------------------------*/
#define TRANS_STRAGE_ADDR   0xA5602000   // storing address of a sdram program
#define TRANS_P_SIZE        0x200000     // sdram program size(Byte)

#define HIF_BUFF_SIZE       0x3FC        // HIFRAM buffer size(192Byte)
#define HIF_HEAD_ADDR       0x0000       // HIFRAM header address
#define HIF_BUFF_ADDR       0x0004       // HIFRAM buffer address

     //------The value when specifying the register of HIF
#define SEL_HIFMCR_LO       0x000A       // HIFMCR[15:0]
#define SEL_HIFBCR_LO       0x003E       // HIFBCR[15:0]
#define SEL_HIFADR_LO       0x0016       // HIFADR[15:0]
#define SEL_HIFDATA_UP      0x0018       // HIFDATA[31:16]
#define SEL_HIFGSR_LO       0x0002       // HIFGSR[15:0]

/*------ Function Definition -------------------------------------*/
void main(void);

unsigned short* write_HIFRAM(unsigned short* , unsigned short , unsigned short );

void sync_7618(unsigned short* , unsigned short , unsigned short , unsigned long ,
unsigned short );

/*-----------------------------------------------------------*/
     //------The address when accessing the register of HIF
// select of the register of HIF
#define HIF_REG_SEL                 (*(volatile unsigned short *)0xB4001000)
// data write to the register of HIF
#define HIF_DATA_WR                 (*(volatile unsigned short *)0xB4000000)
// data read from the HIFGSR register
#define HIF_GSR_RD                  (*(volatile unsigned short *)0xB4001004)

/**********************************************************/
/*          Main routine                                  */
/**********************************************************/
void main(void)
```

```
{
     //------set of bus interface
     BSC.CS5ABCR.BIT.BSZ = 2;
     BSC.CS5AWCR = 0x00000901;

     //------set as a CS5A function
     PFC.PCCR.BIT.PC1MD = 3;     /* bit[2-3]-PC1MD=b'11 : PC1=>CS5A                */

     //------synchronization is taken SH7618, and a program is written to HIFRAM
     sync_7618((unsigned short*)TRANS_P_STRAGE_ADDR , HIF_HEAD_ADDR ,
                                    HIF_BUFF_ADDR , TRANS_P_SIZE,HIF_BUFF_SIZE);

     //------Loop
     while(1);                   /* Loop                                           */
}

/*************************************************************************************/
//     function  : write_HIFRAM
//     operation : boot program writing to HIFRAM
//     argument  : trans_src_addr ; storing address of a boot program
//                 hif_addr       ;
                              head address of HIFRAM which transmits a boot program
//                 t_size         ; transmit program size
//     return    : trans_src_addr ; storing address of a boot program
/*************************************************************************************/
unsigned short* write_HIFRAM(unsigned short *trans_src_addr , unsigned short hif_addr ,
unsigned short t_size)
{
     time = t_size/2 + t_size%2;       /* calculate times of transmission          */

     HIF_REG_SEL = SEL_HIFADR_LO;      /* select HIFADR register                   */
     HIF_DATA_WR = hif_addr;           /* set of HIFRAM address                    */

     HIF_REG_SEL = SEL_HIFDATA_UP;     /* select HIFDATA register[31:16]           */
     HIF_DATA_WR = (*trans_src_addr);  /* set to HIFDATA register[31:16]           */

     trans_src_addr ++;                /* storing address is increment(+2)         */

     HIF_DATA_WR = (*trans_src_addr);  /* set to HIFDATA register[15:0]            */

     trans_src_addr ++;                /* storing address is increment(+2)         */

     HIF_REG_SEL = SEL_HIFMCR_LO;      /* select HIFMCR register[15:0]             */
     HIF_DATA_WR = 0x00A0;             /* set as continuation write mode           */

     HIF_REG_SEL = SEL_HIFDATA_UP;     /* select HIFDATA register[31:16]           */

     for( i=2 ; i<time ; i++){
          HIF_DATA_WR = (*trans_src_addr); /* write to HIFDATA register(HIFRAM)    */

          trans_src_addr ++;               /* storing address is increment(+2)     */
     }

     return(trans_src_addr);
```

```
}

/*************************************************************************************/
//     function  : sync_7618
//     operation : synchronization is taken SH7618, and a program is written to HIFRAM
//     argument  : *s_addr ; pointer of source address
//                  h_addr ; HIFRAM header address
//                  b_addr ; HIFRAM buffer address
//                  t_size ; transmission data size
//                  b_size ; HIFRAM buffer size
//     argument  : non-argument
/*************************************************************************************/
void sync_7618(unsigned short* s_addr , unsigned short h_addr , unsigned short b_addr ,
unsigned long t_size , unsigned short b_size)
{
     volatile unsigned short status;

     while(1){
          status = HIF_GSR_RD;          /* read from HIFGSR register[15:0]          */

          while(status != 0x0001){      /* wait until HIFGSR.HIFS=1                 */
               status = HIF_GSR_RD;     /* read from HIFGSR register[15:0]          */
          }

     //------preparation write to header
          HIF_REG_SEL = SEL_HIFADR_LO;  /* select HIFADR register[15:0]             */
          HIF_DATA_WR = h_addr;         /* set of HIFRAM header address             */
          HIF_REG_SEL = SEL_HIFDATA_UP; /* select HIFDATA register[31:16]           */
          HIF_DATA_WR = 0;              /* set to HIFDATA register[31:16]           */

     //------usual transmission
          if(t_size > b_size)
          {
     //------transmission data size is written to header
               HIF_DATA_WR = b_size;    /* write to HIFDATA register[15:0]          */

     //------write to HIFRAM buffer
               s_addr = write_HIFRAM(s_addr,b_addr,b_size);

     //------transmission end process
               HIF_REG_SEL = SEL_HIFGSR_LO;  /* select HIFGSR register[15:0]        */
               HIF_DATA_WR = 0x0000;         /* set to HIFGSR register[15:0]        */
                                                       // TEND=0 , HIFS=0
          }

     //------last transmission
          else
          {
     //------transmission data size is written to header
               HIF_DATA_WR = (unsigned short)t_size;
                                             /* write to HIFDATA register[15:0]     */

     //------write to HIFRAM buffer
               s_addr = write_HIFRAM(s_addr,b_addr,b_size);
     //------transmission end process
               HIF_REG_SEL = SEL_HIFGSR_LO;  /* select HIFGSR register[15:0]        */
               HIF_DATA_WR = 0x0000;         /* set to HIFGSR register[15:0]        */
                                                       // TEND=0 , HIFS=0

               break;                        /* escape from loop                    */
          }
     //------calculate transmission data size
          t_size = t_size - b_size;
     }

     status = HIF_GSR_RD;                    /* read from HIFGSR register[15:0]     */

     while( status==0x0000 )                 /* wait until HIFGSR.HIFS=1            */
     {
          status = HIF_GSR_RD;               /* read from HIFGSR register[15:0]     */
     }

     HIF_DATA_WR = 0x0002;                   /* set to HIFGSR register[15:0]        */
                                                       // TEND=1 , HIFS=0
}
```

Revision Record

		Description	
Rev.	Date	Page	Summary
1.00	Sep.05.05	—	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.