# V850E2/ML4

## CSIH Control

R01AN1222EJ0100
Rev.1.00
Jun. 22, 2012

## Abstract

This document describes how to set up the CSIH (clocked three-wire serial interface) and also gives an outline of the operation and describes the procedures for using a sample program for the V850E2/ML4.

The features of the operation are described below, including usages of the CSIH in following 4 modes:

- Master Direct-Access Transmit-Only Mode
- Slave Direct-Access Receive-Only Mode
- Master Dual-Buffer Transmit-Only Mode
- Slave Dual-Buffer Receive-Only Mode

## Products

V850E2/ML4

## Integrated development environments

CubeSuite+, GHS MULTI V5.1.7D, and IAR for V850 Kickstart V3.80.

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

## Contents

## 1. Specifications

This application note explains the following four operation modes of the CSIH as usage examples:

- Master dual-buffer transmit-only mode
- Slave dual-buffer receive-only mode
- Master direct-access transmit-only mode
- Slave direct-access receive-only mode

In master mode, the serial communication clock is generated by the internal baud rate generator (BRG) and supplied by signal CSIHnTSCK. In slave mode, another device is the communication master. The communication clock is supplied.

See section “Flow Charts” for the details of the sample program.

The basic specifications of the communication are shown below.

| Memory mode | Direct access mode | Dual buffer mode |
|---|---|---|
| Receive I/F | CSIH1 | |
| Transmit I/F | CSIH0 | |
| Transfer direction | MSB first | |
| Parity bits during transmission/reception | No parity bit | |
| Data length | 8bits | |
| Baud rate | 64kbps | |
| FIFO size | None | 64 bytes (each transmit / receive FIFO) |
| Data size | 6 bytes | 9bytes |
| JOB(job number) | 2JOB | None |
| EDL（extended data length） | None | |
| LBM（loop back mode） | None | |
| SS（slave select） | None | |

Table 1.1 lists the Peripheral Functions and their Applications and Figure 1.1 shows the Usage Example.

**Table 1.1 Peripheral Functions and their Applications**

| Peripheral Function | Application |
|---|---|
| Ports(P1_4, P1_5, P4_3, P4_4) | Connected to LEDs, and light on or off LEDs. |
| CSIH0 - SO0F | CSIH0 output (JA6-9) |
| CSIH0 - SI0F | CSIH0 input (JA6-12) |
| CSIH0 - SCK0F | CSIH0 clock(output) (JA6-11) |
| CSIH1 – SO1F | CSIH1 output (JA2-6) |
| CSIH1 – SI1F | CSIH1 input (JA2-8) |
| CSIH1 – SCK1F | CSIH1 clock (input) (JA2-10) |

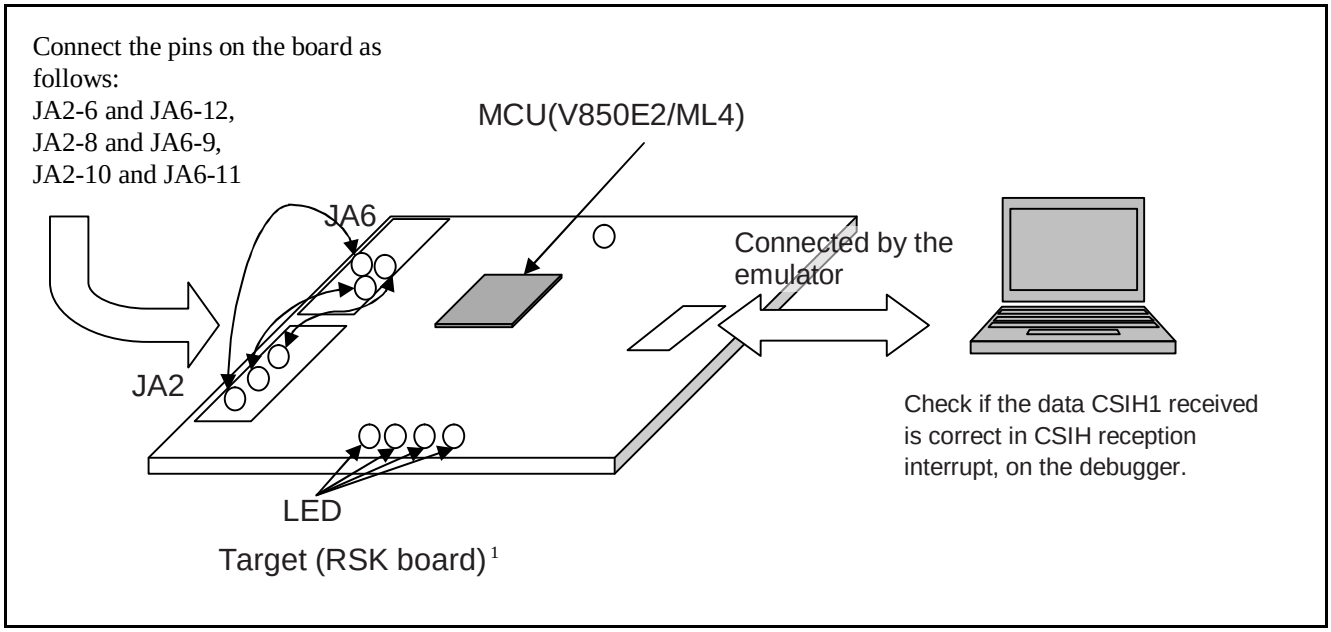


**Figure 1.1 Usage Example[1]**

[1] Mass production of RSK board will start in August, 2012.

## 2. Operation Confirmation Conditions

The sample code accompanying this application note has been run and confirmed under the conditions below.

**Table 2.1 Operation Confirmation Conditions**

| Item | Contents |
|---|---|
| MCU used | V850E2/ML4 |
| Operating frequency | 200MHz (PLL multiplies the oscillator input frequency (fx: 10MHz) by 20.) |
| Operating voltage | 3.3V |
| Integrated development environment | CubeSuite+ V1.00 |
| | GHS MULTI V5.1.7D |
| | IAR for V850 Kickstart V3.80.1 |
| C compiler | CX V1.20(CubeSuite+), optimization: default |
| | C-V850E 5.1.7 RELEASE(GHS MULTI) , optimization: default |
| | IAR C/C++ Compiler for V850 3.80.1 [Kickstart] (3.80.1.30078), optimization: default |
| Operating mode | Normal operation mode |
| Sample code version | V1.00 |
| Board used | RSK board |
| Device used | E1 emulator or MINICUBE |
| Tool used | none |

# 3. Hardware

## 3.1 Hardware Configuration

Figure 3.1 shows a Connection Example.

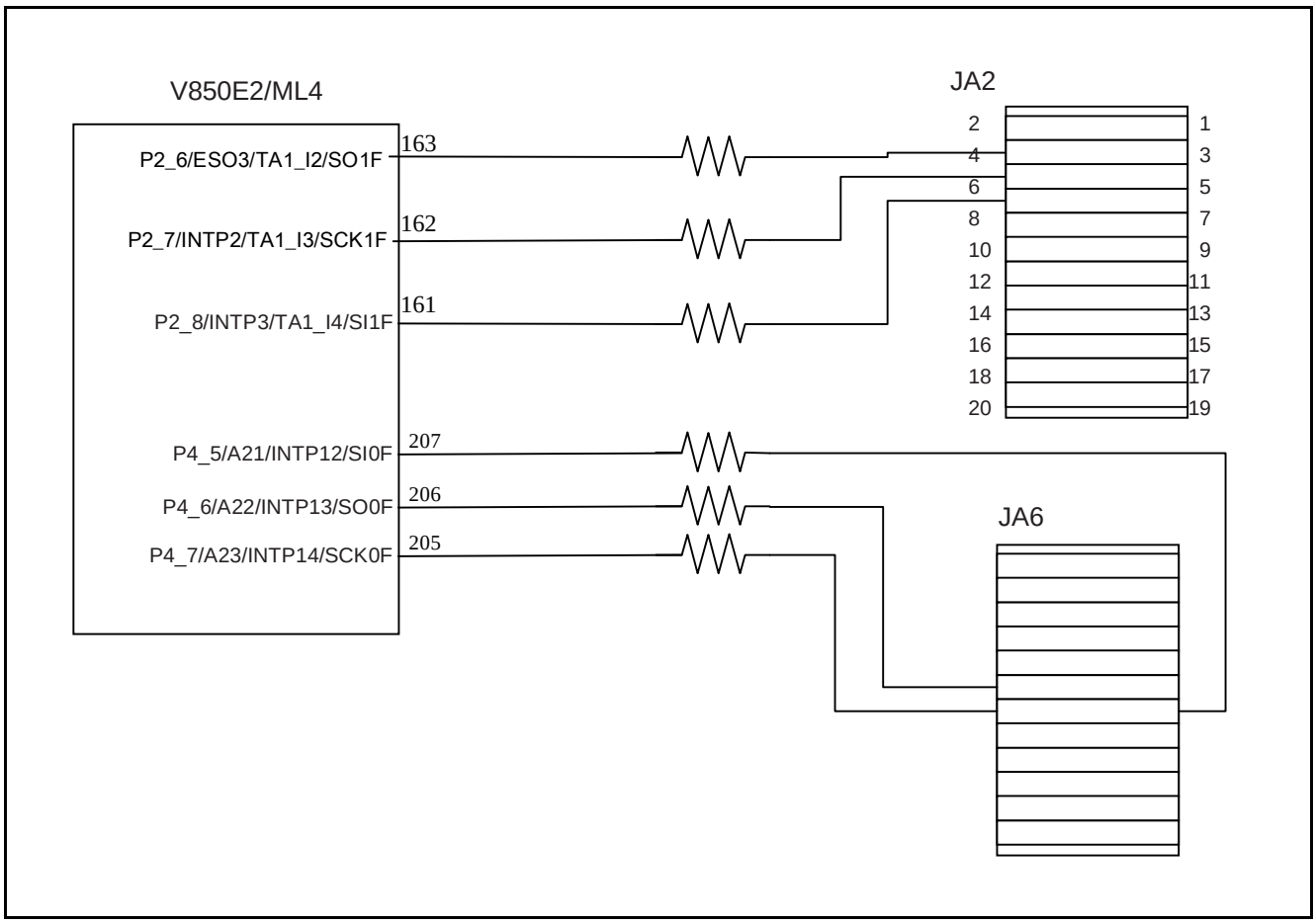


**Figure 3.1 Connection Example on RSK board**

## 3.2 Pin(s) Used

Table 3.1 lists the Pins Used and Its Function.

**Table 3.1 Pins Used and Its Functions**

| Pin Name | I/O | Function |
|---|---|---|
| PORT P1_4 | output | Port mode, output, LED0 |
| PORT P1_5 | output | Port mode, output, LED1 |
| PORT P4_3 | output | Port mode, output, LED2 |
| PORT P4_4 | output | Port mode, output, LED3 |
| SO0F | output | Serial data output for transmission of CSHI0 |
| SI0F | input | Serial data output for reception of CSHI0 (not used.) |
| SCK0F | output | Serial clock output of CSIH0 (output as a master) |
| SO1F | output | Serial data output for transmission of CSHI1 (not used.) |
| SI1F | input | Serial data output for reception of CSHI1 |
| SCK1F | input | Serial clock output of CSIH1 (input as a slave) |

# 4. Software

## 4.1 Operation Overview

Operation Overview is described in the following figure. The main() function initializes each functions, and waits interrupt. When reception interrupt has occurred, the sample program stores the received data to the users' memory, and restarts CSIH.

Figure 4.1 shows the Sequence Diagram.

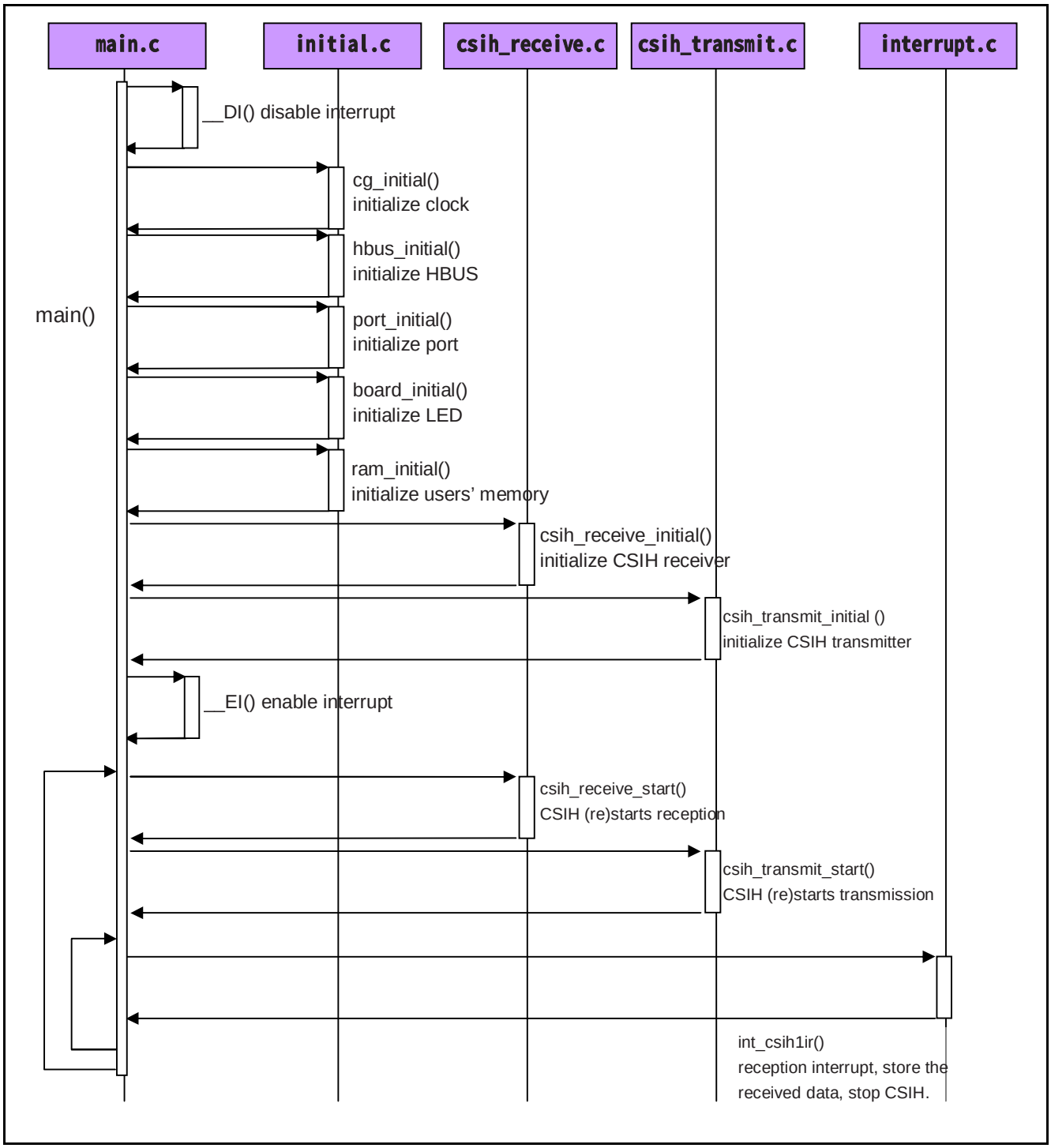


**Figure 4.1 Sequence Diagram**

## 4.2 Required Memory Size

Table 4.1 lists the Required Memory Size. (CubeSuite+, optimization=default)

**Table 4.1 Required Memory Size**

| Memory Used | Size | Remarks |
|---|---|---|
| ROM | 6696 | Shown as ROM area size in map file |
| RAM | 4204 | Shown as RAM area size in map file |
| Maximum user stack usage | 12 | CubeSuite+ stack estimation tool calculated. |
| Maximum interrupt stack usage | 204 | The same as above. |

Note: The required memory size varies depending on the C compiler version and its options.

## 4.3 File Composition

Table 4.2 lists the File(s) Used in the Sample Code. Files not generated by the integrated development environment should not be listed in this table.

**Table 4.2 File(s) Used in the Sample Code**

| File Name | Outline | Remarks |
|---|---|---|
| crtE.s | Initialize hardware | Only in the project for CubeSuite+ |
| startup.s | | Only in the project for GHS MULTI |
| V850E2ML4.dir | Linker directive file | Only in the project for CubeSuite+ |
| V850E2 ML4 CSIH.ld | | Only in the project for GHS MULTI |
| vector.s | Vector table | Only in the project for GHS MULTI |
| csih.h | Declare variables and functions. | |
| df4022_800.h | Declare register macros for V850E2/ML4 | Only in the project for GHS MULTI |
| V850E2ML4PortRegister.h | Header file for portconfig.c | |
| main.c | Main routine | |
| initial.c | Initialize software | |
| csih_transmit.c | CSIH transmitter | |
| csih_receive.c | CSIH receiver | |
| interrupt.c | Interrupt routines | |
| portconfig.c | Initialize port | |

## 4.4 Option-Setting Memory

This sample does not specify any option-bytes. Specify them if necessary.

## 4.5 Variable(s)

Table 4.3 lists the Global Variable(s).

**Table 4.3 Global Variable(s)**

| Type | Variable Name | Contents | Function Used |
|---|---|---|---|
| unsigned short | adc_result[10] | Buffer for A/D converted data | __interrupt void int_adca0i0(void);<br>void ram_initial(void); |
| unsigned char | flag_mode; | Indicates transmission mode | csih_receive_initial()<br>csih_receive_start()<br>csih_transmit_initial()<br>csih_transmit_start()<br>int_csih0ic()<br>int_csih0ijc()<br>int_csih1ir()<br>main() |
| unsigned int | flag_error; | CSIH error flag | main()<br>ram_initial()<br>display()<br>int_csih0ire()<br>int_csih1ire() |
| unsigned char | count; | Counter of the occurrence of CSIH1IR. | ram_initial()<br>int_csih1ire()<br>int_csih1ir() |
| unsigned char | flag_transmit_over; | A flag indicates whether the transmission is over. | main()<br>int_csih0ic()<br>int_csih0ijc()<br>ram_initial()<br>display() |
| unsigned char | flag_receive_over; | A flag indicates whether the reception is over. | ram_initial()<br>int_csih1ir()<br>csih_receive_start()<br>display() |
| unsigned char | flag_job_transmit; | A flag indicates whether the transmission is over. | csih_transmit_1_start()<br>ram_initial()<br>int_csih1ir() |
| unsigned int | buf_receive[NUM]; | Buffer for received data. | ram_initial()<br>clear_receive_buffer()<br>int_csih0ire() |
| unsigned int * | point_receive; | Pointer for received buffer. | ram_initial()<br>clear_receive_buffer()<br>int_csih0ire()<br>int_csih1ire()<br>int_csih1ir() |
| unsigned int * | point_transmit; | Pointer for transmission buffer. | ram_initial()<br>int_csih0ire()<br>int_csih1ire() |
| unsigned int | buf_transmit[NUM]; | Buffer for data to be transmitted. | csih_transmit_1_start()<br>csih_transmit_2_start() |

## 4.6 Function(s)

Table 4.4 lists the Function(s).

**Table 4.4 Function(s)**

| Function Name | Outline |
|---|---|
| void port_initial(void) | Sets up ports and their mode. |
| void PortConfigulation1(void) | Sets up port group 1. |
| void PortConfigulation2(void) | Sets up port group 2. |
| void PortConfigulation4(void) | Sets up port group 4. |
| void cg_initial(void) | Initializes the special clock frequency control register. |
| void hbus_initial(void) | Initializes the AHB bus |
| void board_initial(void) | Initializes the LEDs |
| void ram_initial(void) | Sets up the initial state of the user RAM |
| void display(void) | Displays LED whether there are any transmission or reception errors. |
| void wait(int number) | Waits some time. |
| void clear_receive_buffer(void) | Clears reception buffer |
| void main(void) | Calls necessary initialization functions and (re)starts transmission in infinite loop. |
| void csih_receive_initial(void) | Initializes CSIH receiver. |
| void csih_transmit_initial(void) | Initializes CSIH transmitter. |
| void csih_transmit_start(void) | Starts CSIH transmission. |
| void csih_receive_start(void) | Starts CSIH reception. |
| interrupt void int_csih0ire(void) | Interrupt CSIH0 error reception. |
| interrupt void int_csih0ic(void) | Interrupt CSIH0 transmission over. |
| interrupt void int_csih0ijc(void) | Interrupt CSIH0 transmission job over. |
| interrupt void int_csih1ire(void) | Interrupt CSIH1 error reception. |
| interrupt void int_csih1ir(void) | Interrupt CSIH1 reception. |
| void csih_receive_initial(void) | Initializes CSIH receiver. |
| void csih_receive_1_initial(void) | Initializes CSIHn as a receiver in direct access mode. |
| void csih_receive_2_initial(void) | Initializes CSIHn as a receiver in dual buffer mode. |
| void csih_transmit_initial(void) | Initializes CSIH transmitter. |
| void csih_transmit_1_initial(void) | Initializes CSIHn as a transmitter in direct access mode. |
| void csih_transmit_2_initial(void) | Initializes CSIHn as a transmitter in dual buffer mode. |
| void csih_transmit_1_start(void) | Starts CSIHn transmission in direct access mode. |
| void csih_transmit_2_start(void) | Starts CSIHn transmission in dual buffer mode. |

## 4.7 Function Specification(s)

The following table(s) list(s) the sample code function specification(s).

| main() | |
|---|---|
| **Outline** | Main routine |
| **Header** | — |
| **Declaration** | void main(void) |
| **Description** | Calls initializing functions, enters infinite loop, (re)starts transmission, and waits CSIH interrupt. |
| **Arguments** | none - |
| **Return Value** | none |

| port_initial() | |
|---|---|
| **Outline** | Sets up ports and their mode. |
| **Header** | csih.h |
| **Declaration** | void port_initial (void) |
| **Description** | Sets up ports in port-mode, output, for controlling LEDs. |
| **Arguments** | none - |
| **Return Value** | none |

| PortConfigulation0() ～PortConfigulation8() | |
|---|---|
| **Outline** | Sets up ports and their mode. |
| **Header** | V850E2ML4PortRegister.h |
| **Declaration** | void PortConfigulation0(void)…void PortConfigulation8(void) |
| **Description** | Called in port_initial(), and sets up port group N (0..8). Actually PortConfigulation1(),PortConfigulation2(), and PortConfigulation4() are called. |
| **Arguments** | none - |
| **Return Value** | none |

| cg_initial() | |
|---|---|
| **Outline** | Initialize clock |
| **Header** | csih.h |
| **Declaration** | void cg_initial(void) |
| **Description** | Initializes the special clock frequency control register. |
| **Arguments** | none - |
| **Return Value** | none |

| hbus_initial() | |
|---|---|
| **Outline** | Initialize H-bus |
| **Header** | csih.h |
| **Declaration** | void hbus_initial(void) |
| **Description** | Initializes AHB-bus |
| **Arguments** | none - |
| **Return Value** | none |

| board_initial() | | |
|---|---|---|
| **Outline** | Initialize board | |
| **Header** | csih.h | |
| **Declaration** | void board_initial(void) | |
| **Description** | Initialize LED on the board. | |
| **Arguments** | — | — |
| **Return Value** | — | |

| ram_initial() | | |
|---|---|---|
| **Outline** | Initialize users' memory. | |
| **Header** | csih.h | |
| **Declaration** | void ram_initial(void) | |
| **Description** | Initialize LED on the board. | |
| **Arguments** | — | — |
| **Return Value** | — | |

| display() | | |
|---|---|---|
| **Outline** | Displays error in LED. | |
| **Header** | csih.h | |
| **Declaration** | void display(void) | |
| **Description** | Displays LED whether there are any transmission or reception errors. | |
| **Arguments** | — | — |
| **Return Value** | — | |

| wait() | | |
|---|---|---|
| **Outline** | Waits specified time. | |
| **Header** | csih.h | |
| **Declaration** | void wait (void) | |
| **Description** | Waits some time for CSIH transmission. | |
| **Arguments** | int number | Waiting time. |
| **Return Value** | — | |

| clear_receive_buffer() | | |
|---|---|---|
| **Outline** | Clears receive buffer. | |
| **Header** | csih.h | |
| **Declaration** | void clear_receive_buffer(void) | |
| **Description** | Clear receive buffer to 0, before CSIH reception. | |
| **Arguments** | — | — |
| **Return Value** | — | |

| csih_receive_initial() | | |
|---|---|---|
| **Outline** | Initializes CSIH receiver. | |
| **Header** | csih.h | |
| **Declaration** | void csih_receive_initial(void) | |
| **Description** | Initialize CSIH as a receiver, calls subroutine according to "flag_mode". | |
| **Arguments** | — | — |
| **Return Value** | — | |

csih_receive_1_initial()

| | |
|---|---|
| **Outline** | Initializes CSIH receiver. |
| **Header** | csih.h |
| **Declaration** | void csih_receive_1_initial(void) |
| **Description** | Initializes CSIHn as a receiver in direct access mode. |
| **Arguments** | — — |
| **Return Value** | — |

csih_receive_2_initial()

| | |
|---|---|
| **Outline** | Initializes CSIH receiver. |
| **Header** | csih.h |
| **Declaration** | void csih_receive_2_initial(void) |
| **Description** | Initializes CSIHn as a receiver in dual buffer mode. |
| **Arguments** | — — |
| **Return Value** | — |

csih_transmit_initial()

| | |
|---|---|
| **Outline** | Initializes CSIH transmitter. |
| **Header** | csih.h |
| **Declaration** | void csih_transmit_initial(void) |
| **Description** | Initialize CSIH as a transmitter, calls subroutine according to “flag_mode”. |
| **Arguments** | — — |
| **Return Value** | — |

csih_transmit_1_initial()

| | |
|---|---|
| **Outline** | Initializes CSIH transmitter. |
| **Header** | csih.h |
| **Declaration** | void csih_transmit_1_initial(void) |
| **Description** | Initializes CSIHn as a transmitter in direct access mode. |
| **Arguments** | — — |
| **Return Value** | — |

csih_transmit_2_initial()

| | |
|---|---|
| **Outline** | Initializes CSIH transmitter. |
| **Header** | csih.h |
| **Declaration** | void csih_transmit_2_initial(void) |
| **Description** | Initializes CSIHn as a transmitter in dual buffer mode. |
| **Arguments** | — — |
| **Return Value** | — |

csih_transmit_1_start ()

| | |
|---|---|
| **Outline** | Starts CSIH transmission. |
| **Header** | csih.h |
| **Declaration** | void csih_transmit_1_start (void) |
| **Description** | Starts CSIHn transmission in direct access mode. |
| **Arguments** | — — |
| **Return Value** | — |

csih_transmit_2_start ()

| | |
|---|---|
| **Outline** | Starts CSIH transmission. |
| **Header** | csih.h |
| **Declaration** | void csih_transmit_2_start (void) |
| **Description** | Starts CSIHn transmission in dual buffer mode. |
| **Arguments** | — — |
| **Return Value** | — |

int_csih0ire()

| | |
|---|---|
| **Outline** | CSIH0 error interrupt. |
| **Header** | — |
| **Declaration** | __interrupt void int_ csih0ire(void) |
| **Description** | Interrupt CSIH0 error reception. Restart CSIH0. |
| **Arguments** | — — |
| **Return Value** | — |

int_csih0ic()

| | |
|---|---|
| **Outline** | CSIH0 transmission interrupt. |
| **Header** | — |
| **Declaration** | __interrupt void int_csih0ic(void) |
| **Description** | Interrupt CSIH0 transmission over. |
| **Arguments** | — — |
| **Return Value** | — |

int_csih0ijc()

| | |
|---|---|
| **Outline** | CSIH0 transmission job interrupt. |
| **Header** | — |
| **Declaration** | __interrupt void int_csih0ijc(void) |
| **Description** | Interrupt CSIH0 transmission job over. |
| **Arguments** | — — |
| **Return Value** | — |

int_csih1ire()

| | |
|---|---|
| **Outline** | CSIH1 error interrupt. |
| **Header** | — |
| **Declaration** | __interrupt void int_csih1ire(void) |
| **Description** | Interrupt CSIH1 error reception. Restarts CSIH1. |
| **Arguments** | — — |
| **Return Value** | — |

int_csih1ir()

| | |
|---|---|
| **Outline** | CSIH1 reception interrupt. |
| **Header** | — |
| **Declaration** | __interrupt void int_csih1ir(void) |
| **Description** | Interrupt CSIH1 data reception. Stores received data to users' memory. |
| **Arguments** | — — |
| **Return Value** | — |

## 4.8 Flowchart(s)

### 4.8.1 Main Processing

Figure 4.1 shows the Main Processing.

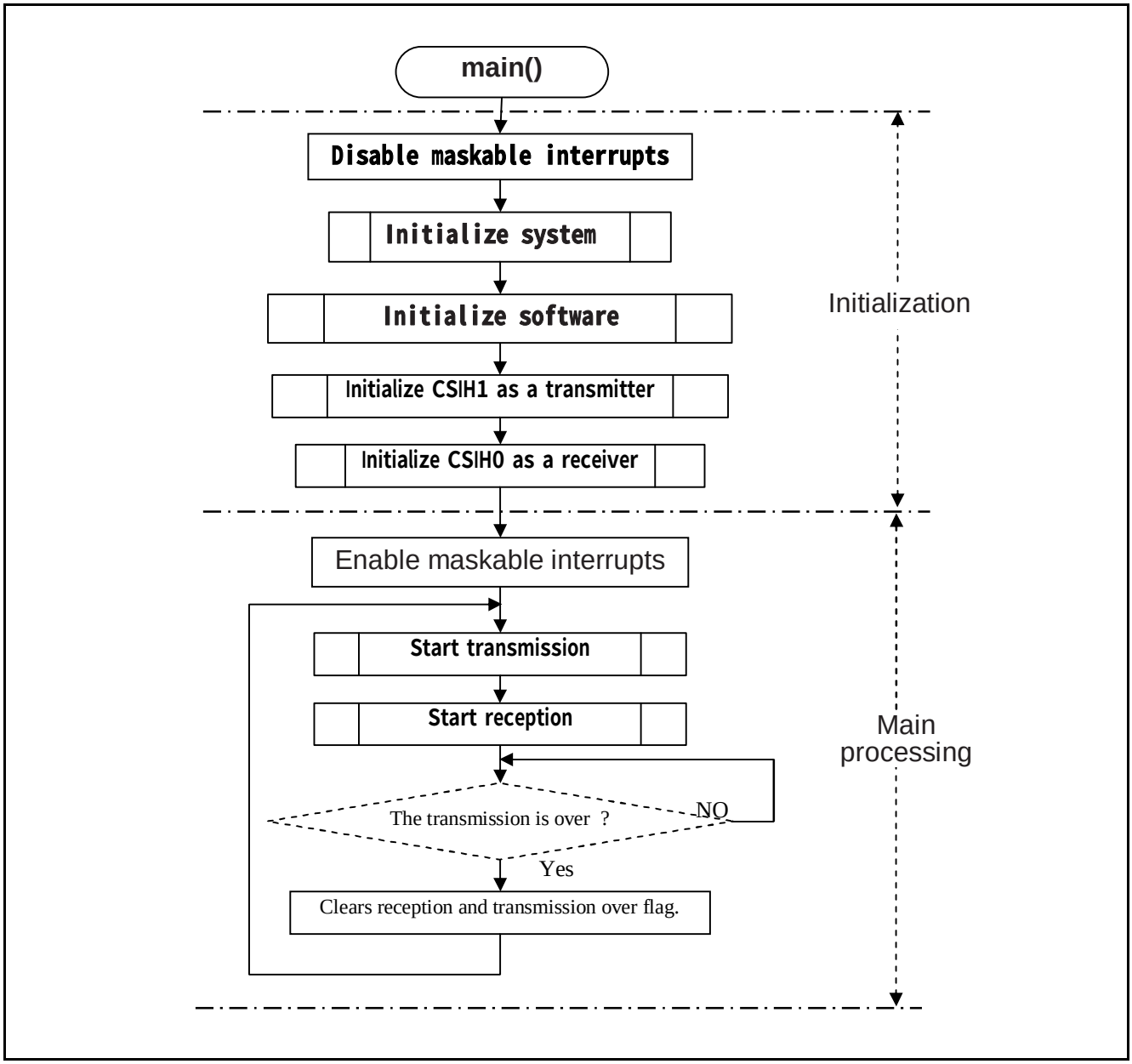


**Figure 4.1 Main Processing**

### 4.8.2 Master Direct-Access Transmit-Only Mode

Master direct-access transmit-only mode is started by writing transmit data to the CSIHnTX0W register. Forty-eight bits (six bytes) of transmit data are divided into two jobs before transmission. The communication data length is eight bits.

Figure 4.2 shows the flowchart

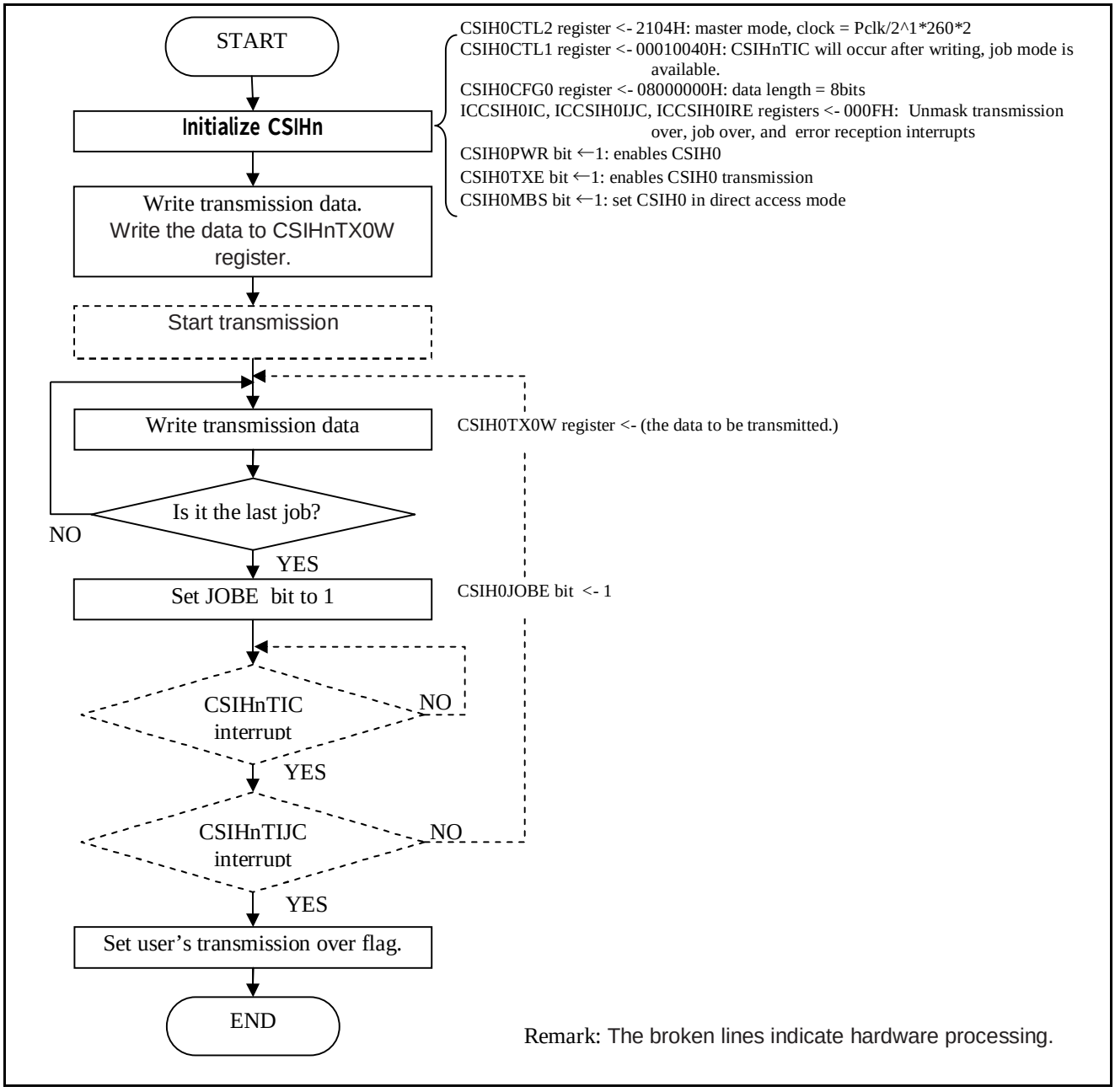


Figure 4.2 flowchart of Master Direct-Access Transmit-Only Mode

### 4.8.3 Slave Direct-Access Receive-Only Mode

Master direct-access transmit-only mode is started by writing transmit data to the CSIHnTX0W register. Forty-eight bits (six bytes) of transmit data are divided into two jobs before transmission. The communication data length is eight bits.

Figure 4.3 shows the flowchart

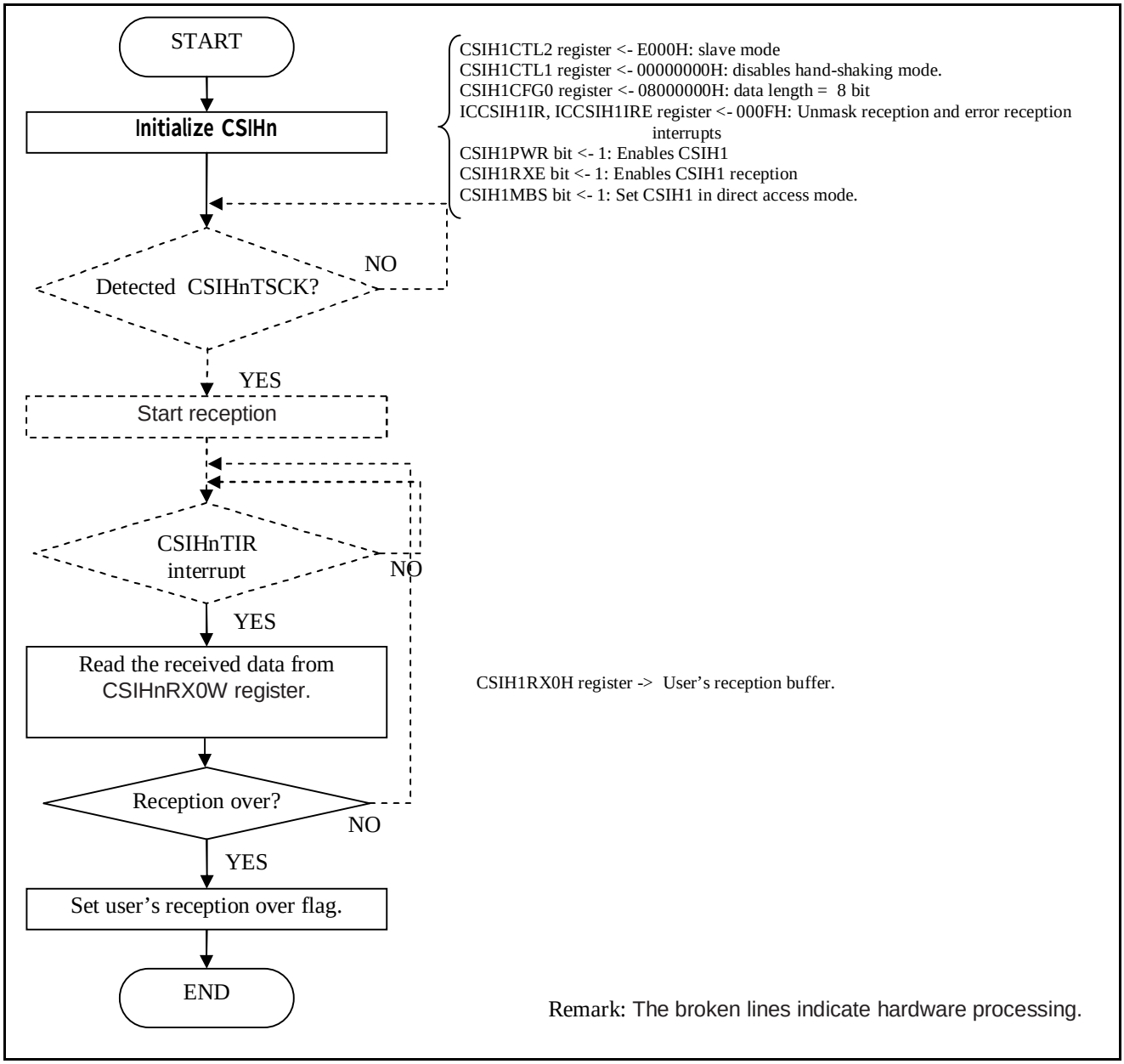


Figure 4.3 flowchart of Slave Direct-Access Receive-Only Mode

### 4.8.4 Master Dual-Buffer Transmit-Only Mode

Master dual-buffer transmit-only mode is started by setting the CSIHnMCTL2.CSIHnBTST bit to 1.
Figure 4.4 shows the flowchart.

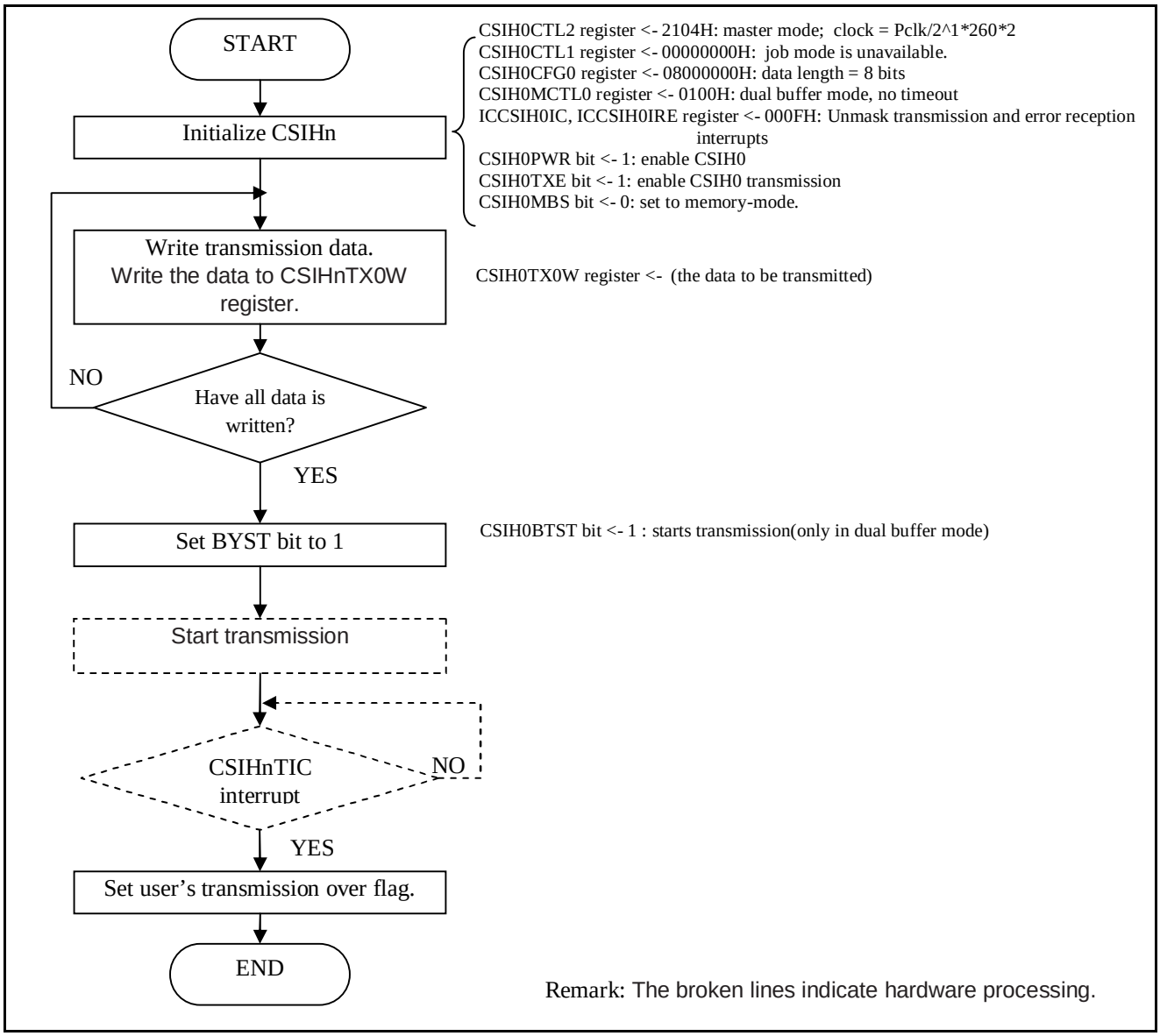


Figure 4.4 flowchart of Master Dual-Buffer Transmit-Only Mode

### 4.8.5 Slave Dual-Buffer Receive-Only Mode

Slave dual-buffer receive-only mode is started by setting the CSIHnMCTL2.CSIHnBTST bit to 1 and then detecting external clock CSIHTSCK.
Figure 4.5 shows the flowchart.

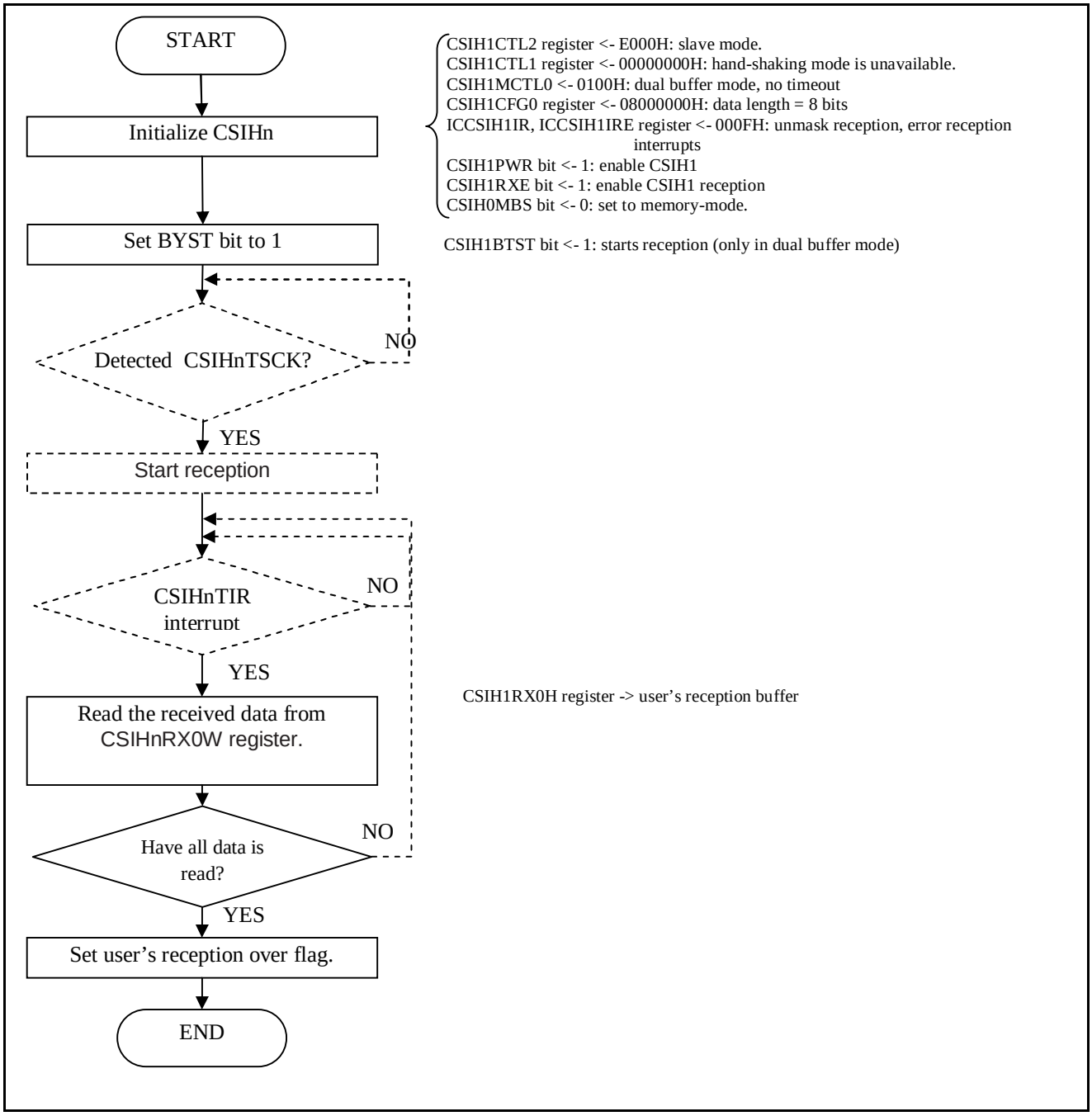


Figure 4.5 flowchart of Slave Dual-Buffer Receive-Only Mode

### 4.8.6 Interrupt Processing

Figure 4.6 shows the flowchart of Communication Error Interrupt Processing.

If a communication error occurs, a communication error interrupt is generated. Then, the communication error interrupt processing is executed. The communication is stopped and the SFR error flag is cleared. The CSIHn is reset at the same time.

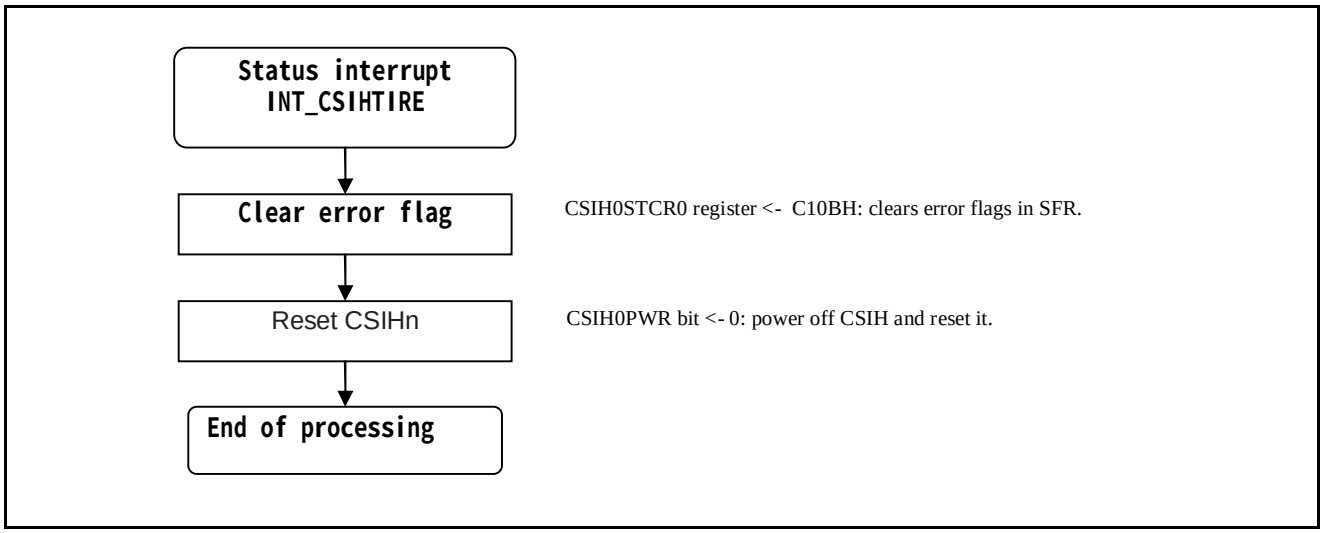


Figure 4.6 Communication Error Interrupt Processing

Figure 4.7 shows the flowchart of CSIH0 transmission Processing.

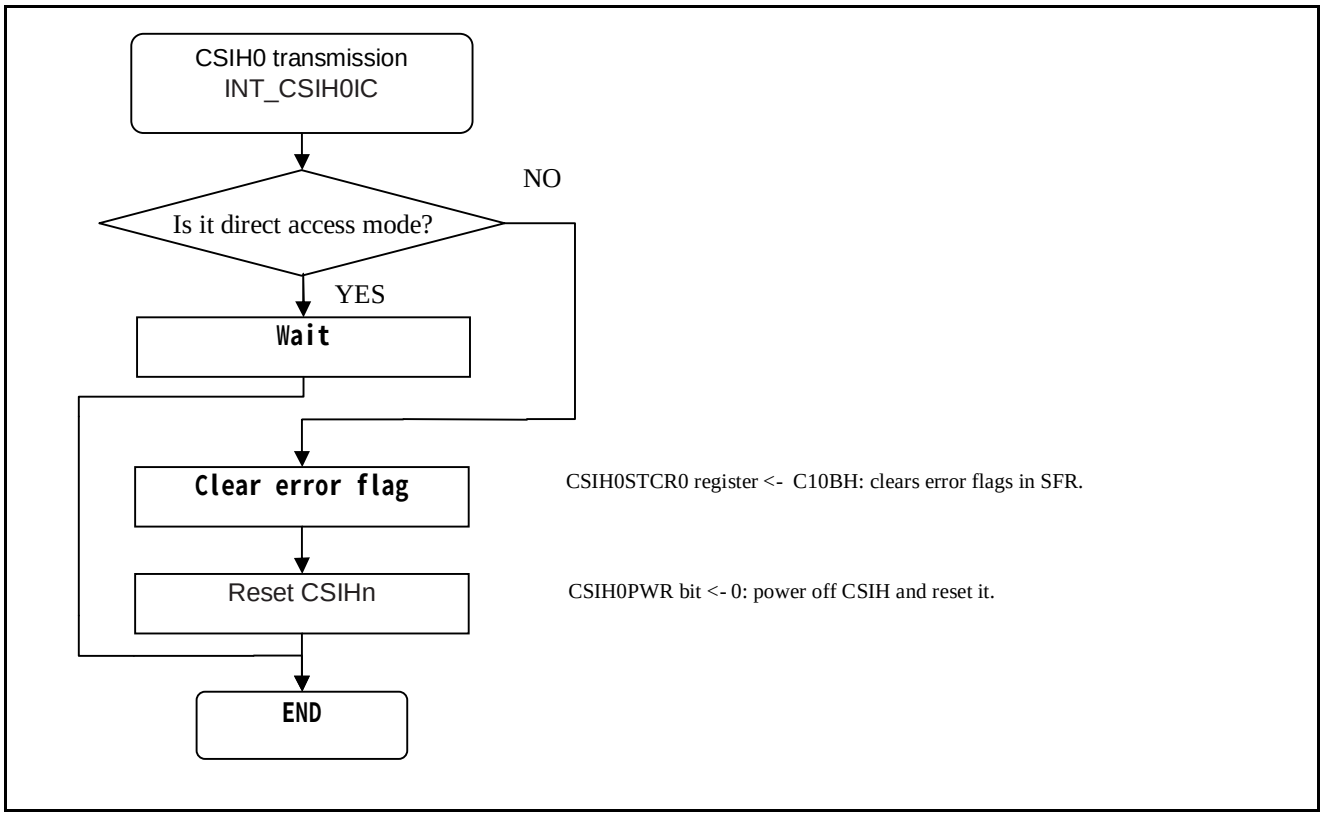


Figure 4.7 CSIH0 transmission Processing.

Figure 4.8 shows the flowchart of End of CSIH0 job Processing in direct-access mode.

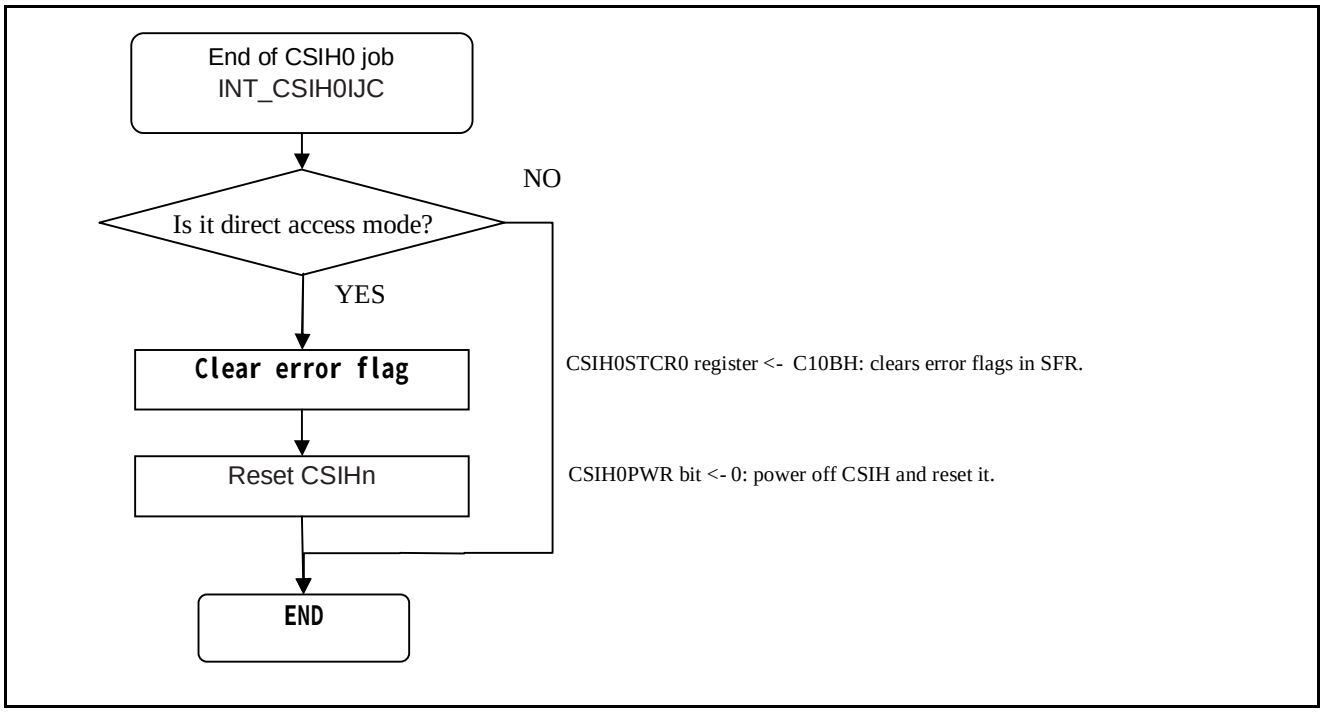


Figure 4.8 End of CSIH0 job Processing

Figure 4.9 shows the flowchart of CSIH1 reception error Processing in direct-access mode.

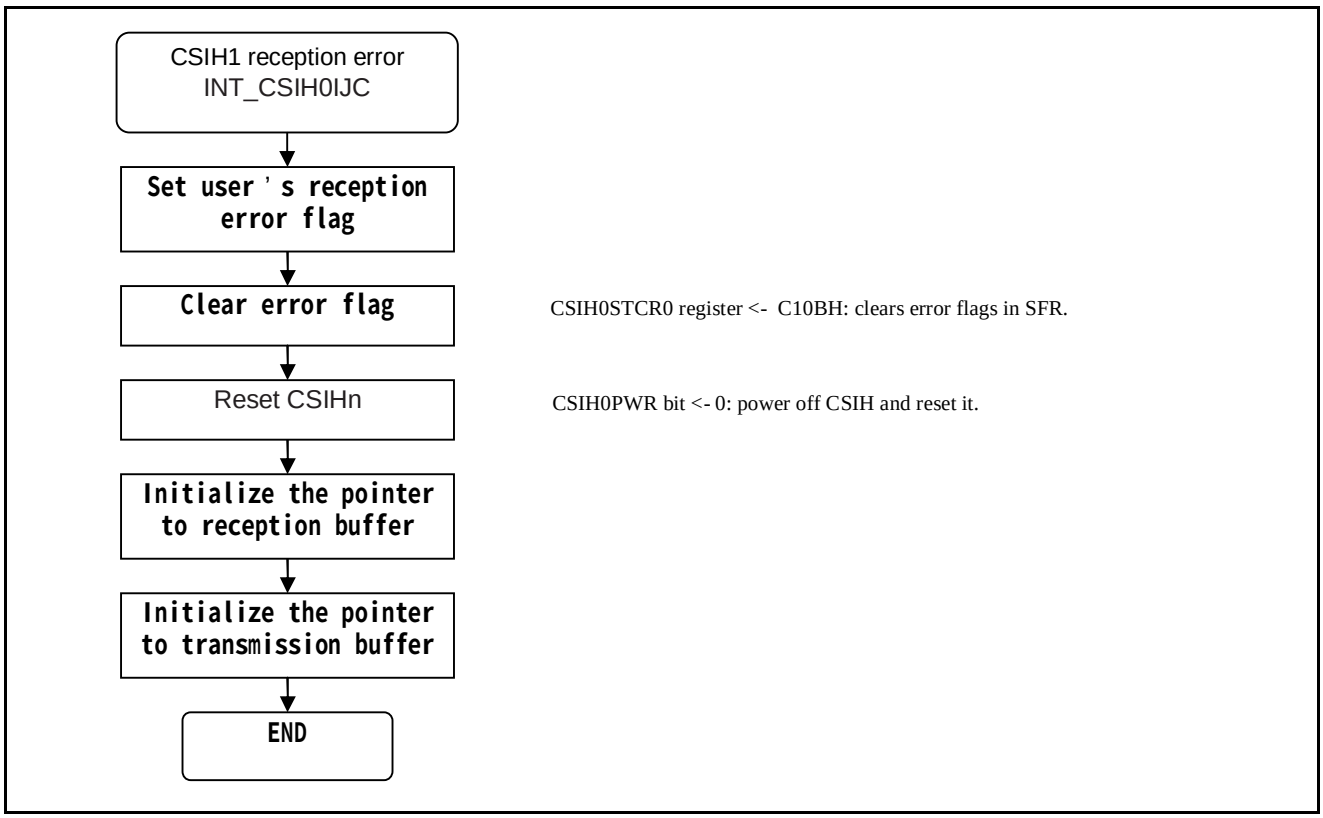


Figure 4.9 CSIH1 reception error Processing

Figure 4.10 shows the flowchart of CSIH1 reception processing in direct-access mode.

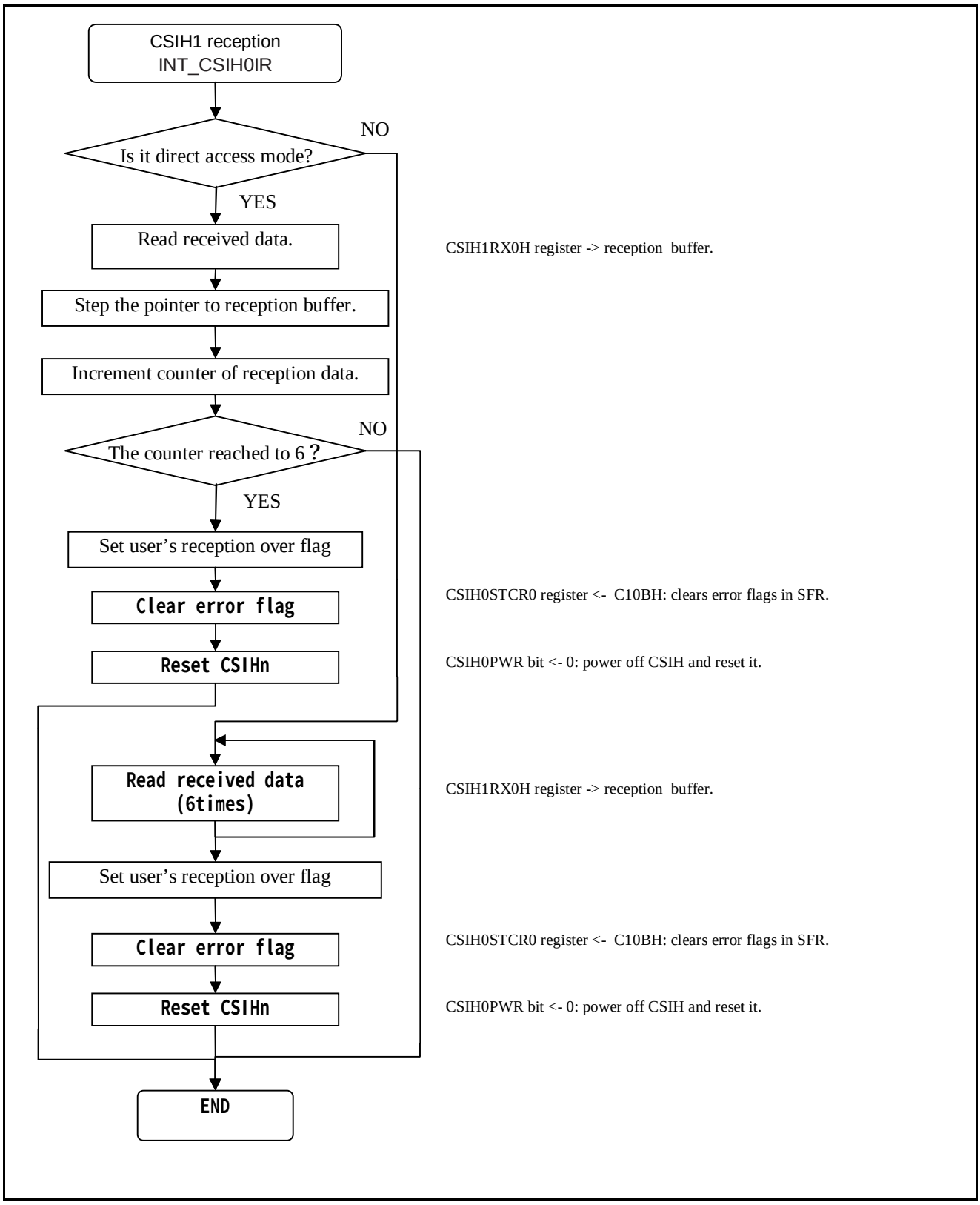


Figure 4.10 CSIH1 reception Processing

## 5. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

## 6. Reference Documents

User's Manual: Hardware

V850E2/ML4 User's Manual: Hardware (R01UH0262EJ)

The latest version can be downloaded from the Renesas Electronics website.

Technical Update/Technical News

The latest information can be downloaded from the Renesas Electronics website.

## Website and Support

Renesas Electronics website

http://www.renesas.com

Inquiries

http://www.renesas.com/contact/

| REVISION HISTORY | V850E2/ML4 Application Note A/D converter |
|---|---|

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.00 | Jun. 22, 2012 | — | First edition issued |
| | | | |

## General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins
   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.
   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on
   The state of the product is undefined at the moment when power is supplied.
   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.
5. Differences between Products
   Before changing from one product to another, i.e. to one with a different type number, confirm that the change will not lead to problems.
   The characteristics of MPU/MCU in the same group but having different type numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different type numbers, implement a system-evaluation test for each of the products.

# Notice

1. 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.
2. 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.
3. 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.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "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 etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). 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 for which it is not intended. 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 which the product is not intended by Renesas Electronics.
6. 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.
7. 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 systems manufactured by you.
8. 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.
9. 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. You should not use Renesas Electronics products or 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. When exporting the Renesas Electronics 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.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
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.

RENESAS

SALES OFFICES — Renesas Electronics Corporation — http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.


**Renesas Electronics America Inc.**
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

**Renesas Electronics Canada Limited**
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

**Renesas Electronics Europe Limited**
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

**Renesas Electronics Europe GmbH**
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

**Renesas Electronics (China) Co., Ltd.**
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

**Renesas Electronics (Shanghai) Co., Ltd.**
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

**Renesas Electronics Hong Kong Limited**
Unit 1601-1613, 16F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022/9044

**Renesas Electronics Taiwan Co., Ltd.**
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

**Renesas Electronics Singapore Pte. Ltd.**
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

**Renesas Electronics Malaysia Sdn.Bhd.**
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +603-7955-9390, Fax: +603-7955-9510

**Renesas Electronics Korea Co., Ltd.**
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141