RX610 Group

Deep Software Standby Mode (On-chip RAM0 Internal Power Supply Shutoff)

R01AN0202EJ0100
Rev.1.00
Dec 09, 2010

Abstract

This application note presents a sample program showing the use of the deep software standby mode.

Target Device

RX610 Group

Introduction

This application note was compiled using as a basis RX610 Group MCUs, on which operation has been confirmed.

The program can be used with other RX Family MCUs that have the same I/O registers (peripheral device control registers) as the RX610 Group. Check the latest version of the manual for any additions and modifications to the functions or address map used by this application note. Careful evaluation is recommended before using this application note.

The program works with an endian specification of big or little and with left or right specified as the bit order.

Contents

1. Overview

RX610 MCUs have a power-down (low-power) mode called deep software standby mode.

In deep software standby mode, the CPU, on-chip peripheral functions, the on-chip RAM1 module, and all oscillator functions stop operating. Furthermore, the internal power supply to these modules is shut off, providing a significant reduction in power consumption. At this time, the contents of all the registers of the CPU and on-chip peripheral functions, and all on-chip RAM1 data, become undefined. The states of the address bus and bus control signals can be selected by setting the deep standby control register (DPSBYCR).

By means of a deep standby control register setting, it is also possible to select whether or not power is provided to the on-chip RAM0 module by the internal power supply. If the internal power supply continues to operate, the data in on-chip RAM0 is retained. If the internal power supply to on-chip RAM0 is shut off, overall power consumption is reduced, but the data in on-chip RAM0 becomes undefined.

The deep standby backup registers (DPSBKRm, m = 0 to 31) are a set of 32-byte readable/writable registers for storing data in deep software standby mode. The value of this register is retained even in deep software standby mode where on-chip RAM data is not retained.

This sample program shows how to shut off the power supply to the on-chip RAM0 module and make use of the deep standby backup registers in deep software standby mode.

Note: Deep software standby mode uses the WAIT instruction. WAIT is a privileged instruction. As a result, if the WAIT instruction is executed when the processor mode is set to user mode, a privileged instruction exception occurs and it is not possible to transition to deep software standby mode.
In order to use deep software standby mode, it is necessary first to set the processor mode to supervisor mode.

2. Operation

2.1 Operation Specifications

Figure 1 is a connection diagram for this sample application.

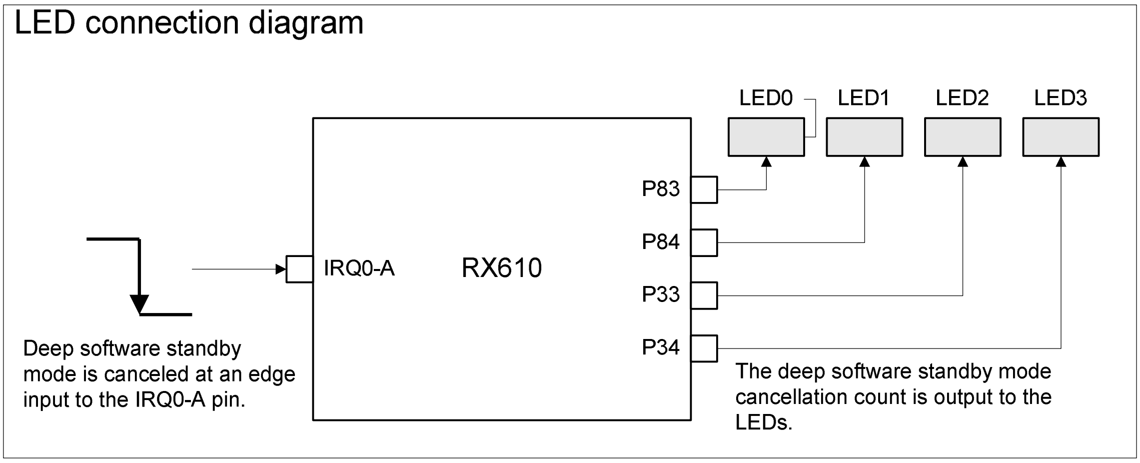


Figure 1 Connection Diagram

In the sample program, deep software standby mode is canceled by edge input on the IRQ0-A pin.

In addition, a counter variable is used to count the cancellations of deep software standby mode. The counter variable is incremented each time deep software standby mode is canceled. The value of the counter variable is output to the LEDs on the evaluation board.

The wait time from the point the source that clears deep software standby mode occurs until deep software standby mode is cleared is set to 524,288 states. In addition, the states of the address bus and bus control signals in deep software standby mode are set to high-impedance.

Note:

1. When transitioning to deep software standby mode, the device passes through software standby mode.
   If a cancel source for software standby mode has been set, and contention occurs with the software standby mode cancel source during the transition to software standby mode, software standby mode is canceled and the interrupt exception handler specified for the cancel source is launched. In this case, no transition to deep software standby mode occurs.
   No software standby mode cancel source is set in the sample program, so the above-mentioned problem should not arise.
2. The DMAC DMSCNT.DMST and DTC DTCST.DTCST bits must be cleared to 0 before executing a WAIT instruction. If the WDT is used in watchdog timer mode, the microcontroller will not transition to software standby mode. The watchdog timer must be stopped before executing the WAIT instruction.

2.2 Operation Sequence

Figure 2 shows the deep software standby mode operation sequence in the sample program.

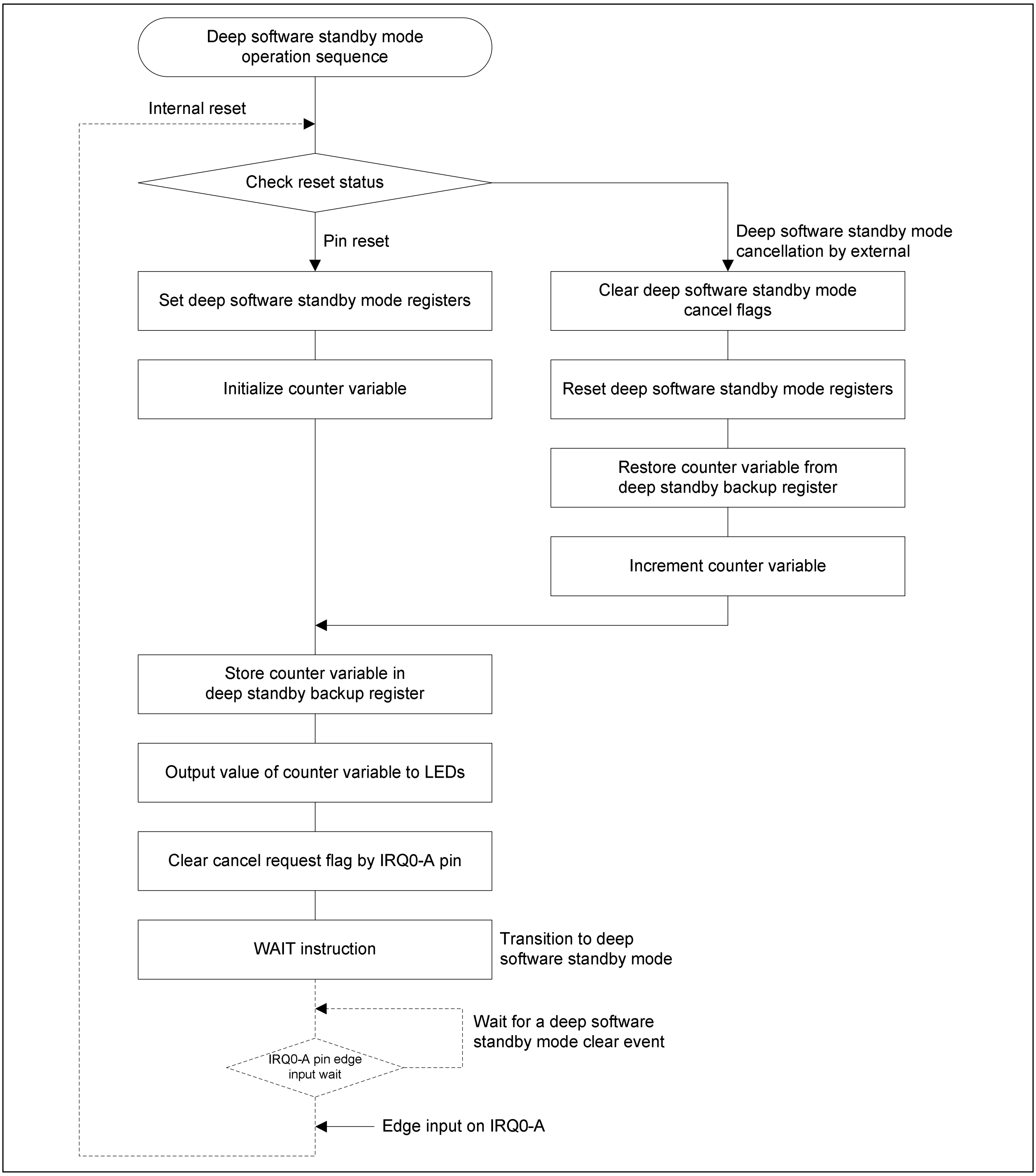


Figure 2 Deep Software Standby Mode Operation Sequence

The sample program repeatedly transitions to and cancels deep software standby mode, increments a counter variable at each cancellation, and displays the value of the counter variable using LEDs. In addition, the power supply to the on-chip RAM0 module is shut off in deep software standby mode. When this occurs, the value of the counter variable is stored in the deep standby backup register, where it is retained.

After a reset the reset status is checked. In case of a pin reset, settings are made in the deep software standby mode registers and the counter variable is initialized. If deep software standby mode was canceled by an external interrupt, the deep software standby mode cancel flags are cleared, the deep software standby mode registers are reset, the counter variable is restored from the deep standby backup register, and then the counter variable is incremented.

After this, the value of the counter variable is stored in the deep standby backup register, and the counter variable value is output to the LEDs. Finally, the WAIT instruction is executed, causing the device to transition to deep software standby mode.

When an event that clears deep software standby mode occurs, at the same time the clock oscillator is started, internal power supply is started and an internal reset signal for the whole RX610 is generated. After the set wait time has elapsed, the stabilized clock is supplied to the whole RX610 and the internal reset is cleared. Deep software standby mode is cleared at the same time as the internal reset is cleared and reset exception handling starts.

3. Program Description

3.1 Register Settings

Table 1 lists the setting registers and setting values for deep software standby mode used in the sample program.

Table 1 Deep Software Standby Mode Register Settings

Register Name	Symbol	Setting Value
Standby control register	SBYCR	8F00h
Deep standby control register	DPSBYCR	F1h
Deep standby wait control register	DPSWCR	0Fh
Deep standby interrupt enable register	DPSIER	01h
Deep standby interrupt flag register	DPSIFR	00h
Deep standby interrupt edge register	DPSIEGR	00h
Reset status register	RSTSR	00h

3.2 Setting Register Details

Details of each of the setting registers are provided below. Note that the bits not listed below should be left in their initial settings following a reset.

Standby Control Register (SBYCR) **Address: 0008000Ch**

Bit	Symbol	Setting Value	Bit Name	Description
b15	SSBY	1	Software standby bit	1: Transition to software standby mode after execution of WAIT instruction
b14	OPE	0	Output port enable bit	0: Address bus and bus control signals set to high-impedance state in software standby mode and deep software standby mode

Deep Standby Control Register (DPSBYCR) **Address: 0008C280h**

Bit	Symbol	Setting Value	Bit Name	Description
b7	DPSBY	1	Deep software standby bit	SSBY b7 1 1: Transition to deep software standby mode after execution of WAIT instruction
b6	IOKEEP	1	I/O port retention bit	0: Simultaneous cancellation of deep software standby mode and I/O port retention 1: Cancellation of I/O port retention when 0 is written to IOKEEP bit after cancellation of deep software standby mode
b5	RAMCUT2	1	On-chip RAM off 2 bit	b5 b4 b0 1 1 1: Power not supplied to on-chip RAM (RAM0) in deep software standby mode
b4	RAMCUT1	1	On-chip RAM off 1 bit	
b0	RAMCUT0	1	On-chip RAM off 0 bit	

Deep Standby Wait Control Register (DPSWCR) **Address: 0008C281h**

Bit	Symbol	Setting Value	Bit Name	Description
b5 to b0	WTSTS [5:0]	001111	Deep software standby waiting time bits	b5 b0 0 0 1 1 1 1: Waiting time = 524,288 states

Deep Standby Interrupt Enable Register (DPSIER) **Address: 0008C282h**

Bit	Symbol	Setting Value	Bit Name	Description
b0	DIRQ0E	1	IRQ0 pin enable bit	1: Canceling of deep software standby mode by IRQ0 pin enabled

Deep Standby Interrupt Flag Register (DPSIFR) **Address: 0008C283h**

Bit	Symbol	Setting Value	Bit Name	Description
b0	DIRQ0F	0/1	IRQ0 deep standby cancel flag	0: No cancel request generated by IRQ0 pin 1: Cancel request generated by IRQ0 pin

Deep Standby Interrupt Edge Register (DPSIEGR) **Address: 0008C284h**

Bit	Symbol	Setting Value	Bit Name	Description
b0	DIRQ0EG	0	IRQ0 edge select bit	0: Cancel request generated at falling edge

Reset Status Register (RSTSR) **Address: 0008C285h**

Bit	Symbol	Setting Value	Bit Name	Description
b7	DPSRSTF	0/1	Deep software standby reset flag	0: No deep software standby mode cancel source generated by external interrupt 1: Deep software standby mode cancel source generated by external interrupt

3.3 File Structure

Table 2 shows the file structure. In addition to the files listed in table 2, some files generated automatically by HEW are used as well.

Table 2 File Structure

File Name	Description
resetprg.c *	Initial settings process
main.c	Main process, deep software standby mode settings/transition

Note: * This file is generated automatically by HEW, but it is necessary to comment out the processor mode switching portion of the PowerON_Reset_PC function. For details, see 3.5.1, PowerON_Reset_PC Function.

3.4 Variable Structure

Table 3 lists the variables used.

Table 3 List of Variables

Variable Name	Type	Description
dpstby_count	unsigned char	Deep software standby mode cancellation counter Counts the number of times deep software standby mode is canceled.

3.5 Function Structure

The specifications of the functions are described below.

Table 4 List of Functions

Variable Name	File Name	Description
PowerON_Reset_PC *	resetprg.c	Initial settings after reset function
main	main.c	Main function
init	main.c	MCU initial settings function, operating clock setting
dpstby_init_and_recovery	main.c	Deep software standby mode settings
clear_PSW_I	main.c	Interrupt disable setting
output_LED	main.c	LED output function

Note: * This function is generated automatically by HEW, but it is necessary to comment out the processor mode switching portion. For details, see 3.5.1, PowerON_Reset_PC Function.

3.5.1 PowerON_Reset_PC Function

(1) Functional overview

The PowerON_Reset_PC function is called after a reset. It uses embedded functions and a standard library function to make settings to CPU registers. Then it calls the main function.

Note: This function is generated automatically by HEW. In its initial form, it switches the processor mode to user mode. However, using the WAIT instruction in this mode generates a privileged instruction exception that prevents the transition to deep software standby mode. The processor mode switching portion of the PowerON_Reset_PC function must therefore be commented out for use with the sample program.

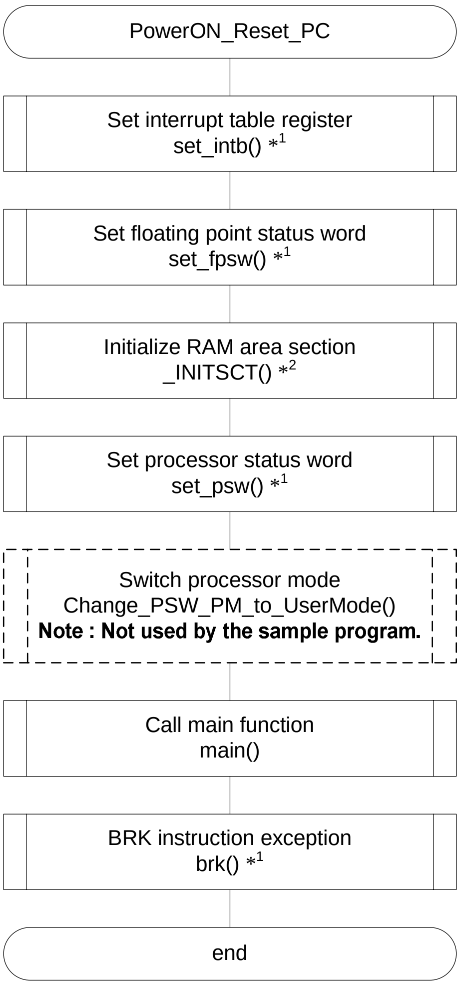
(2) Arguments

None

(3) Return values

None

(4) Flowchart



Notes: 1. Embedded function (set_intb, set_fpsw, set_psw, and brk functions)
2. Standard library function (_INITSCT function)
For details, see “Development Environment Manual” in 5, Reference Documents.

Figure 3 Flowchart (PowerON_Reset_PC Function)

3.5.2 main Function

(1) Functional overview

The main function makes settings for deep software standby mode, then transitions the device to the power-down (low-power) state.

It calls the dpstby_init_and_recovery function to make settings for deep software standby mode.

After the deep software standby mode settings, the standby mode cancellation timer value is output to the LEDs, the output port is stopped, and the IRQ0-A pin is used to clear the cancel request flag. Then the WAIT instruction is executed to transition to deep software standby mode.

(2) Arguments

None

(3) Return values

None

(4) Flowchart

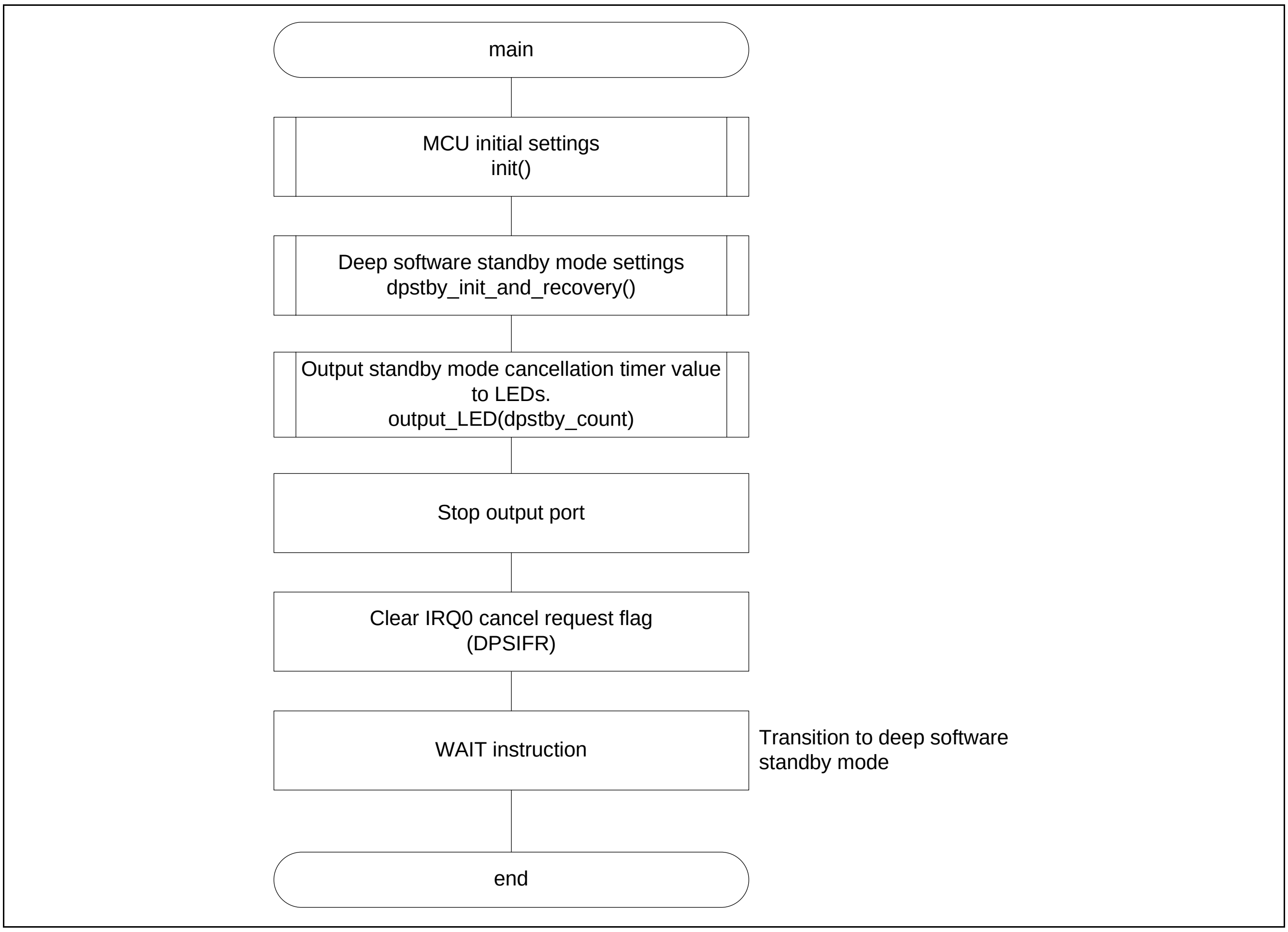


Figure 4 Flowchart (main Function)

3.5.3 init Function

(1) Functional overview

The init function performs initial program settings, setting the operating clock, disabling interrupts, and setting the output port.

(2) Arguments

None

(3) Return values

None

(4) Flowchart

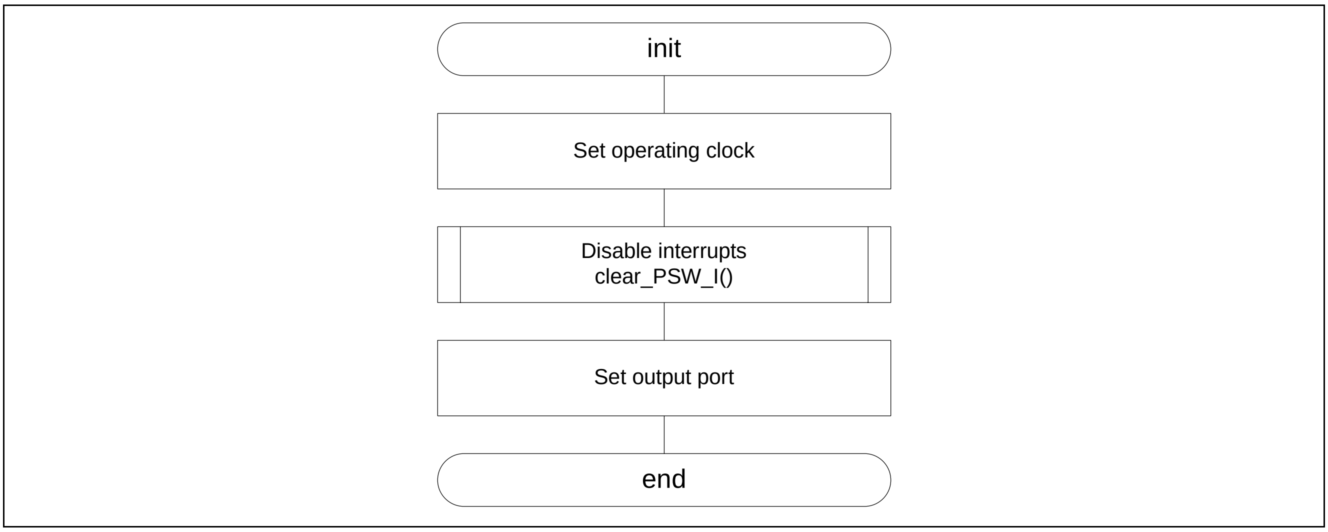


Figure 5 Flowchart (init Function)

3.5.4 dpstby_init_and_recovery Function

(1) Functional overview

The dpstby_init_and_recovery function performs settings for deep software standby mode and handles operation after its cancellation.

The reset status is checked at the start of the function.

In case of a pin reset, settings are made in the deep software standby mode registers and the deep software standby mode cancellation counter is initialized.

If deep software standby mode was canceled by an external interrupt, the deep software standby mode cancel flags are cleared and the deep software standby mode registers are reset. Also, the deep software standby mode cancellation counter value is restored from the deep standby backup register, and then the counter variable is incremented.

After this, the value of the deep software standby mode cancellation counter is stored in the deep standby backup register, and function operation ends.

(2) Arguments

None

(3) Return values

None

(4) Flowchart

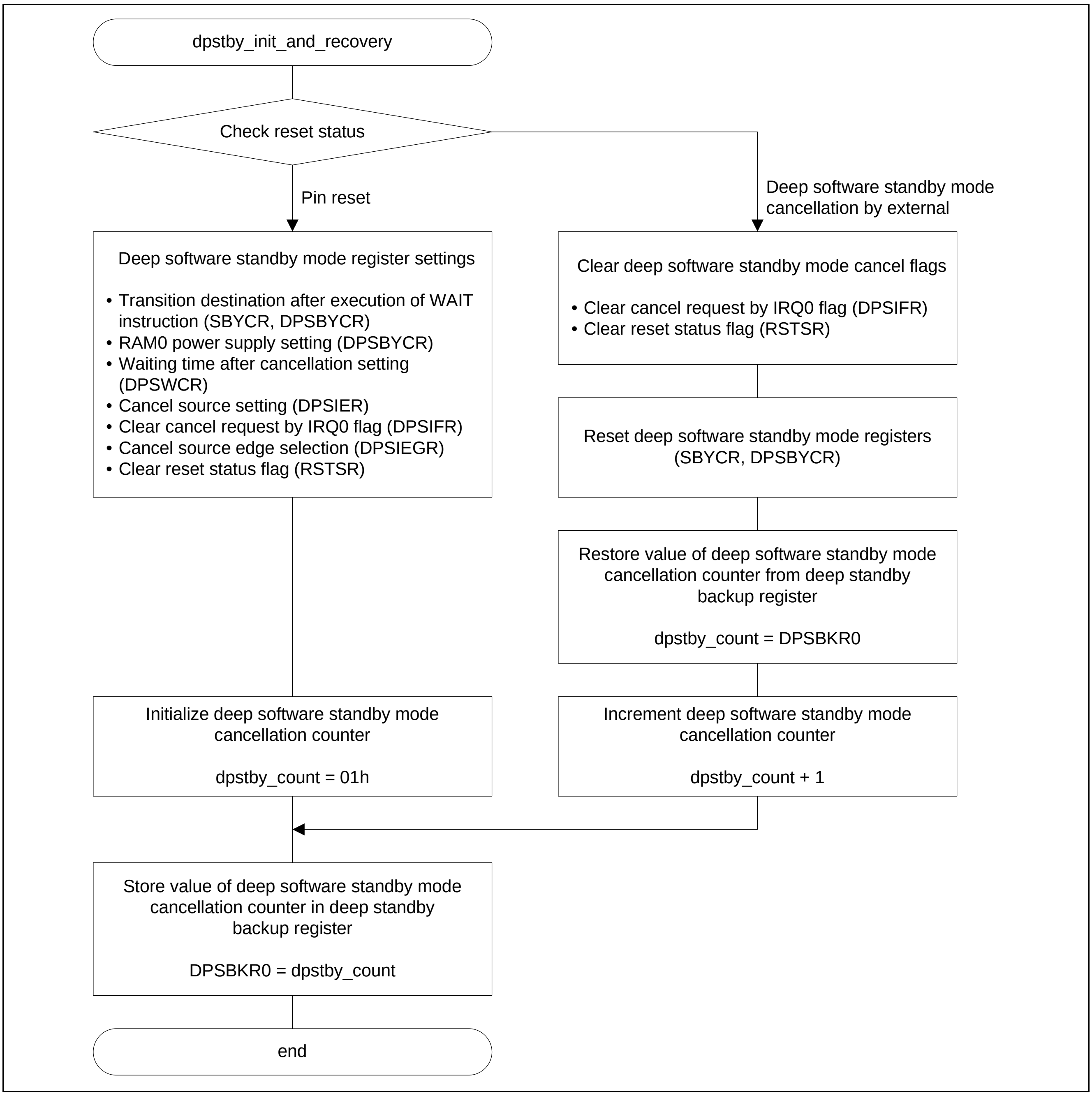


Figure 6 Flowchart (dpstby_init_and_recovery Function)

3.5.5 clear_PSW_I Function

(1) Functional overview

The clear_PSW_I function clears the I bit in PSW to disable interrupts.

(2) Arguments

None

(3) Return values

None

(4) Flowchart

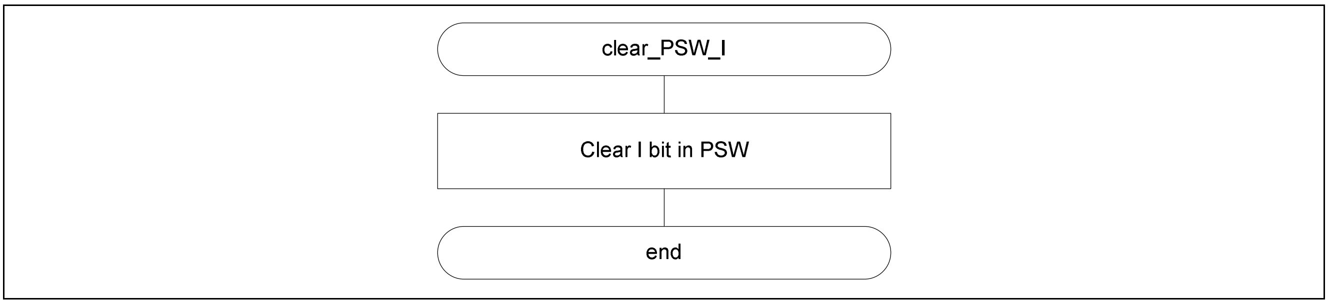


Figure 7 Flowchart (clear_PSW_I Function)

3.5.6 output_LED Function

(1) Functional overview

The output_LED function sends output to the LEDs. The argument is converted to a bit representation, the lower 4 bits are output to the LEDs, and the LEDs remain lit for a fixed time period.

(2) Arguments

Arguments	Type	Description
led_data	unsigned char	LED output value (The lowest 4 bits are output.)

(3) Return values

None

(4) Flowchart

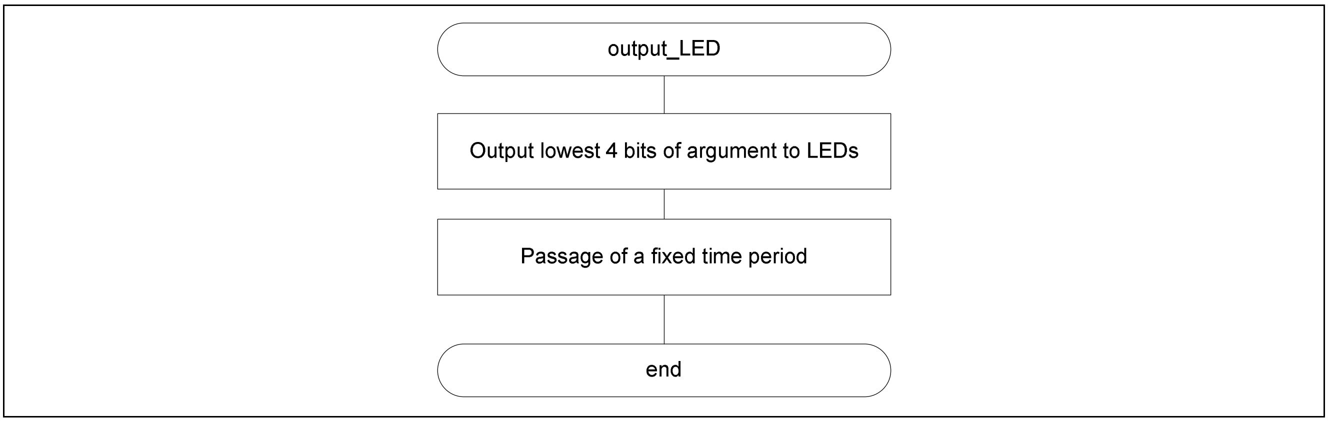


Figure 8 Flowchart (output_LED Function)

4. Operation Confirmation Environment

Table 5 shows the environment on which the operation of the example program has been confirmed.

Table 5 Operation Confirmation Environment

Item	Name
Device	RX610 (R5F56108VNFP)
Board	Evaluation board
Power supply voltage	5.0 V (CPU operating voltage: 3.3 V)
Input clock	12.5 MHz (ICLK = 100 MHz, PCLK = 50 MHz, BCLK = 25 MHz)
Operating temperature	Room temperature
HEW	Version 4.07.00.007
Toolchain	RX Standard Toolchain (V.1.0.0.0) RX Family C/C++ Compile Driver V.1.00.00.001 RX Family C/C++ Compiler V.1.00.00.001 RX Family Assembler V.1.00.00.001 Optimizing Linkage Editor V.10.00.00.001 RX Family C/C++ Standard Library Generator V.1.00.00.001
Debugger	RX E20 SYSTEM V.1.00.00.000

5. Reference Documents

- Hardware Manual
  RX610 Group Hardware Manual
  (The latest version can be downloaded from the Renesas Electronics Web site.)
- Development Environment Manual
  RX Family C/C++ Compiler Package User's Manual
  (The latest version can be downloaded from the Renesas Electronics Web site.)
- Technical Updates
  (The latest information can be downloaded from the Renesas Electronics Web site.)

Website and Support

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

Inquiries
http://www.renesas.com/inquiry

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Dec.09.10	—	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 document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

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 a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of an MPU or MCU in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

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, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
7F, 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: +60-3-7955-9390, Fax: +60-3-7955-9510

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

Colophon 1.0