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.

RENESAS

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.

H8/300L SLP Series

Example of Connecting the MAXIM SD ADC (16 Bits)

Introduction

The 16-bit MAX1408 ΣΔ A/D conversion results are serially input to the H8/38024 CPU by providing a clock, and the outputs are displayed on an array of 7-segment LEDs as a 4-digit hexadecimal number.

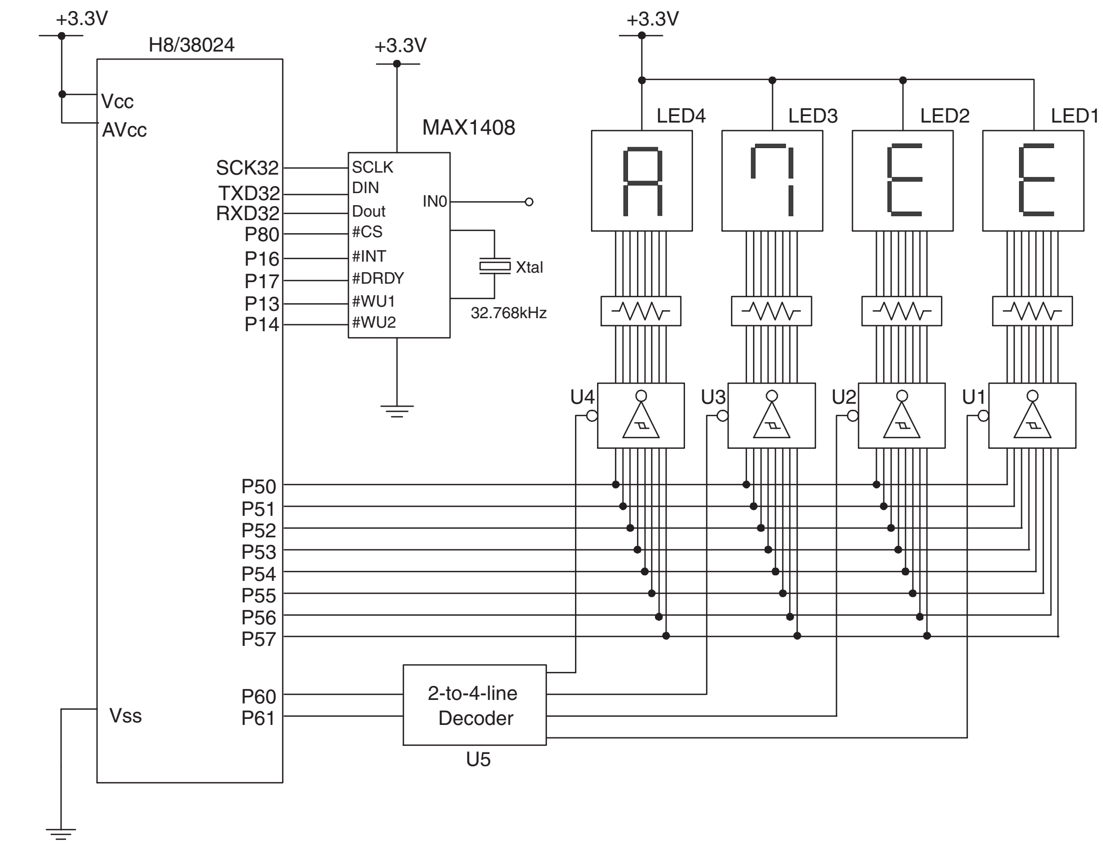
Target Device

H8/300L Super Low Power Series H8/38024 CPU

Contents

1. Specifications

1. Figure 1 shows an example of the hardware configuration required to connect a serial output analog/digital converter (hereinafter referred to as an ADC). Analog voltages are input to analog input pin 0 (IN0 pin) of the ADC and are then A/D converted by the 16-bit delta sigma modulator.
2. The conversion results are input to the microcomputer by using the clock applied by the microcomputer through synchronous serial communication.
3. The received 16-bit data is converted to a 4-digit hexadecimal number and the result is displayed on a 7-segment LED array.



Legend:

MAX1408	: 16-bit ADC (MAX1408, made by Maxim Integrated Products, Inc.)
Xtal	: Quartz crystal (32.768 kHz)
LED1 to LED4	: Common-anode 7-segment LEDs
U1 to U4	: Tri-state output inverter drivers (HD74LS240)
U5	: 2-to-4-line decoder (HD74HC139)

Figure 1 Hardware Configuration

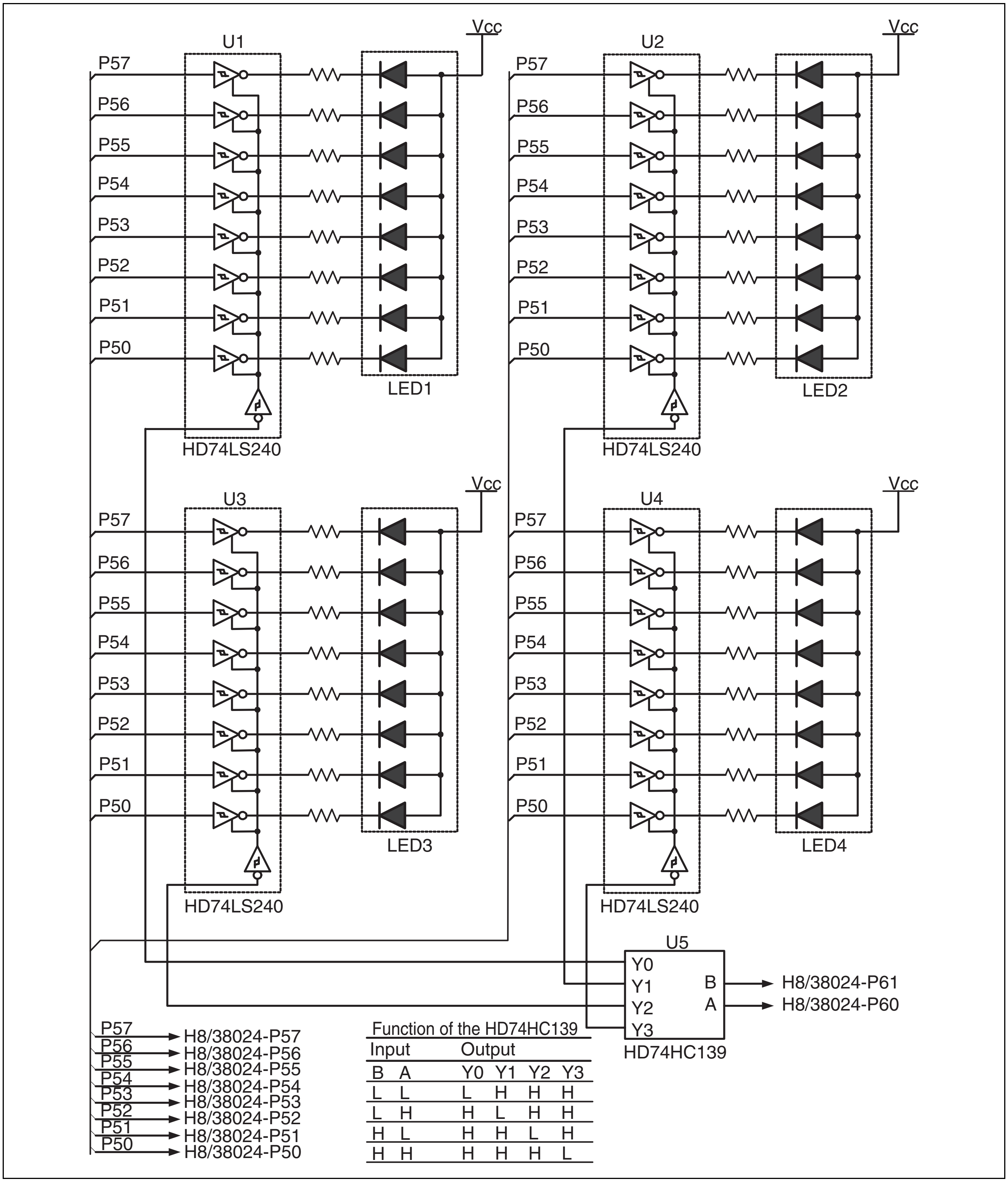
4. In this sample task, the operating voltage (Vcc) and analog power supply voltage (AVcc) of the H8/38024 are both 3.3 V, the OSC clock frequency is 10 MHz, and the watch clock frequency is 32.768 kHz.
5. The ADC used in this sample task is a serial-output analog digital converter (model MAX1408) manufactured by Maxim Integrated Products, Inc. The specifications are as follows:

   A. Characteristics of the MAX1408.

      a. Power voltage: +2.7 V to +3.6 V

   b. Power consumption: 1.15mA (2.5 µA in the sleep mode)
   c. Package: 28-pin SSOP
   d. Multi-channel 16-bit sigma-delta ADC
B. The MAX1418 incorporates a delta-sigma modulator for A/D conversion. Since the converter has a high 16-bit resolution, it can be used for medical equipment, industrial controllers, portable devices, automatic metering, and robots.

6. The circuit in this sample task operates as follows.
   A. Voltages input from analog pin 0 (IN0) of the MAX1418 are delta-sigma converted in cooperation with an external crystal resonator.
   B. The 16-bit A/D converted data is input to the microcomputer by providing a serial clock.
   C. The input 16-bit data is converted to a 4-digit hexadecimal number and displayed on the LED array.
   D. For example, when a voltage of 0.82 V is applied to the IN0 pin, A/D conversion produces "1010 0111 1110 1110". The microcomputer then latches and converts it and then generates hexadecimal "A7EE" for the LED display.
   E. A reference voltage of "1.25" is displayed on the LEDs as "FFFF" while 0 V is displayed as "0000".
7. In this sample task, the 7-segment LED display is set up by connecting the port outputs to the tri-state output inverter drivers (HD74LS240) and connecting the driver outputs to the cathodes of the 7-segment LEDs. In addition, all the ports used for the four 7-segment LEDs are connected to the 7-segment LEDs and the enable pins of the tri-state inverter drivers are used to renew between the 7-segment LEDs. The signal generation for renewing between the LEDs is controlled by the two port outputs of a 2-to-4 line decoder (HD74HC139). Figure 2 illustrates the method of controlling the 7-segment LEDs.



Input		Output			
B	A	Y0	Y1	Y2	Y3
L	L	L	H	H	H
L	H	H	L	H	H
H	L	H	H	L	H
H	H	H	H	H	L

Figure 2 Method of Controlling 7-Segment LEDs

8. In this sample task, the 16-bit data obtained as the result of conversion by the MAX1408 ΣΔ A/D, is displayed on the 7-segment LED array as a 4-digit hexadecimal number. Figure 3 shows how the result of MAX1408 ΣΔ A/D conversion is displayed on the LEDs.

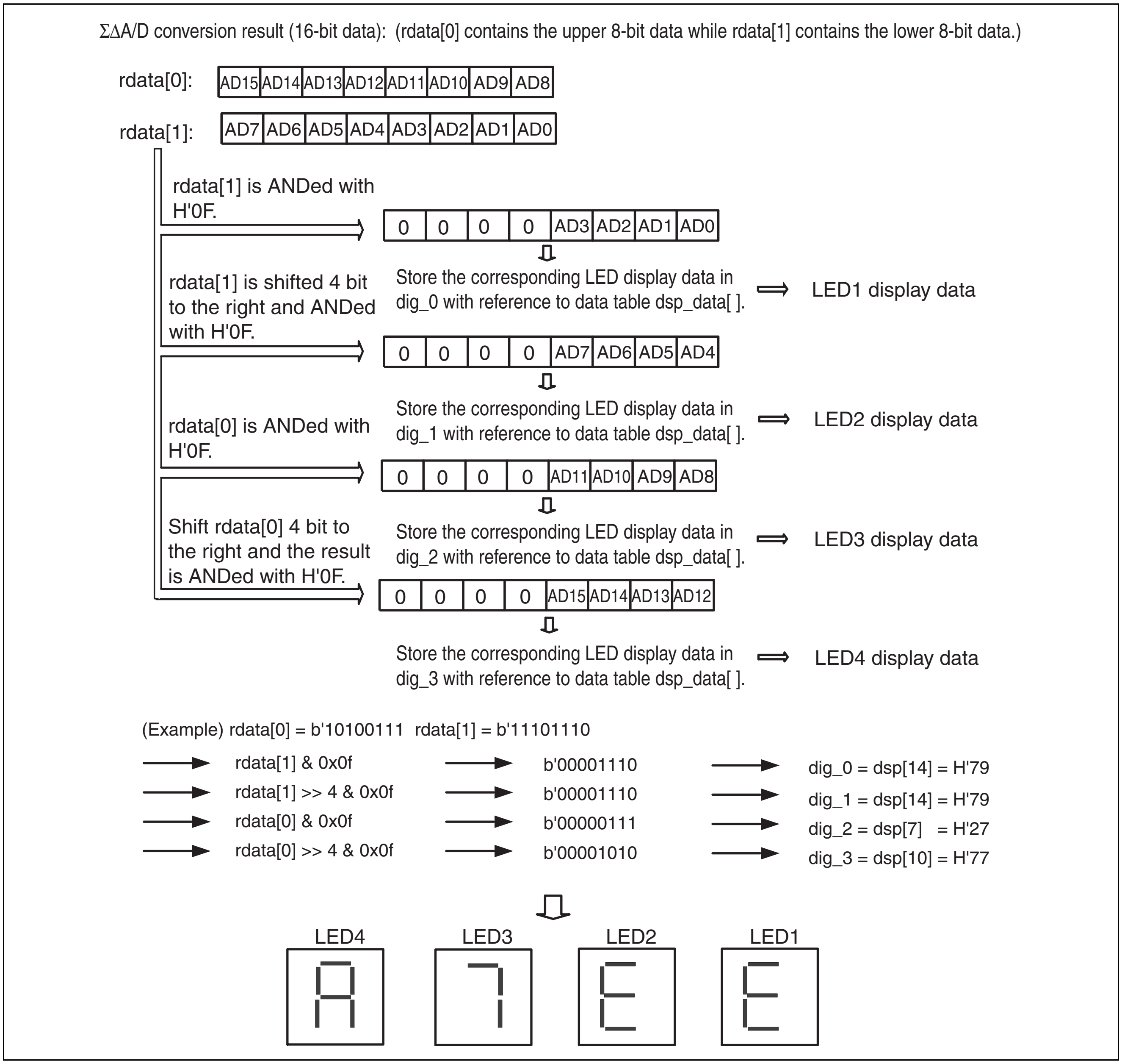


Figure 3 Display of Result of MAX1408 ΣΔ A/D Conversion on the LEDs

2. Description of Functions

1. Figure 4 is a block diagram of the H8/38024 functions used in this task. Table 1 lists the function allocations.

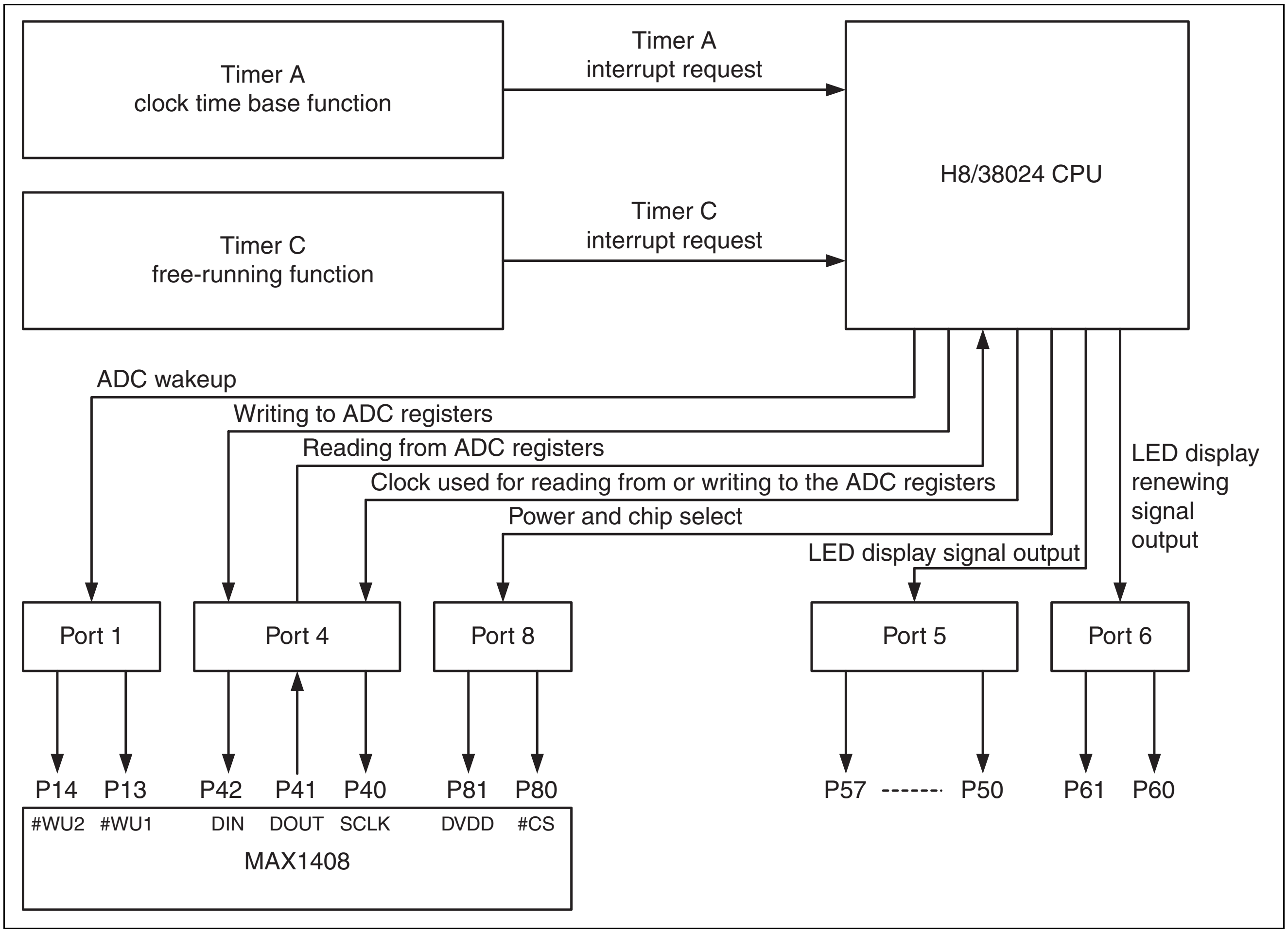


Figure 4 Block Diagram of Functions Used

Table 1 Function Assignments

Function	Function assignment
Timer A	The clock time base function of timer A is used to convert the DATA register value, obtained as a result of MAX1408 ΣΔ A/D conversion, to LED display data. Then it is stored into RAM.
Timer C	The free-running function of timer C is used to control the renewing between the 7-segment LEDs. Each of the four 7-segment LEDs is lit in sequence at intervals of 3.2768 ms (the time at which timer C overflows), enabling dynamic illumination of the LEDs.
Port 1	The P13 and P14 output pins of port 1 are used to wake up the MAX1408 ΣΔ ADC.
Port 4	The P42 output pin of port 4 is used to write to the MAX1408 ΣΔ ADC registers. The P40 output pin of port 4 is used to set the clock for writing to or reading from the MAX1408 ΣΔ ADC registers. The P41 input pin of port 4 is used to read from the MAX1408 ΣΔ ADC registers.
Port 5	The P50-P57 output pins of port 5 are used to display data on the currently active 7-segment LED. The 10-bit data, obtained as a result of the MAX1408 ΣΔ A/D conversion, is converted to 4-digit hexadecimal display data and output to the LED.
Port 6	The P60 and P61 output pins of port 6 are used to renew between the four 7-segment LEDs. The P60 and P61 output pins are connected to the input pins of the 2-to-4 line decoder.
Port 8	The P80 and P81 output pins are used to connect to the power and chip select of the MAX1408 ΣΔ ADC.

2. Figure 5 shows how the 7-segment LED used in this task is connected. A high output from port 5 lights the corresponding segment of the LED, as shown in the figure. Table 2 lists the relationships between port 5 outputs and display on the LED display.

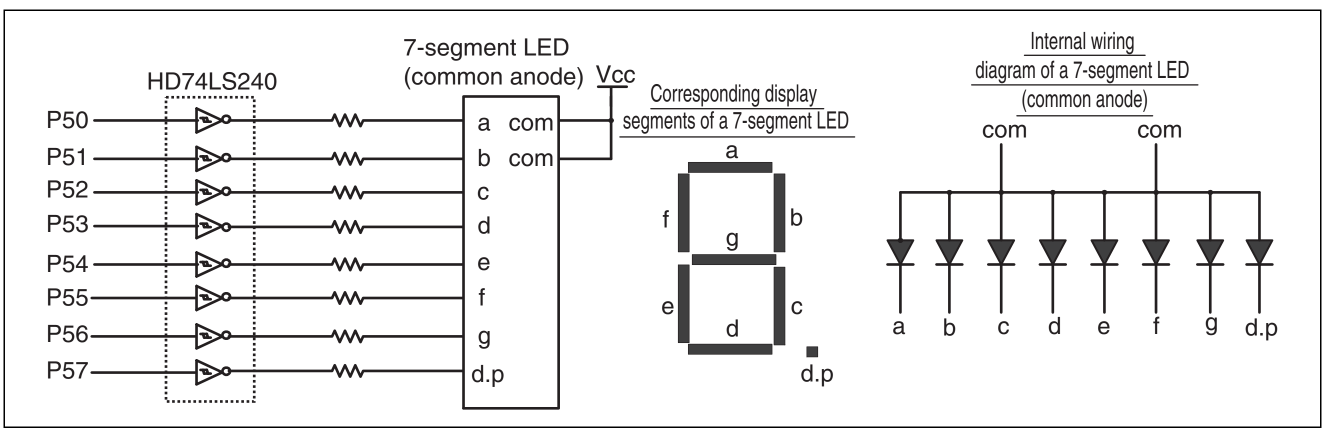
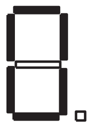
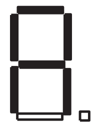
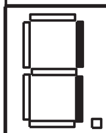
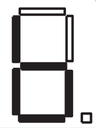
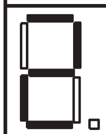
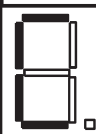
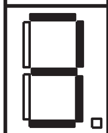
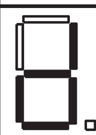
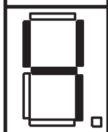
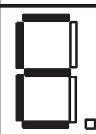
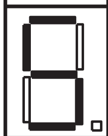
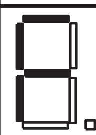
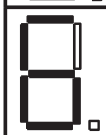
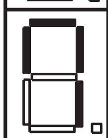
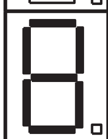
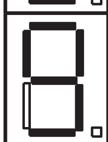


Figure 5 Connection Diagram and Internal Connections of 7-Segment LED

Table 2 Relationship between Port 5 Outputs and 7-Segment LED Display Data

LED display	Port 5 output data								LED display	Port 5 output data							
	P57	P56	P55	P54	P53	P52	P51	P50		P57	P56	P55	P54	P53	P52	P51	P50
	0	0	1	1	1	1	1	1		0	1	1	1	0	1	1	1
	0	0	0	0	0	1	1	0		0	1	1	1	1	1	0	0
	0	1	0	1	1	0	1	1		0	0	1	1	1	0	0	1
	0	1	0	0	1	1	1	1		0	1	0	1	1	1	1	0
	0	1	1	0	0	1	1	0		0	1	1	1	1	0	0	1
	0	1	1	0	1	1	0	1		0	1	1	1	0	0	0	1
	0	1	1	1	1	1	0	1									
	0	0	1	0	0	1	1	1									
	0	1	1	1	1	1	1	1									
	0	1	1	0	1	1	1	1									

3. Description of Operation

1. Figure 6 shows how the 16-bit DATA register value, obtained as a result of MAX1408 ΣΔ A/D conversion, is converted to LED display data and then stored into RAM. In this sample task, the result of MAX1408 ΣΔ A/D conversion is converted to LED display data and stored into RAM at the timer A interrupt interval in the tmra routine, as shown in this figure.

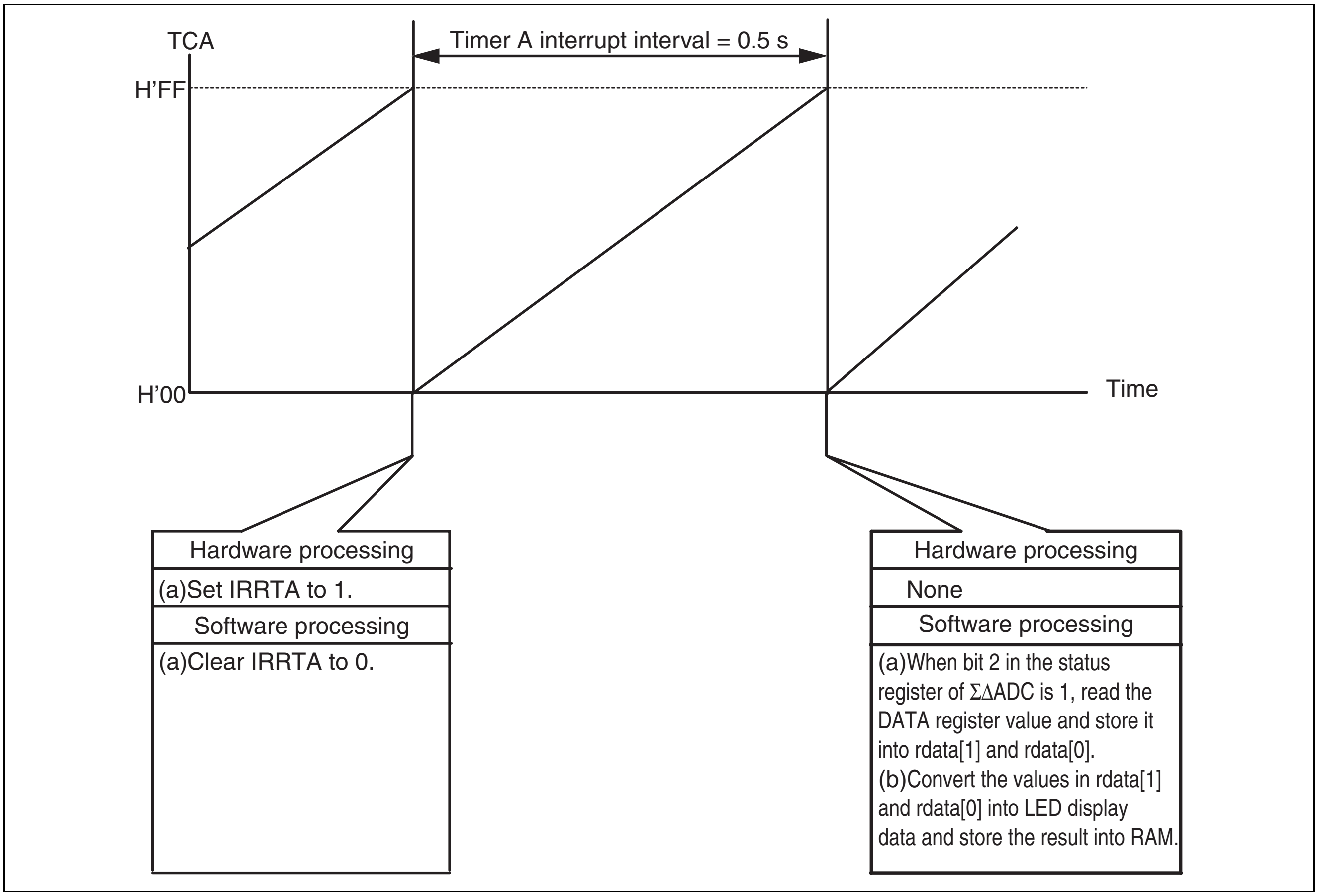


Figure 6 Description of Storing the ΣΔ A/D Conversion Result into RAM

2. The following describes the descriptions of 7-segment LED operation. Figure 7 shows how a value of "A7EE" is displayed on LED4 to LED1. As shown in the figure, each of LED1 to LED4 is lit in sequence at the timer C overflow interval, resulting in dynamic display on the 7-segment LED.

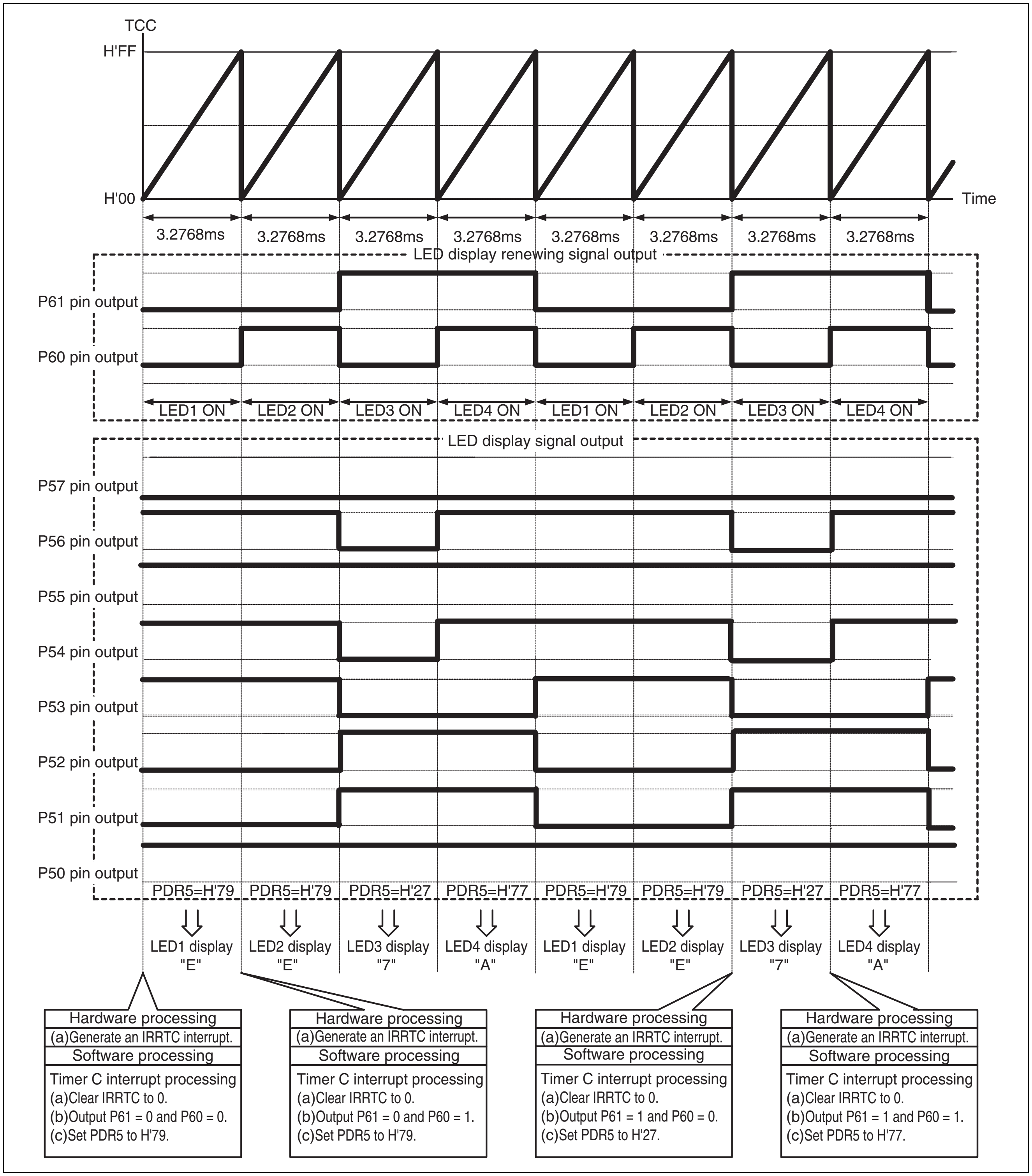


Figure 7 Descriptions of 7-Segment LED Display Control

4. Description of Software

1. Modules
   Table 3 lists the modules used in this sample task.

Table 3 Modules

Module name	Label name	Function
Main routine	main	Makes the initial settings, calls the initial setting function for the MAX1408 ΣΔ A/D converter, and then enables an interrupt.
Timer A interrupt processing routine	tmra	Clears the interrupt flag. This routine also converts the DATA register value, obtained as a result of MAX1408 ΣΔ A/D conversion, to LED display data after bit 2 of the STATUS register is set to 1, and then stores it into RAM.
Timer C interrupt processing routine	tmrc	Clears the interrupt flag, outputs the LED display data, and controls LED display switching.
ADC_Init( ) processing routine	ADC_Init	Initializes the MAX1408 ΣΔ A/D converter.
DataIn( ) processing routine	DataIn	Writes to or reads from the MAX1408 ΣΔ A/D converter registers.
DataOut( ) processing routine	DataOut	Writes to the MAX1408 ΣΔ A/D converter registers.

2. Arguments
   No arguments are used in this sample task.
3. Internal Registers
   Table 4 lists the internal registers used in this task.

Table 4 Internal Registers

Register name		Description	Address	Setting
TMA		Timer mode register A: Selects a prescaler and input clock.	H'FFB0	H'0C (At initial setting)
	TMA3	Internal clock select 3: Selects the operating mode of timer A. When TMA3 = 1, it functions as a clock time base that counts the output of prescaler W.	Bit 3	1
	TMA2	Internal clock select 2-0: When TMA3 = 1, the clock time base (32.768kHz) is selected. When TMA2 = 1, TMA1 = 0, and TMA0 = 0, TCA reset is selected. When TMA2 = 0, TMA1 = 0, and TMA0 = 1, the TCA overflow period is 0.5 s.	Bit 2	0/1
	TMA1		Bit 1	0
	TMA0		Bit 0	0/1
TMC		Timer mode register C: Selects automatic reloading, controls count-up or count-down of the counter, and controls the input clock.	H'FFB4	H'1B
	TMC7	Selects the automatic reload function: When TMC7 = 0, it selects the interval function.	Bit 7	0
	TMC6	Controls counting-up or count-down: When TMC6 = 0 and TMC5 = 0, TCC functions as an up-counter.	Bit 6	0
	TMC5		Bit 5	0

Register name		Description	Address	Setting
	TMC2	Clock select:	Bit 2	0
	TMC1	When TMC2 = 0, TMC = 1, and TMC = 1, counting	Bit 1	1
	TMC0	is performed using an internal clock of ϕ/64.	Bit 0	1
TLC		Timer load register C: Sets the TCC reload value.	H'FFB5	H'00
CKSTRP1		Clock stop register 1: Bit 5: Controls module standby mode of SCI3. When this bit is 1, module standby mode of SCI3 is released.	H'FFFA	H'FF
PDR1		Port data register 1	H'FFD4	H'00
PCR1		Port control register 1: Specifies, bit-by-bit, whether each of the pins (P17, P16, P14, and P13) of port 1 is an input pin or output pin. When PCR1 = H'3F, the P14 and P13 pins function as output pins.	H'FFE4	H'3F
PUCR1		Port pull-up control register 1: Controls, bit-by-bit, the pull-up MOS of each of P17, P16, P14, and P13. When PUCR1 = H'00, the pull-up MOS for the P17, P16, P14, and P13 is off.	H'FFE0	H'00
PMR1		Port mode register 1:	H'FFC8	H'42
	IRQ3	P17/IRQ3/TMIF pin function switching: When this bit is 0, this pin functions as the P17 general I/O port.	Bit 7	0
	IRQ4	P14/IRQ4/ADTRG pin function switching: When this bit is 0, this pin functions as the P14 general I/O port.	Bit 4	0
	TMIG	P13/TMIG pin function switching: When this bit is 0, this pin functions as the P13 general I/O port.	Bit 3	0
PMR2		Port mode register 2: Turns the PMOS for the P35 pin on or off, selects a watchdog timer clock, selects TMIG noise cancel, and controls P43/IRQ0 pin function switching.	H'FFC9	H'D8
PDR4		Port data register 4	H'FFD7	H'F8
PCR4		Port control register 4: Specifies, bit-by-bit, whether the pins (P42-P40) of port 4 is an input pin or output pin. When PCR4 = H'FD, the P42 and P40 pins function as output pins and the P41 pin functions as an input pin.	H'FFE7	H'FD
PUCR6		Port pull-up control register 6: Controls, bit-by-bit, the pull-up MOS of each pin of port 6 that is set as an input port. When PUCR6 = H'00, the pull-up MOS for the P67-P60 pins is off.	H'FFE3	H'00
PDR6		Port data register 6: General I/O port data register for port 6	H'FFD9	H'00

Register name		Description	Address	Setting
PCR6		Port control register 6: Selects bit-by-bit, whether each pin used as a general I/O port for port 6 is an input pin or output pin. When PCR6 = H'FF, the P67-P60 pins function as general output pins.	H'FFE9	H'FF
PMR5		Port mode register 5: Sets the pin function of port 5.	H'FFCC	H'00
	WKP7	P57/_WKP7/SEG7 pin function switching: When this bit is 0, the P57 general I/O port function is selected.	Bit 7	0
	WKP6	P56/_WKP6/SEG6 pin function switching: When this bit is 0, the P56 general I/O port function is selected.	Bit 6	0
	WKP5	P55/_WKP5/SEG5 pin function switching: When this bit is 0, the P55 general I/O port function is selected.	Bit 5	0
	WKP4	P54/_WKP4/SEG4 pin function switching: When this bit is 0, the P54 general I/O port function is selected.	Bit 4	0
	WKP3	P53/_WKP3/SEG3 pin function switching: When this bit is 0, the P53 general I/O port function is selected.	Bit 3	0
	WKP2	P52/_WKP2/SEG2 pin function switching: When this bit is 0, the P52 general I/O port function is selected.	Bit 2	0
	WKP1	P51/_WKP1/SEG1 pin function switching: When this bit is 0, the P51 general I/O port function is selected.	Bit 1	0
	WKP0	P50/_WKP0/SEG0 pin function switching: When this bit is 0, the P50 general I/O port function is selected.	Bit 0	0
PUCR5		Port pull-up control register 5: Controls, bit-by-bit, the pull-up MOS of each pin of port 5 that is set as an input port. When PUCR5 = H'00, the pull-up MOS for the P57-P50 pins is off.	H'FFE2	H'00
PDR5		Port data register 5: General I/O port data register for port 5:	H'FFD8	H'00
PCR5		Port control register 5: Selects, bit-by-bit, whether each pin used as a general I/O port for port 5 is an input pin or output pin. When PCR5 = H'FF, the P57-P50 pins function as general output pins.	H'FFE8	H'FF
PDR8		Port data register 8: General I/O port data register for port 8	H'FFDB	H'00

Register name		Description	Address	Setting
PCR8		Port control register 8: Selects, bit-by-bit, whether each pin used as a general I/O port for port 8 is an input pin or output pin. When PCR8 = H'FF, the P87-P80 pins function as general output pins.	H'FFEB	H'FF
IENR1		Interrupt enable register 1: Specifies whether an interrupt request is enabled or disabled.	H'FFF3	-
	IENTA	Timer A interrupt request enable: When this bit is 1, a timer A overflow interrupt request is enabled.	Bit 5	1
IRR1		Interrupt request register 1: When an interrupt request for timer A, IRQ4, IRQ3, IRQAEC, IRQ1, or IRQ0 is issued, the corresponding flag is set to 1.	H'FFF6	-
	IRRTA	Timer A interrupt request flag: This bit is set to 1 when the counter value of timer A overflows (H'FF → H'00). This bit is cleared to 0 when 0 is written to IRRTA.	Bit 7	0/1
IENR2		Interrupt enable register 2: Specifies whether an interrupt request is enabled or disabled.	H'FFF4	-
	IENTC	Timer C interrupt request enable When this bit is 1, a timer C overflow or underflow interrupt request is enabled.	Bit 1	1
IRR2		Interrupt request register 2: When an interrupt request for direct transition, A/D converter, timer G, timer FH, timer FL, timer C, or asynchronous event counter is issued, the corresponding flag is set to 1.	H'FFF7	-
	IRRTC	Timer C interrupt request flag: This bit is set to 1 when the counter value of timer C overflows (H'FF → H'00) or underflows (H'00 → H'FF). This bit is cleared to 0 when 0 is written to IRRTC.	Bit 7	0/1

4. Description of RAM
   Table 5 describes the RAM used in this sample task.

Table 5 Description of RAM

Label name	Description	Address	Module label name
ptr	Location where the address of Dig_0 is stored	H'FB82	tmrc,
dig_0	Stores LED1 display data (1 byte)	H'FB84	main, tmra, tmrc
dig_1	Stores LED2 display data (1 byte)	H'FB85	main, tmra, tmrc
dig_2	Stores LED3 display data (1 byte)	H'FB86	main, tmra, tmrc
dig_3	Stores LED4 display data (1 byte)	H'FB87	main, tmra, tmrc
cnt	8-bit counter for renewing display between LED1-LED4 (1 byte)	H'FB88	main, ttmrc
sdata	Stores a register value for controlling MAX1408 ΣΔ A/D	H'FB89	tmra,ADC_Init,DataIn,DataOut
rdata	Stores the DATA register value that results from MAX1408 ΣΔ A/D conversion.	H'FB8E	tmra, DataIn,

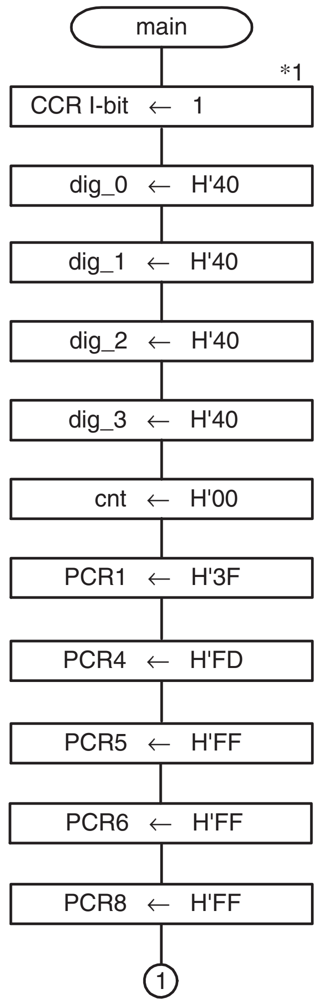
5. Description of Data Tables
   In this sample task, display data for the 7-segment LEDs is stored in ROM as a data table consisting of a 1-dimensional array (data table). Table 6 describes the data table for 7-segment LED display (dsp_data [ ]).

Table 6 Description of Data Table (dsp_data[ ]) for 7-Segment LED Display

Element	Data	Description	Data size	Address
dsp_data[0]	H'3F	Port 5 output data for displaying "0" on a LED	1 byte	H'7B4
dsp_data[1]	H'06	Port 5 output data for displaying "1" on a LED	1 byte	H'7B5
dsp_data[2]	H'5B	Port 5 output data for displaying "2" on a LED	1 byte	H'7B6
dsp_data[3]	H'4F	Port 5 output data for displaying "3" on a LED	1 byte	H'7B7
dsp_data[4]	H'66	Port 5 output data for displaying "4" on a LED	1 byte	H'7B8
dsp_data[5]	H'6D	Port 5 output data for displaying "5" on a LED	1 byte	H'7B9
dsp_data[6]	H'7D	Port 5 output data for displaying "6" on a LED	1 byte	H'7BA
dsp_data[7]	H'27	Port 5 output data for displaying "7" on a LED	1 byte	H'7BB
dsp_data[8]	H'7F	Port 5 output data for displaying "8" on a LED	1 byte	H'7BC
dsp_data[9]	H'6F	Port 5 output data for displaying "9" on a LED	1 byte	H'7BD
dsp_data[10]	H'77	Port 5 output data for displaying "A" on a LED	1 byte	H'7BE
dsp_data[11]	H'7C	Port 5 output data for displaying "B" on a LED	1 byte	H'7BF
dsp_data[12]	H'39	Port 5 output data for displaying "C" on a LED	1 byte	H'7C0
dsp_data[13]	H'5E	Port 5 output data for displaying "D" on a LED	1 byte	H'7C1
dsp_data[14]	H'79	Port 5 output data for displaying "E" on a LED	1 byte	H'7C2
dsp_data[15]	H'71	Port 5 output data for displaying "F" on a LED	1 byte	H'7C3

5. Flowchart

1. Main Routine (main)



Mask interrupt requests by setting interrupt mask bit (I) in CCR to 1.

RAM area initial settings
dig_0 = H'40: LED1 initial display data "-"
dig_1 = H'40: LED2 initial display data "-"
dig_2 = H'40: LED3 initial display data "-"
dig_3 = H'40: LED4 initial display data "-"
cnt = H'00: Initial value of the LED enable counter = 0

Port 1 initial settings
P13/TMIG : P13 output pin (pull-up MOS = OFF)
P14/_IRQ4/_ADTRG : P14 output pin (pull-up MOS = OFF)
P16 : P16 input pin (pull-up MOS = OFF)
P17/_IRQ3/TMIF : P17 input pin (pull-up MOS = OFF)

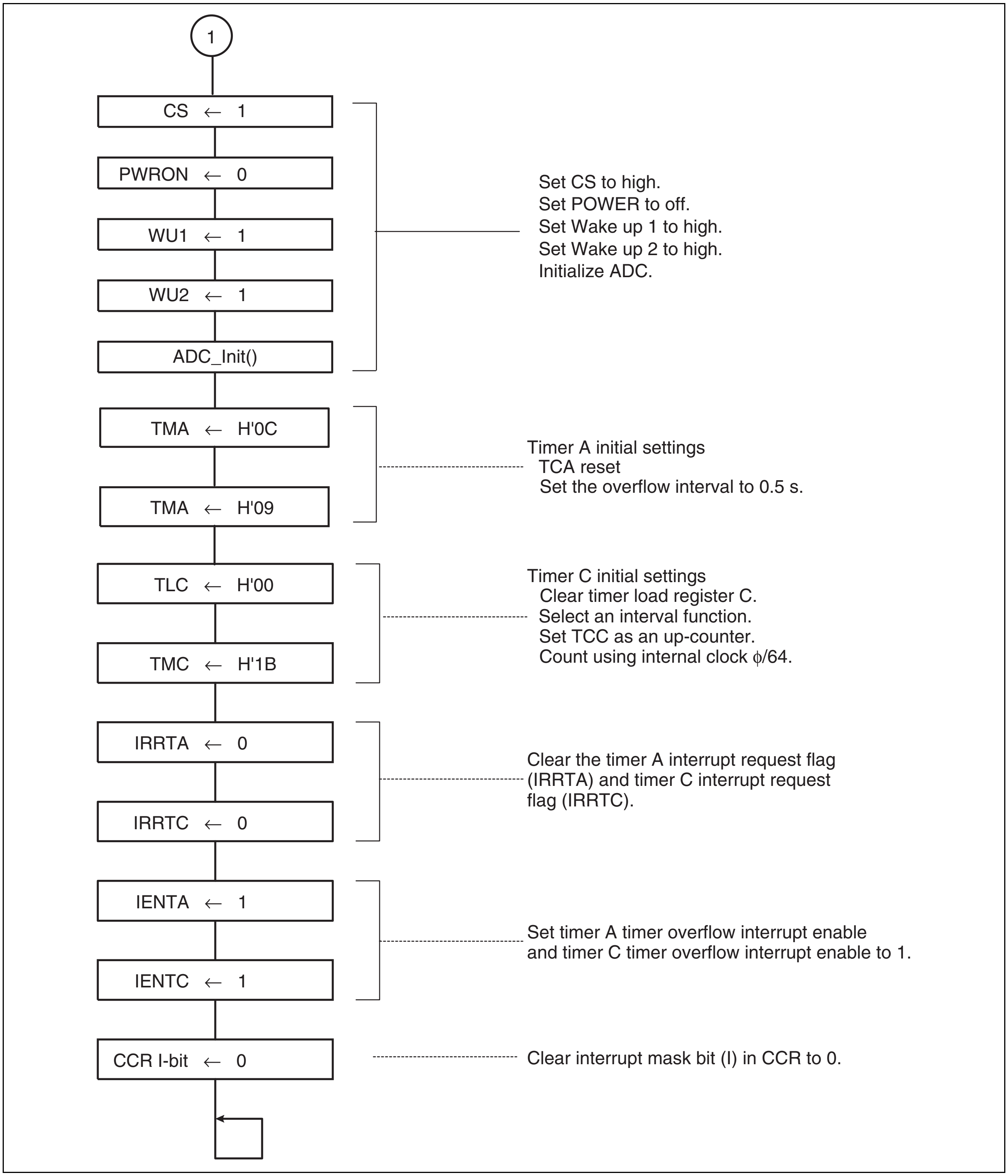
Port 4 initial settings
P40/SCK32 : P40 output pin
P41/RXD32 : P41 input pin
P42/TXD32 : P42 output pin

Port 5 initial settings
P50/_WKP0/SEG1 : P50 output pin (output data = 0)
P51/_WKP1/SEG2 : P51 output pin (output data = 0)
P52/_WKP2/SEG3 : P52 output pin (output data = 0)
P53/_WKP3/SEG4 : P53 output pin (output data = 0)
P54/_WKP4/SEG5 : P54 output pin (output data = 0)
P55/_WKP5/SEG6 : P55 output pin (output data = 0)
P56/_WKP6/SEG7 : P56 output pin (output data = 0)
P57/_WKP7/SEG8 : P57 output pin (output data = 0)

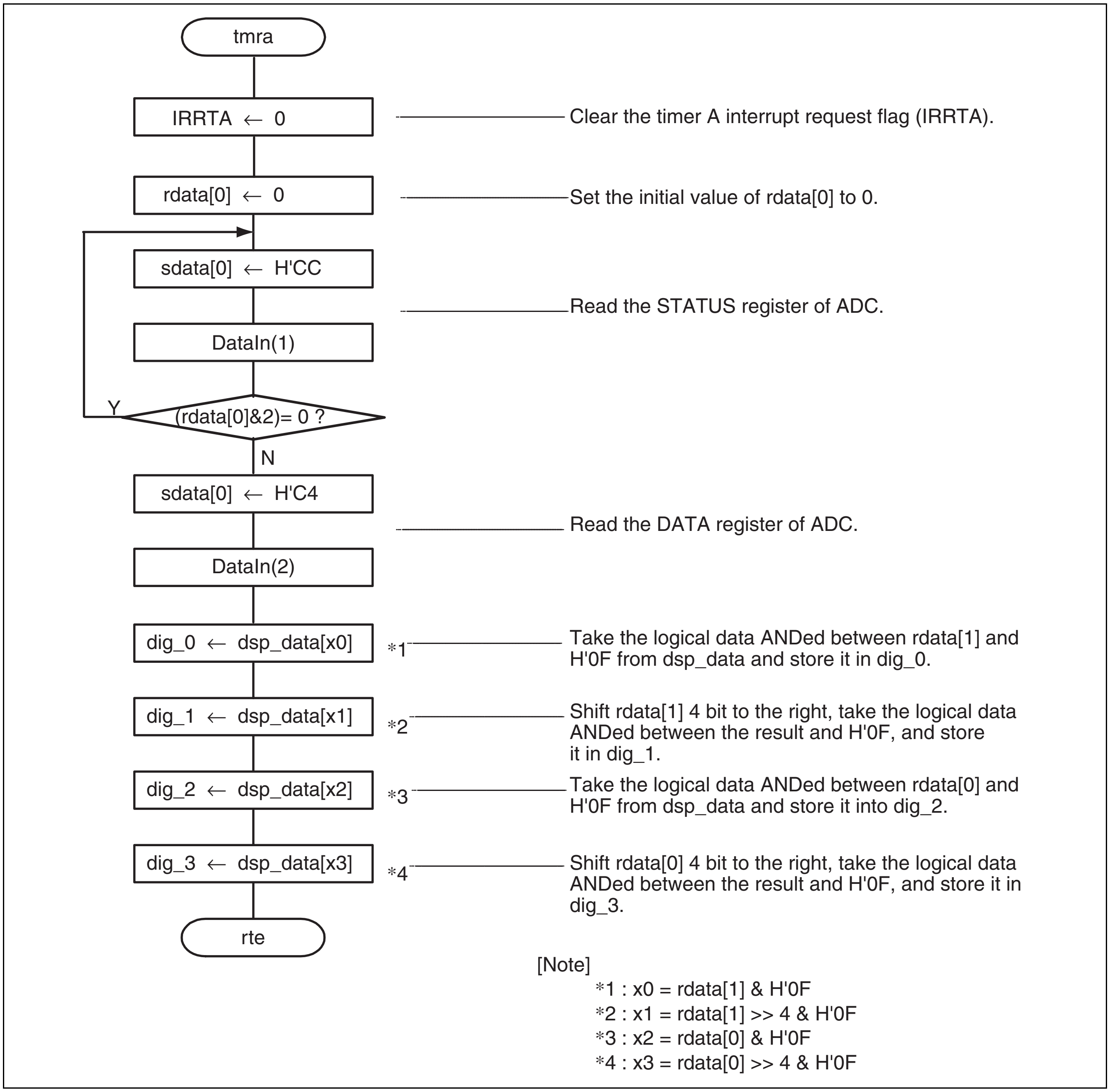
Port 6 initial settings
P60/SEG9 : P60 output pin
P61/SEG10 : P61 output pin
P62/SEG11 : P62 output pin
P63/SEG12 : P63 output pin
P64/SEG13 : P64 output pin
P65/SEG14 : P65 output pin
P66/SEG15 : P66 output pin
P67/SEG16 : P67 output pin

Port 8 initial settings
P80/SEG25 : P80 output pin
P81/SEG26 : P81 output pin
P82/SEG27 : P82 output pin
P83/SEG28 : P83 output pin
P84/SEG29 : P84 output pin
P85/SEG30 : P85 output pin
P86/SEG31 : P86 output pin
P87/SEG32 : P87 output pin

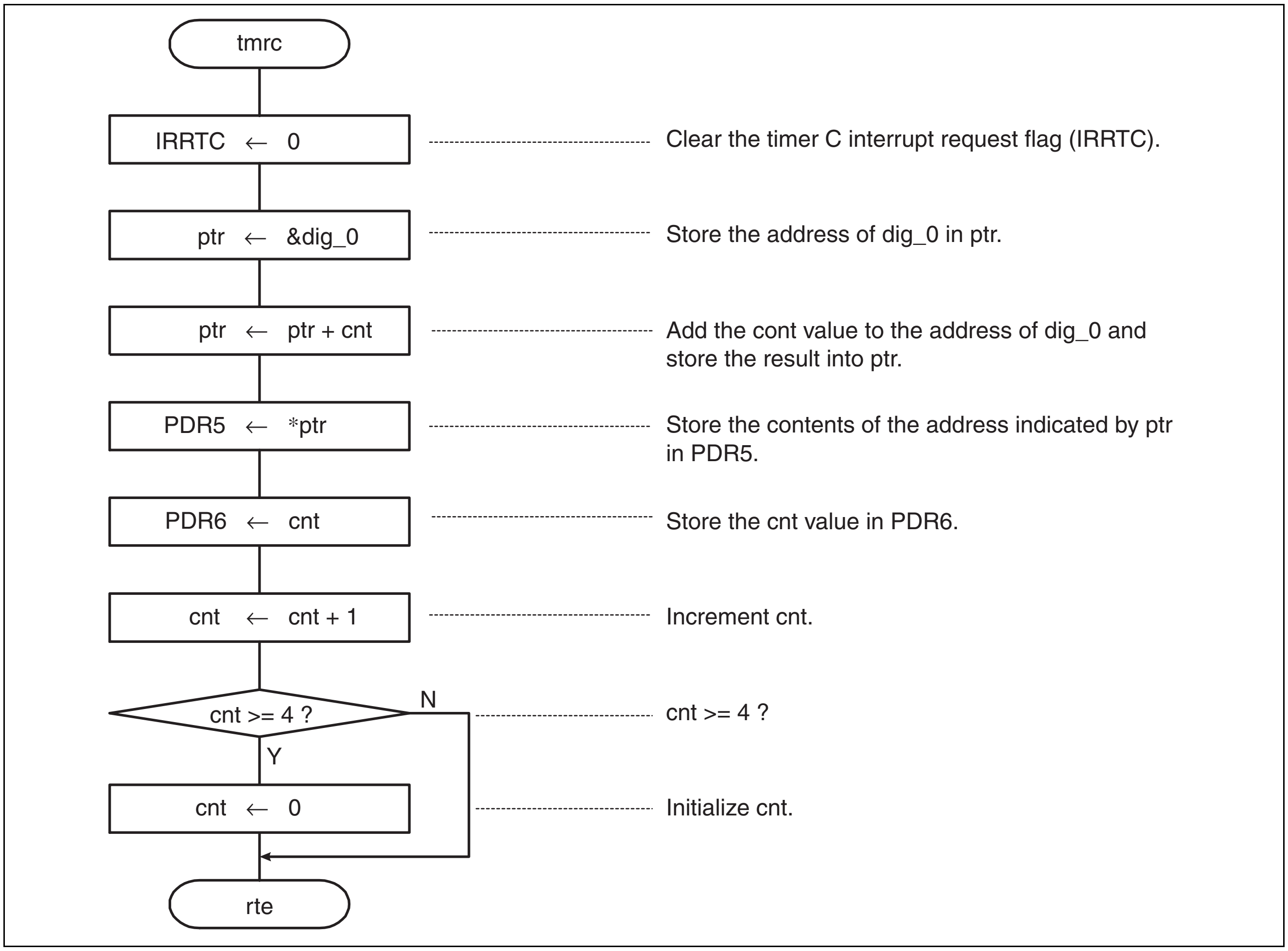
[Note]
*1: In this sample task, INIT.SRC (assembly language) is used to set the stack pointers.


1
CS ← 1
PWRON ← 0
WU1 ← 1
WU2 ← 1
ADC_Init()
Set CS to high.
Set POWER to off.
Set Wake up 1 to high.
Set Wake up 2 to high.
Initialize ADC.
TMA ← H'0C
TMA ← H'09
Timer A initial settings
TCA reset
Set the overflow interval to 0.5 s.
TLC ← H'00
TMC ← H'1B
Timer C initial settings
Clear timer load register C.
Select an interval function.
Set TCC as an up-counter.
Count using internal clock ϕ/64.
IRRTA ← 0
IRRTC ← 0
Clear the timer A interrupt request flag (IRRTA) and timer C interrupt request flag (IRRTC).
IENTA ← 1
IENTC ← 1
Set timer A timer overflow interrupt enable and timer C timer overflow interrupt enable to 1.
CCR I-bit ← 0
Clear interrupt mask bit (I) in CCR to 0.

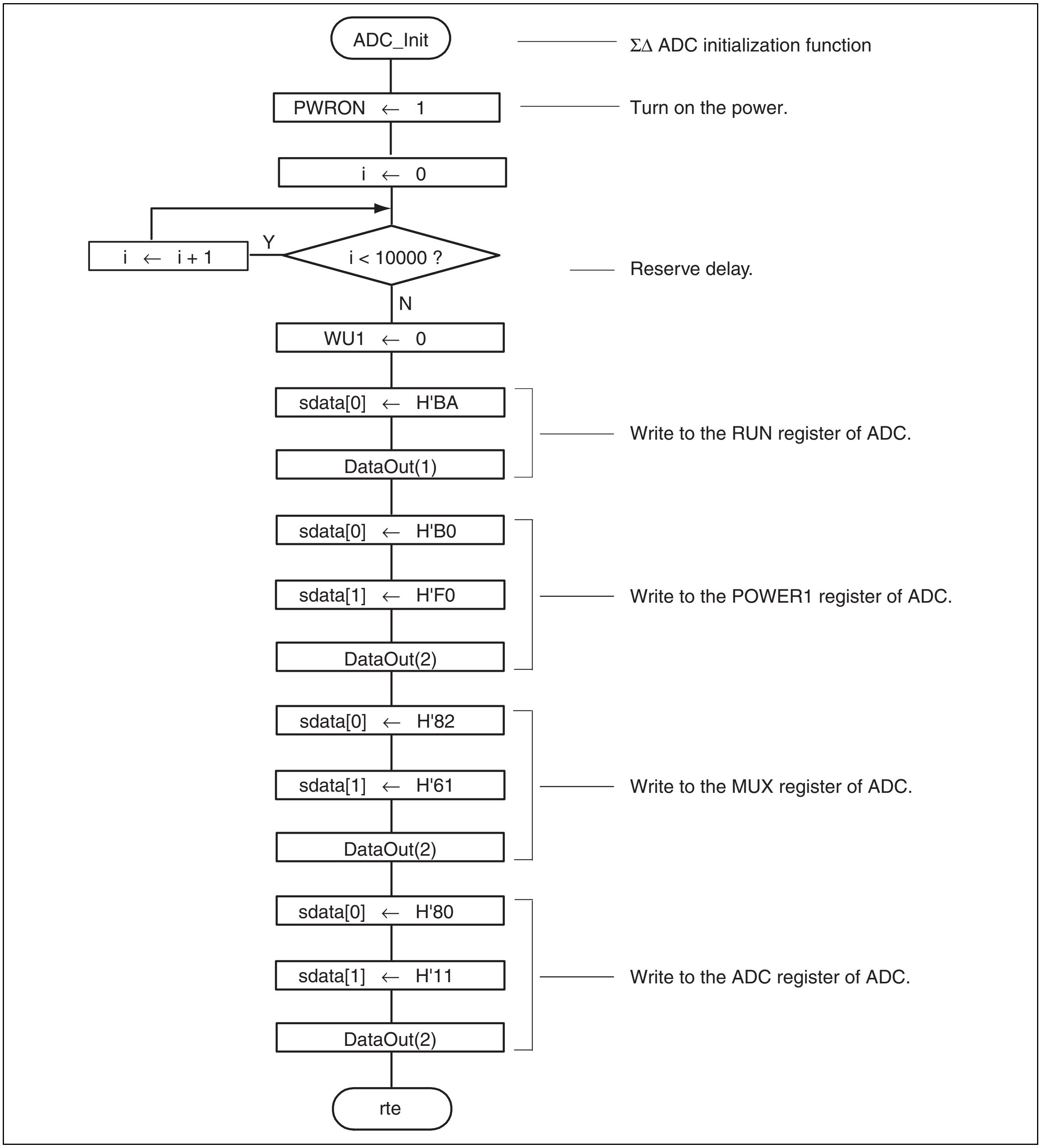
2. Timer A Interrupt Processing Routine (tmra)



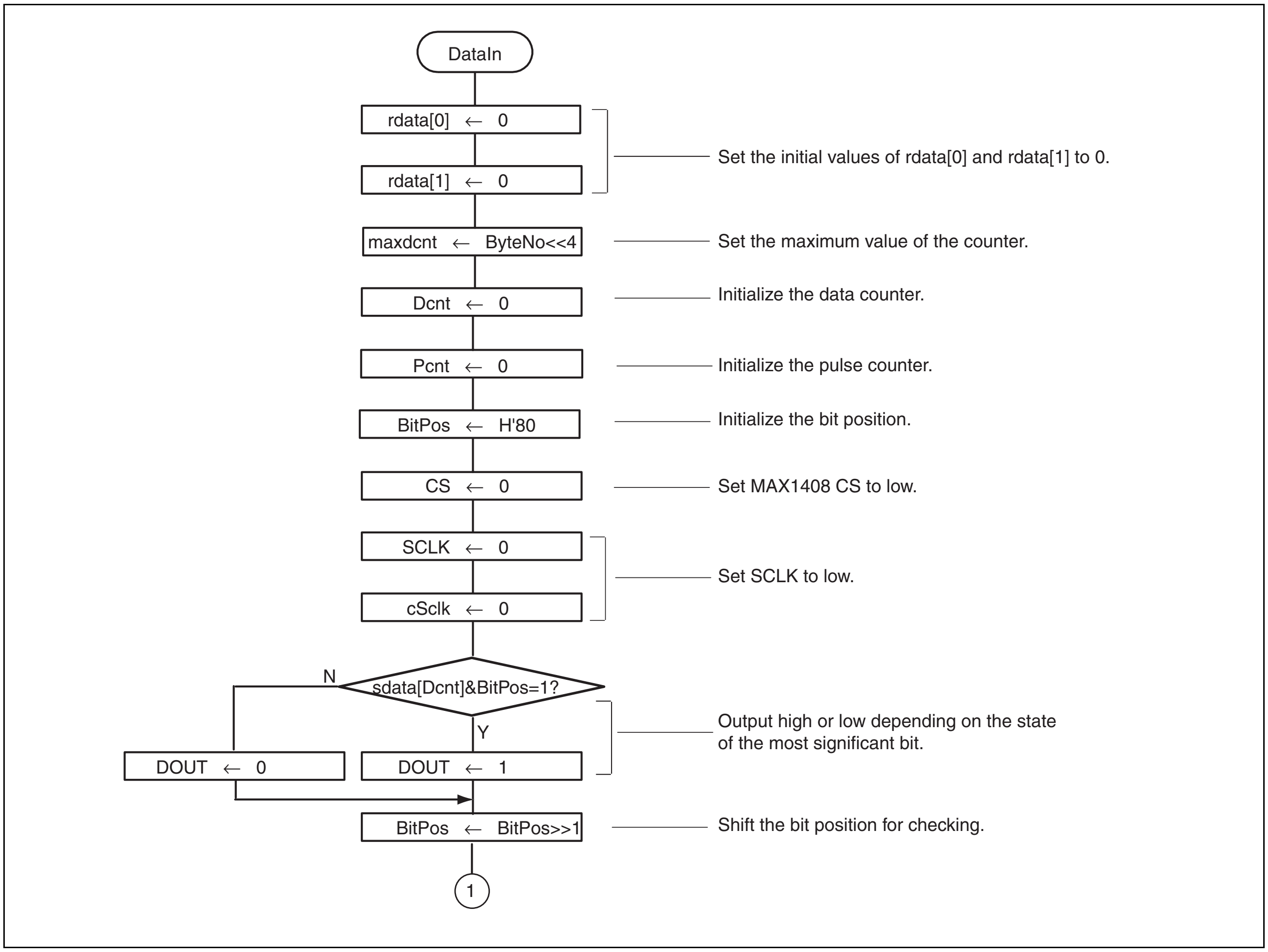
3. Timer C Interrupt Processing Routine (tmrc)

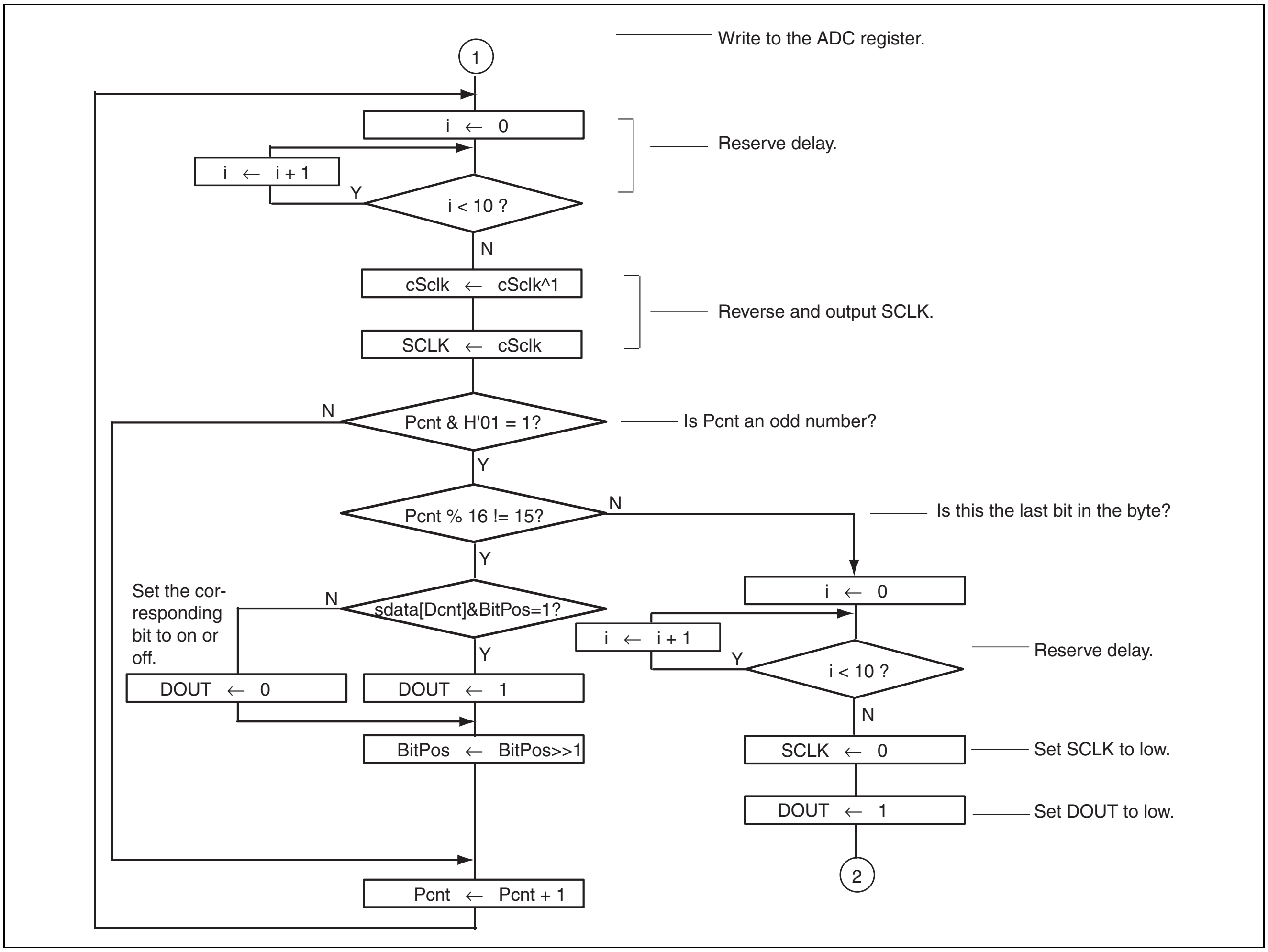


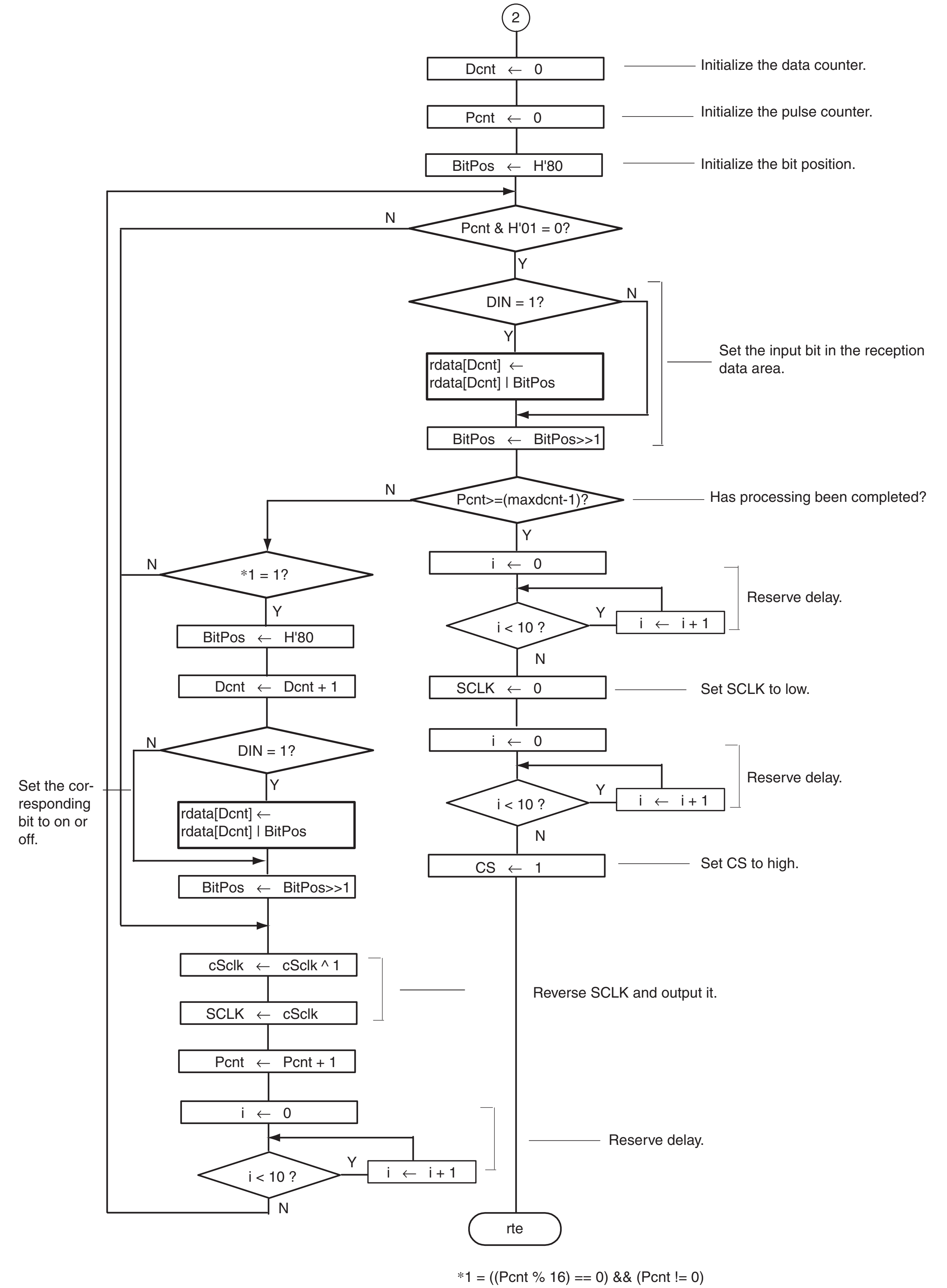
4. ADC_Init() Processing Routine



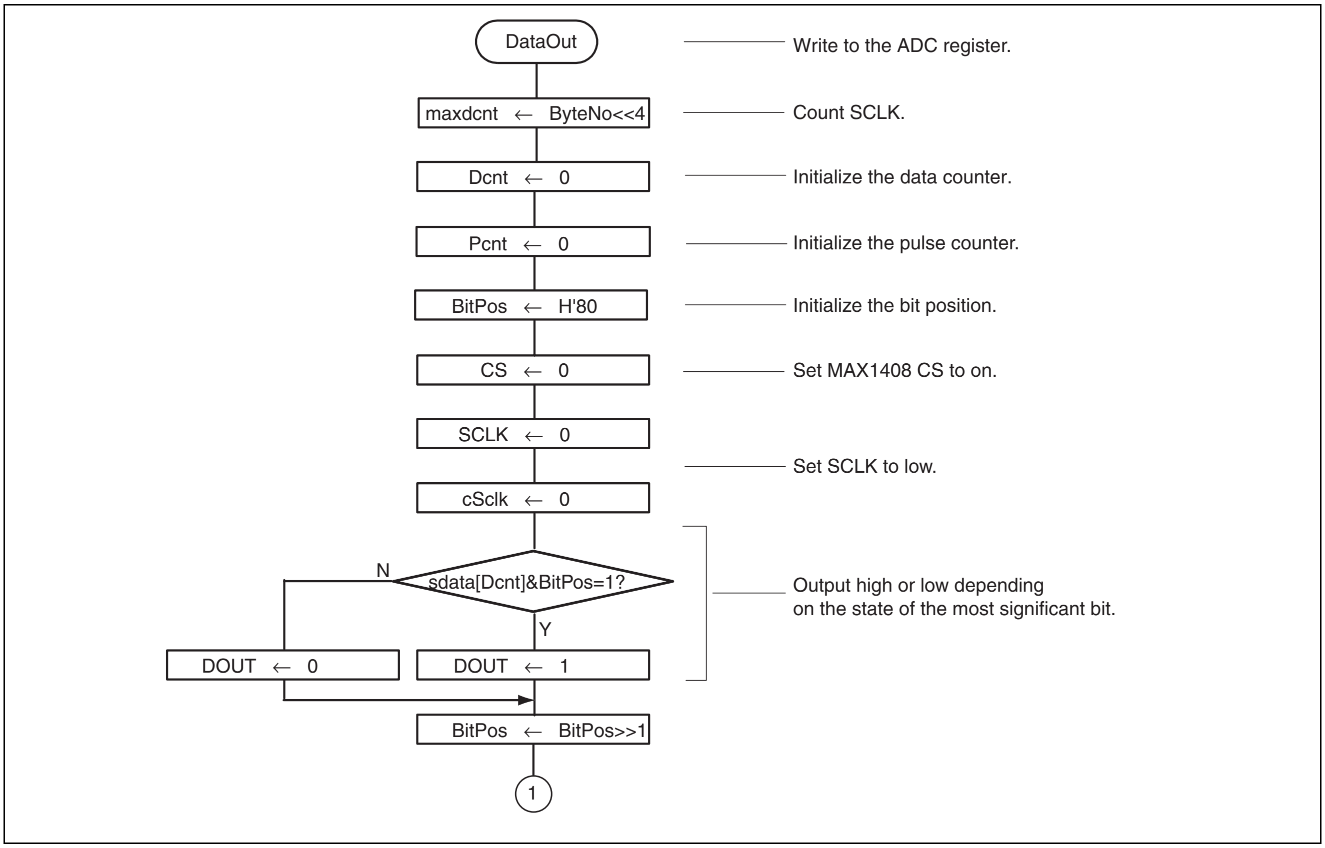
5. DataIn() Processing Routine

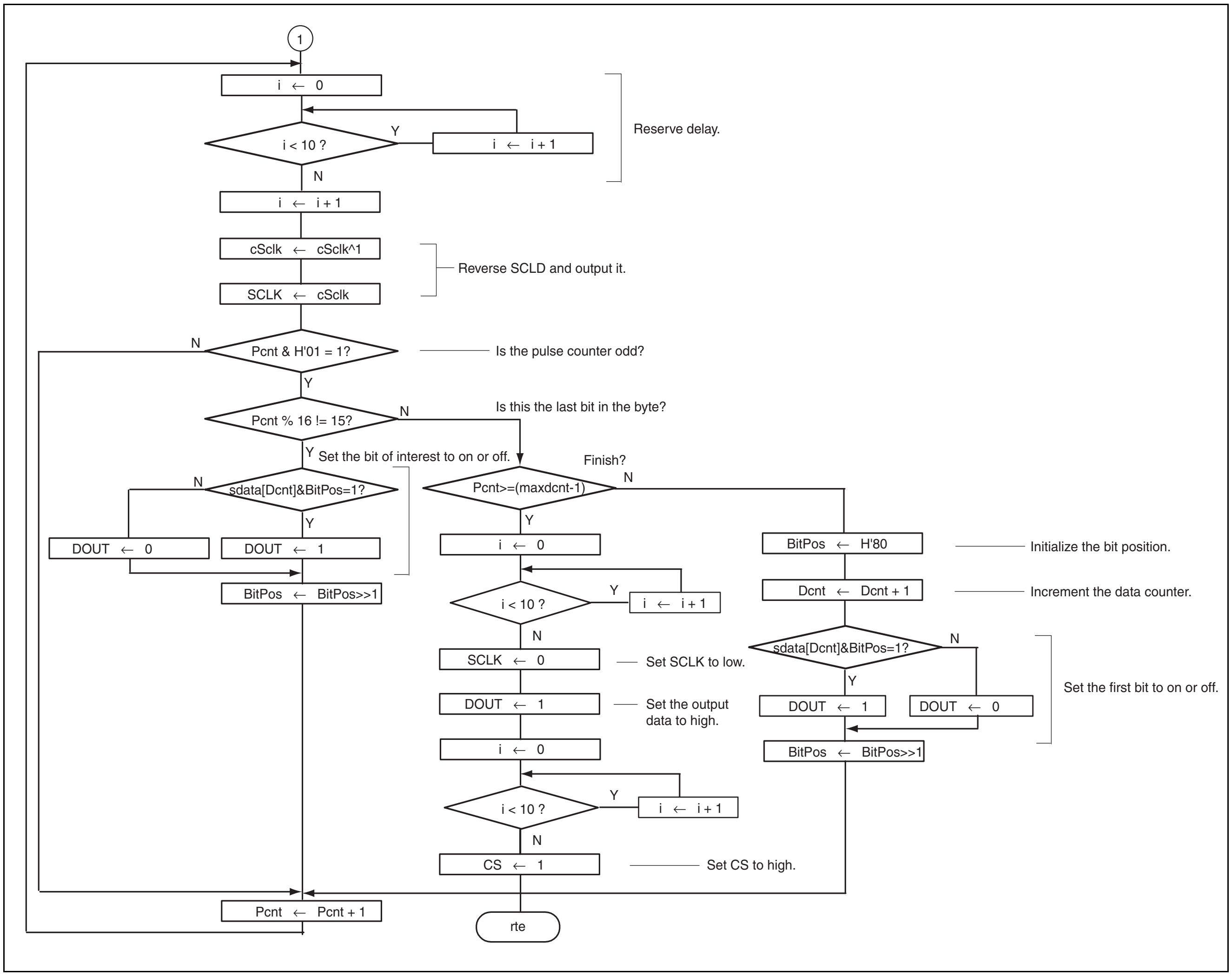



1
Write to the ADC register.
i ← 0
i ← i + 1
Y
i < 10 ?
Reserve delay.
N
cSclk ← cSclk^1
SCLK ← cSclk
Reverse and output SCLK.
N
Pcnt & H'01 = 1?
Is Pcnt an odd number?
Y
Pcnt % 16 != 15?
N
Is this the last bit in the byte?
Y
Set the cor-
responding
bit to on or
off.
N
sdata[Dcnt]&BitPos=1?
Y
DOUT ← 0
DOUT ← 1
BitPos ← BitPos>>1
Pcnt ← Pcnt + 1
i ← 0
i ← i + 1
Y
i < 10 ?
Reserve delay.
N
SCLK ← 0
Set SCLK to low.
DOUT ← 1
Set DOUT to low.
2


2
Dcnt ← 0
Initialize the data counter.
Pcnt ← 0
Initialize the pulse counter.
BitPos ← H'80
Initialize the bit position.
Pcnt & H'01 = 0?
N
Y
DIN = 1?
N
Y
rdata[Dcnt] ← rdata[Dcnt] | BitPos
Set the input bit in the reception data area.
BitPos ← BitPos>>1
Pcnt>=(maxdcnt-1)?
Has processing been completed?
N
Y
*1 = 1?
N
Y
BitPos ← H'80
Dcnt ← Dcnt + 1
DIN = 1?
N
Y
rdata[Dcnt] ← rdata[Dcnt] | BitPos
Set the corresponding bit to on or off.
BitPos ← BitPos>>1
i ← 0
i < 10 ?
Y
N
i ← i + 1
Reserve delay.
SCLK ← 0
Set SCLK to low.
i ← 0
i < 10 ?
Y
N
i ← i + 1
Reserve delay.
CS ← 1
Set CS to high.
cSclk ← cSclk ^ 1
SCLK ← cSclk
Reverse SCLK and output it.
Pcnt ← Pcnt + 1
i ← 0
i < 10 ?
Y
N
i ← i + 1
Reserve delay.
rte
*1 = ((Pcnt % 16) == 0) && (Pcnt != 0)

6. DataOut() Processing Routine




1
i ← 0
i < 10 ?
Y
N
i ← i + 1
Reserve delay.
i ← i + 1
cSclk ← cSclk^1
SCLK ← cSclk
Reverse SCLD and output it.
Pcnt & H'01 = 1?
Is the pulse counter odd?
Pcnt % 16 != 15?
Is this the last bit in the byte?
Set the bit of interest to on or off.
sdata[Dcnt]&BitPos=1?
DOUT ← 0
DOUT ← 1
BitPos ← BitPos>>1
Finish?
Pcnt>=(maxdcnt-1)
i ← 0
i < 10 ?
i ← i + 1
SCLK ← 0
Set SCLK to low.
DOUT ← 1
Set the output data to high.
i ← 0
i < 10 ?
i ← i + 1
CS ← 1
Set CS to high.
BitPos ← H'80
Initialize the bit position.
Dcnt ← Dcnt + 1
Increment the data counter.
sdata[Dcnt]&BitPos=1?
DOUT ← 1
DOUT ← 0
BitPos ← BitPos>>1
Set the first bit to on or off.
Pcnt ← Pcnt + 1
rte

6. Program Listing

```
/* Super Low Power Series -H8/38024- Application note */
/* Application example */
/* Example of connecting to ΣΔ ADC */

#include <machine.h>

/* Symbol definition */
struct BIT {
    unsigned char b7:1;                              /* bit 7 */
    unsigned char b6:1;                              /* bit 6 */
    unsigned char b5:1;                              /* bit 5 */
    unsigned char b4:1;                              /* bit 4 */
    unsigned char b3:1;                              /* bit 3 */
    unsigned char b2:1;                              /* bit 2 */
    unsigned char b1:1;                              /* bit 1 */
    unsigned char b0:1;                              /* bit 0 */
};

#define PDR1   *(volatile unsigned char *)0xFFD4     /* Port data register 1 */
#define PCR1   *(volatile unsigned char *)0xFFE4     /* Port control register 1 */
#define PUCR1  *(volatile unsigned char *)0xFFE0     /* Port pull-up control register 1 */
#define PMR1   *(volatile unsigned char *)0xFFC8     /* Port mode register 1 */
#define PMR2   *(volatile unsigned char *)0xFFC9     /* Port mode register 2 */
#define PDR1_BIT (*(struct BIT *)0xFFD4)
#define WU1    PDR1_BIT.b3                           /* MAX1408 WU1 */
#define WU2    PDR1_BIT.b4                           /* MAX1408 WU2 */

#define PDR4   *(volatile unsigned char *)0xFFD7     /* Port data register 4 */
#define PCR4   *(volatile unsigned char *)0xFFE7     /* Port control register 4 */
#define PDR4_BIT (*(struct BIT *)0xFFD7)
#define SCLK   PDR4_BIT.b0                           /* MAX1408 SCLK */
#define DOUT   PDR4_BIT.b2                           /* MAX1408 DOUT */
#define DIN    PDR4_BIT.b1                           /* MAX1408 DIN */

#define PMR5   *(volatile unsigned char *)0xFFCC     /* Port mode register 5 */
#define PUCR5  *(volatile unsigned char *)0xFFE2     /* Port pull-up control register 5 */
#define PDR5   *(volatile unsigned char *)0xFFD8     /* Port data register 5 */
#define PCR5   *(volatile unsigned char *)0xFFE8     /* Port control register 5 */

#define PUCR6  *(volatile unsigned char *)0xFFE3     /* Port pull-up control register 6 */
#define PDR6   *(volatile unsigned char *)0xFFD9     /* Port data register 6 */
#define PCR6   *(volatile unsigned char *)0xFFE9     /* Port control register 6 */

#define PDR8   *(volatile unsigned char *)0xFFDB     /* Port data register 8 */
#define PCR8   *(volatile unsigned char *)0xFFEB     /* Port control register 8 */
#define PDR8_BIT (*(struct BIT *)0xFFDB)
#define CS     PDR8_BIT.b0                           /* MAX1408 #CS */
#define PWRON  PDR8_BIT.b1                           /* MAX1408 POWER */

#define TMA    *(volatile unsigned char *)0xFFB0     /* Timer mode register A */
```

```
#define CKSTPR1    *(volatile unsigned char *)0xFFFA /* Clock stop register 1 */

#define TMC    *(volatile unsigned char *)0xFFB4     /* Timer mode register C */
#define TLC    *(volatile unsigned char *)0xFFB5     /* Timer Load register C */

#define IRR1   *(volatile unsigned char *)0xFFF6     /* Interrupt request register 1 */
#define IRR1_BIT (*(struct BIT *)0xFFF6)
#define IRRTA  IRR1_BIT.b7                           /* Timer A interrupt request flag */
#define IENR1  *(volatile unsigned char *)0xFFF3     /* Interrupt enable register 1 */
#define IENR1_BIT (*(struct BIT *)0xFFF3)
#define IENTA  IENR1_BIT.b7                          /* Timer A interrupt enable */

#define IRR2   *(volatile unsigned char *)0xFFF7     /* Interrupt request register 2 */
#define IRR2_BIT (*(struct BIT *)0xFFF7)
#define IRRTC  IRR2_BIT.b1                           /* Timer C interrupt request flag */
#define IENR2  *(volatile unsigned char *)0xFFF4     /* Interrupt enable register 2 */
#define IENR2_BIT (*(struct BIT *)0xFFF4)
#define IENTC  IENR2_BIT.b1                          /* Timer C interrupt enable */

#pragma interrupt (tmra)
#pragma interrupt (tmrc)

/* Function definition */
extern void INIT(void);                              /* Stack pointer set */
void main(void);                                     /* main routine */
void tmra(void);                                     /* Timer A interrupt routine */
void tmrc(void);                                     /* Timer C interrupt routine */
void ADC_Init(void);                                 /* ADC initialize   */
void DataIn(int ByteNo);                             /* ADC read */
void DataOut(int ByteNo);                            /* ADC write */

/* Data table */
const unsigned char dsp_data[16] =
{
    0x3f,                                            /* LED display data = "0" */
    0x06,                                            /* LED display data = "1" */
    0x5b,                                            /* LED display data = "2" */
    0x4f,                                            /* LED display data = "3" */
    0x66,                                            /* LED display data = "4" */
    0x6d,                                            /* LED display data = "5" */
    0x7d,                                            /* LED display data = "6" */
    0x27,                                            /* LED display data = "7" */
    0x7f,                                            /* LED display data = "8" */
    0x6f,                                            /* LED display data = "9" */
    0x77,                                            /* LED display data = "A" */
    0x7c,                                            /* LED display data = "b" */
    0x39,                                            /* LED display data = "C" */
    0x5e,                                            /* LED display data = "d" */
    0x79,                                            /* LED display data = "E" */
    0x71                                             /* LED display data = "F" */
};

/* RAM define */
unsigned char dig_0;                                 /* Dig-0 LED display data store */
```

```
unsigned char dig_1;                            /* Dig-1 LED display data store */
unsigned char dig_2;                            /* Dig-2 LED display data store */
unsigned char dig_3;                            /* Dig-3 LED display data store */
unsigned char cnt;                              /* LED enable counter */
unsigned int temp;
unsigned char *ptr;                             /* Pointer set */
volatile unsigned char   sdata[5];              /* send data area          */
volatile unsigned char   rdata[5];              /* receive data area */

/* Vector address */
#pragma section V1                              /* Vector section set */
void (*const VEC_TBL1[])(void) = {
    INIT                                        /* H'0000 Reset vector */
};
#pragma section V2                              /* Vector section set */
void (*const VEC_TBL2[])(void) = {
    tmra                                        /* H'0016 Timer A interrupt vector */
};
#pragma section V3                              /* Vector section set */
void (*const VEC_TBL3[])(void) = {
    tmrc                                        /* H'001a Timer C interrupt vector */
};
#pragma section                                 /* P */

/************************************************************/
/* Main program                                             */
/************************************************************/
void main(void)
{
    set_imask_ccr(1);                           /* CCR I-bit = 1 */

    dig_0 = 0x40;                               /* Used RAM area initialize */
    dig_1 = 0x40;                               /* Used RAM area initialize */
    dig_2 = 0x40;                               /* Used RAM area initialize */
    dig_3 = 0x40;                               /* Used RAM area initialize */
    cnt   = 0x00;                               /* Used RAM area initialize */

    PCR1  = 0x3f;                               /* Port 1 initialize */
    PCR4  = 0xfd;                               /* Port 4 initialize */
    PCR5  = 0xff;                               /* Port 5 initialize */
    PCR6  = 0xff;                               /* Port 6 initialize */
    PCR8  = 0xff;                               /* Port 8 initialize */

    CS = 1;                                     /* CS High    */
    PWRON = 0;                                  /* POWER OFF  */
    WU2 = 1;                                    /* Wake up OFF       */
    WU2 = 1;                                    /* Wake up OFF       */

    ADC_Init();                                 /* ADC initialize    */

    TMA   = 0x0c;                               /* Clear Timer Counter A to 0 */
    TMA   = 0x09;                               /* Timer A initialize */
    TLC   = 0x00;                               /* Clear Timer Load register C to 0 */
    TMC   = 0x1b;                               /* Timer C initialize */
```

```
    IRRTA = 0;                                      /* Clear IRRTA to 0 */
    IRRTC = 0;                                      /* Clear IRRTC to 0 */
    IENTA = 1;                                      /* Timer A interrupt enable */
    IENTC = 1;                                      /* Timer C interrupt enable */

    set_imask_ccr(0);                               /* CCR I-bit = 0 */

    while(1);
}

/*************************************************************/
/* Timer A Interrupt                                         */
/*************************************************************/
void tmra(void)
{
    IRRTA = 0;                                      /* Clear IRRTA to 0 */

    rdata[0] = 0x00;
    do {
        sdata[0] = 0xcc;                            /* Read to Status Register */
        DataIn(1);
    } while((rdata[0] & 0x02) == 0);
    sdata[0] = 0xc4;                                /* Read DATA Register value */
    DataIn(2);

    dig_0 = dsp_data[rdata[1] & 0x0f];              /* Dig-0 LED display data set */
    dig_1 = dsp_data[rdata[1] >> 4 & 0x0f];         /* Dig-1 LED display data set */
    dig_2 = dsp_data[rdata[0] & 0x0f];              /* Dig-2 LED display data set */
    dig_3 = dsp_data[rdata[0] >> 4 & 0x0f];         /* Dig-3 LED display data set */
}

/*************************************************************/
/* Timer C Interrupt                                         */
/*************************************************************/
void tmrc(void)
{
    IRRTC = 0;                                      /* Clear IRRTC to 0 */

    ptr = &dig_0;                                   /* LED display data store address set */
    ptr += cnt;                                     /* LED display data read */
    PDR5 = *ptr;                                    /* LED display data output */
    PDR6 = cnt;                                     /* LED enable data output */

    cnt++;                                          /* "cnt" increment */
    if (cnt >= 4){                                  /* 4 times end ? */
        cnt = 0;                                    /* "cnt" initialize */
    }
}

/*************************************************************/
/* ADC initialize function                                   */
/*************************************************************/
void ADC_Init(void)
```

```
{
    long i;

    PWRON = 1;                                          /* POWER ON  */

    for(i=0;i<10000;i++);                               /* delay     */
    WU1 = 0;

    sdata[0] = 0xba;                                    /* Write RUN Register */
    DataOut(1);

    sdata[0] = 0xb0;                                    /* Write POWER1 Register */
    sdata[1] = 0xf0;
    DataOut(2);

    sdata[0] = 0x82;                                    /* Write MUX Register */
    sdata[1] = 0x61;
    DataOut(2);

    sdata[0] = 0x80;                                    /* Write ADC Register */
    sdata[1] = 0x11;
    DataOut(2);
}

/*************************************************************/
/* ADC register write and read function                      */
/*************************************************************/
void DataIn(int ByteNo)
{
    int cSclk, Dcnt, maxdcnt, i, Pcnt;
    unsigned  char BitPos;

    rdata[0] = rdata[1] = 0x00;                         /* data area initialize    */

    maxdcnt = (ByteNo << 4);
    Dcnt = Pcnt = 0;                                    /* counter initialize      */
    BitPos = 0x80;                                      /* bit position initialize */
    CS = 0;                                             /* CS ON     */
    SCLK = cSclk = 0;                                   /* SCLK Low  */
    if(sdata[Dcnt]  & BitPos)                           /* output data     */
        DOUT = 1;
    else
        DOUT = 0;
    BitPos >>= 1;                                       /* next bit position*/

    /* write register  */
    while(1){
        for(i=0;i<10;i++);                              /* delay     */
        cSclk ^= 1;
        SCLK = cSclk;                                   /* output reverse SCLK     */
        if(Pcnt & 0x01) {
            if((Pcnt % 16) != 15){                      /* last bit ?*/
                if(sdata[Dcnt] & BitPos)                /* output data      */
                    DOUT = 1;
```

```
                else
                    DOUT = 0;
                BitPos >>= 1;                           /* next bit position*/
            } else {
                for(i=0;i<10;i++);                      /* delay     */
                SCLK = 0;                               /* SCLK Low  */
                DOUT = 1;                               /* DOUT Low  */
                break;
            }
        }
        Pcnt++;                                         /* pulse count increment  */
    }

    /* read register   */
    Dcnt = Pcnt = 0;                                    /* SCLK count */
    BitPos = 0x80;                                      /* bitb position initialize       */
    while(1){
        if((Pcnt & 0x01) == 0x00) {
                if(DIN)
                    rdata[Dcnt] |= BitPos;              /* input data */
                BitPos >>= 1;                           /* next bit position*/
                if(Pcnt >= (maxdcnt - 1)) {             /* end ?     */
                    for(i=0;i<10;i++);                  /* delay     */
                    SCLK = 0;                           /* SCLK Low  */
                    for(i=0;i<10;i++);                  /* delay     */
                    CS = 1;                             /* CS High   */
                    return;
                } else if(((Pcnt % 16) == 0) && (Pcnt != 0)){/* next data */
                    BitPos = 0x80;                      /* bitb position initialize       */
                    Dcnt++;                             /* data count increment    */
                    if(DIN)                             /* input data */
                        rdata[Dcnt] |= BitPos;
                    BitPos >>= 1;                       /* next bit position*/
                }
        }
        cSclk ^= 1;
        SCLK = cSclk;                                   /* reverse SCLK      */
        Pcnt++;                                         /* pulse count increment   */
        for(i=0;i<10;i++);                              /* delay     */
    }
}

/************************************************************/
/* ADC register write function                              */
/************************************************************/
void DataOut(int ByteNo)
{
    int cSclk, Dcnt, maxdcnt, i, Pcnt;
    unsigned  char BitPos;

    maxdcnt = (ByteNo << 4);
    Dcnt = Pcnt = 0;                                    /* counter initialize      */
    BitPos = 0x80;                                      /* bit position initialize */
    CS = 0;                                             /* CS ON     */
```

```
    SCLK = cSclk = 0;                                   /* SCLK Low  */
    if(sdata[Dcnt]  & BitPos)                           /* output data      */
        DOUT = 1;
    else
        DOUT = 0;
    BitPos >>= 1;                                       /* next bit position*/

    while(1){
        for(i=0;i<10;i++);                              /* delay     */
        cSclk ^= 1;
        SCLK = cSclk;                                   /* reverse SCLK     */
        if(Pcnt & 0x01) {
            if((Pcnt % 16) != 15){                      /* not last bit ?   */
                if(sdata[Dcnt] & BitPos)                /* output data      */
                    DOUT = 1;
                else
                    DOUT = 0;
                BitPos >>= 1;                           /* next bit position*/
            } else {
                if(Pcnt >= (maxdcnt - 1)) {             /* end ?     */
                    for(i=0;i<10;i++);                  /* delay     */
                    SCLK = 0;                           /* SCLK Low  */
                    DOUT = 1;                           /* output High     */
                    for(i=0;i<10;i++);                  /* delay     */
                    CS = 1;                             /* CS OFF    */
                    return;
                } else {
                    BitPos = 0x80;                      /* bit position initialize */
                    Dcnt++;                             /* data count increment    */
                    if(sdata[Dcnt] & BitPos)            /* output data      */
                        DOUT = 1;
                    else
                        DOUT = 0;
                    BitPos >>= 1;                       /* next bit position*/
                }
            }
        }
        Pcnt++;                                         /* pulse count increment   */
    }
}
```

Revision Record

Rev.	Date	Description	
		Page	Summary
1.00	Sept.19.03	—	First edition issued