To our customers,

Old Company Name in Catalogs and Other Documents

On April 1st, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

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

April 1st, 2010
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (http://www.renesas.com)

Send any inquiries to http://www.renesas.com/inquiry.

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

   “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.

   “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

H8/300L SLP Series

Using an Infrared Remote Controller in Transmission and Reception

Introduction

Timers C and F are used for transmission by the infrared remote controller.

Target Device

H8/38024

Contents

1. Specifications

1. Timers C and F are used for transmission by the infrared remote controller.
2. Timer G is used for reception by the infrared remote controller.
3. The sample task is such that the key switches (0 to 9 and A to F) connected to the transmit side are pressed to display their key numbers on the LCD on the receive side.
4. The infrared remote controller operates in half-duplex communications mode.
5. Figure 1.1 shows the connection of the infrared remote controller.

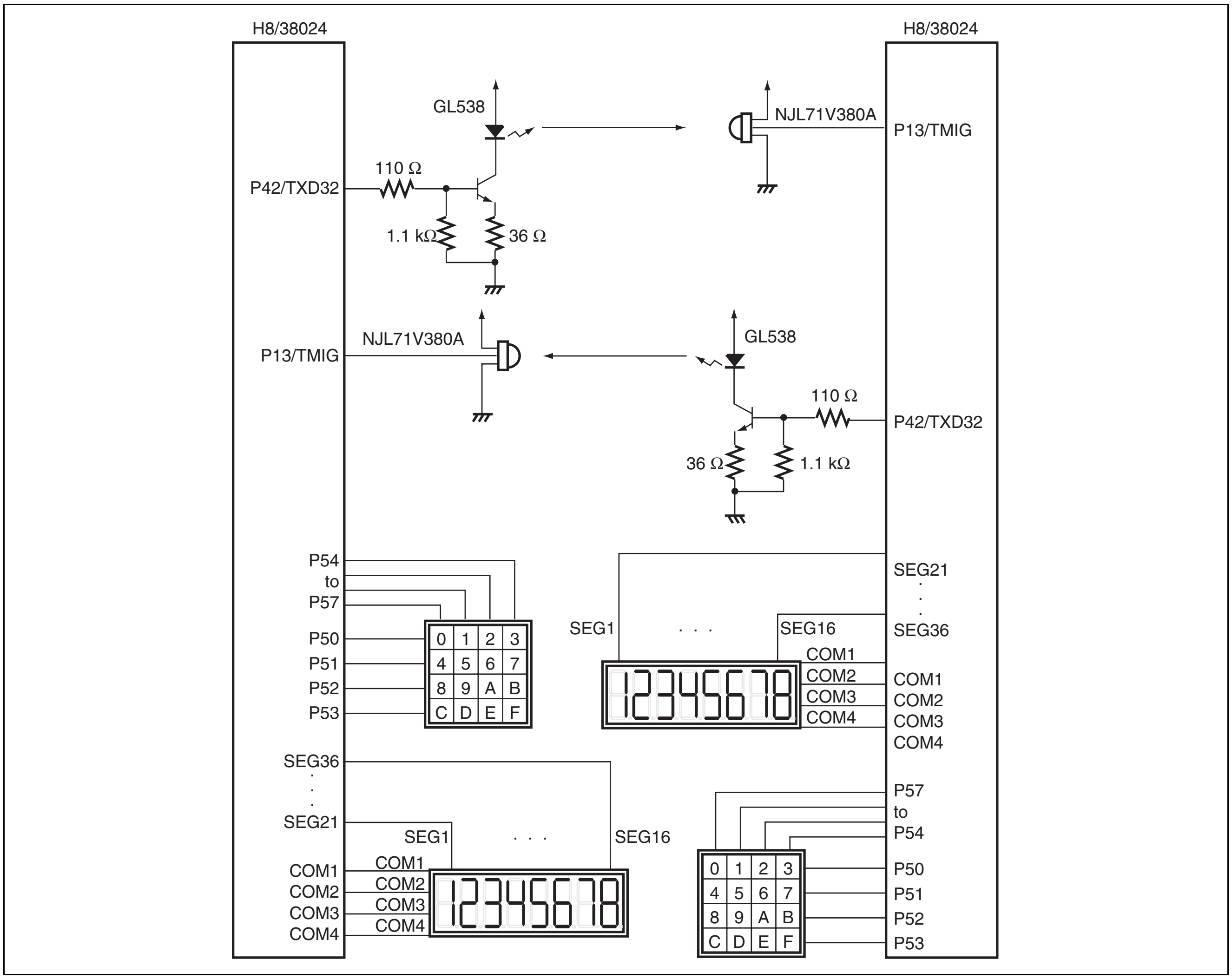


Figure 1.1 Example of Connection of Infrared Remote Controller

2. Concepts

1. Figure 2.1 shows examples of signals for infrared remote control processing used in the sample task. The infrared light reception element used for the sample task outputs the low level when it receives a carrier frequency of 38 kHz.

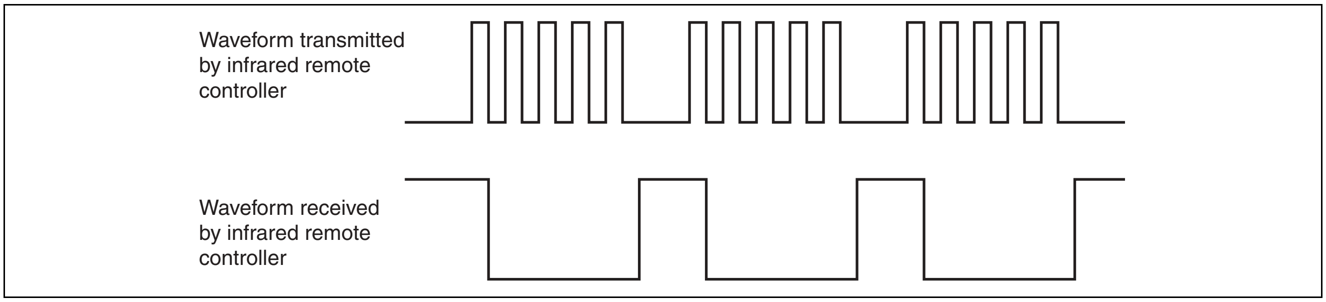

Waveform transmitted by infrared remote controller
Waveform received by infrared remote controller

Figure 2.1 Examples of Signals Used for Infrared Remote Control Processing

2. Discrimination between 0 and 1
   The sample task discriminates between 0 and 1 depending on the length of the carrier wave.

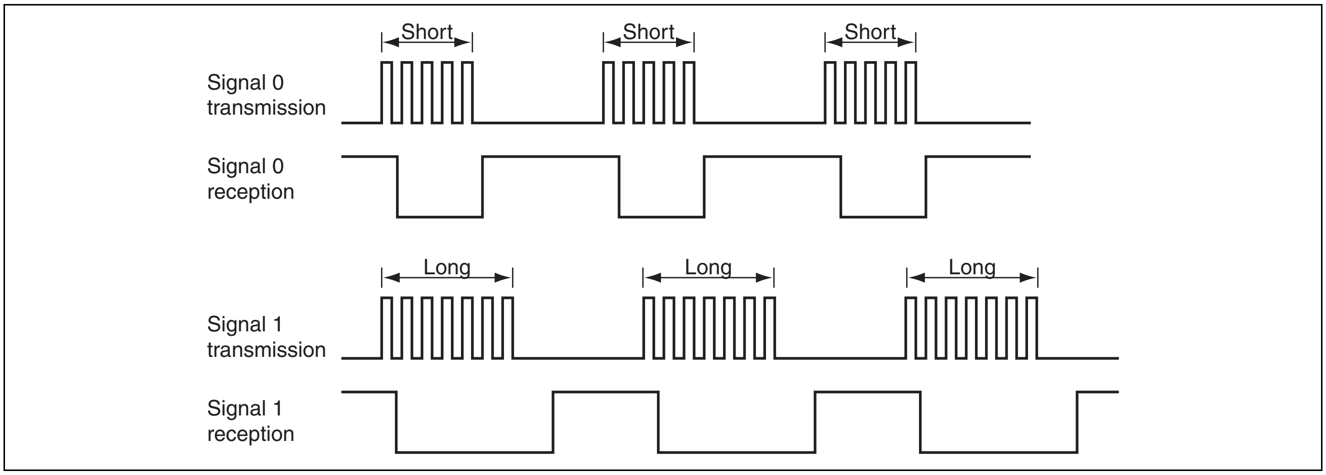


Figure 2.2 Discrimination between 0 and 1

3. Description of Functions

1. Table 3.1 lists specifications of infrared light-emitting diode GL538 used for the sample task. Table 3.2 lists specifications of infrared light reception device NJL71V380A used for the sample task. (If stands for the forward current, Ifm for the peak forward current, and Vr for the reverse voltage.)

Table 3.1 Specifications of Infrared Light-Emitting Diode

Item	Condition	Remarks
Product type	—	GL538
Manufacturer	—	SHARP CORPORATION
Operating power supply voltage Vf	If = 50 mA	1.3 V to 1.5 V
Peak forward voltage Vfm	Ifm = 0.5 A	1.9 V to 3.0 V
Reverse current Ir	Vr = 3 V	10 μA
Emitted peak waveform λp	If = 5 mA	950 nm
Half-value waveform Δλ	If = 5 mA	45 nm
Emitted output Ie	If = 50 mA	15 to 30 mW/sr
Half-value angle Δθ	If = 20 mA	±13 deg
Oscillation	—	300 kHz

Table 3.2 Specifications of Infrared Light Reception Device

Item	Condition	Remarks
Product type	—	NJL71V380A
Manufacturer	—	New Japan Radio Co., Ltd.
Carrier frequency	—	38 kHz
Operating power supply voltage	—	2.4 V to 5.5 V
Current consumption	No incident light	0.6 mA (Maximum value)
Reach	Optical axis center direction, carrier frequency	18 m
Vertical half-value angle	Horizontal directivity at the half of the reach	45 deg
	Vertical directivity at the half of the reach	30 deg
Low-level output voltage	30 cm in the optical axis direction	0.5 V (Maximum value)
High-level output voltage	30 cm in the optical axis direction	2.8 V
Low-level pulse width (TwL)	Defined by the period width of output TwL and TwH in the range from 30 cm in the optical axis direction to the reach. (Average value of 50 pulses)	350 to 800 μs
High-level pulse width (TwH)		400 to 850 μs

2. This paragraph explains the functions supported by the infrared remote controller.

   A. Figure 3.1 shows the block diagram of the H8/38024 functions used for infrared remote control processing.

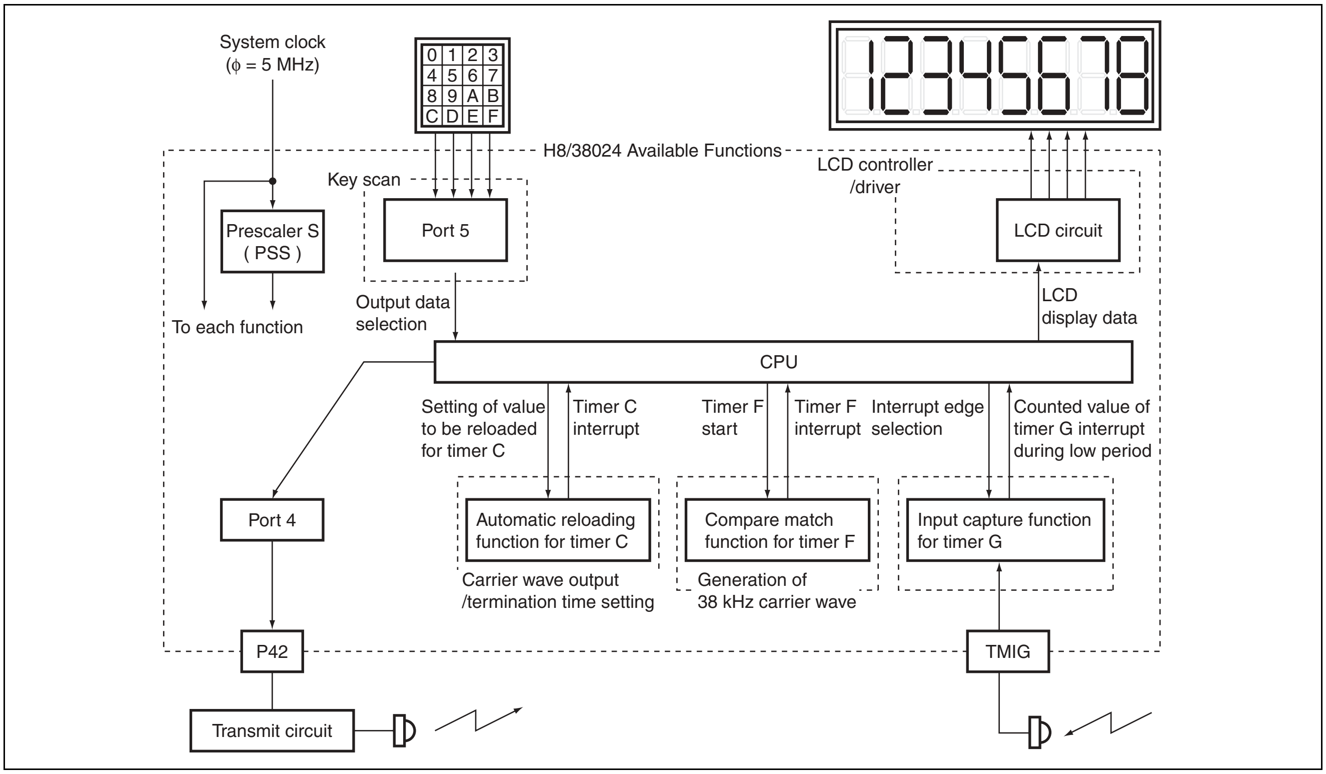


Figure 3.1 Block Diagram of Infrared Remote Control Processing

B. The following explains the H8/38024 functions supported by the infrared remote controller.

- System clock (ϕ)
  This is a 5-MHz clock obtained by dividing the 10-MHz OSC clock by two, being the reference clock used to operate the CPU and its peripheral functions.
- Prescaler S (PSS)
  This is a 13-bit counter that receives ϕ as the input, counting up every cycle.
- Port 4
  Uses P42 as the output pin to output the carrier wave.
- Port 5
  This port is connected to the key switch to select infrared transmit data.
- Timer C automatic reloading function
  Controls the interval between carrier wave output and termination.
- Timer F compare match function
  Generates the 38-kHz carrier wave.
- Timer G input capture function
  Receives an output value from the infrared light reception device for receive data determination.
- LCD controller/driver
  Displays receive data on the LCD.

3. This paragraph explains the functions supported by port 4.

   A. Figure 3.2 shows the block diagram of carrier wave transmission by port 42.

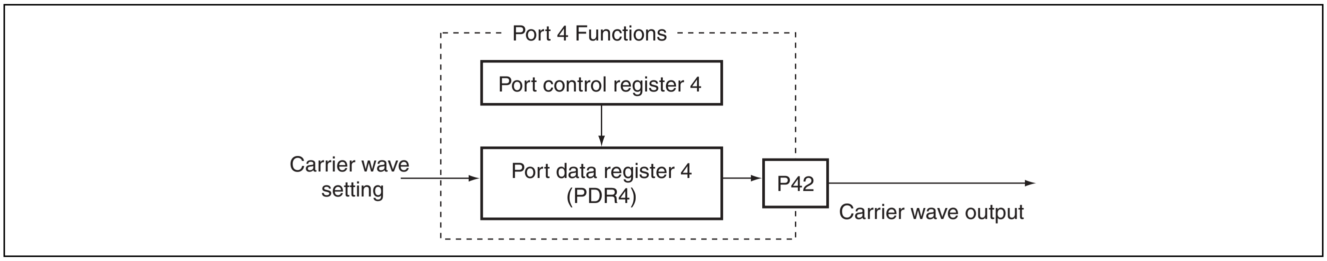


Figure 3.2 Block Diagram of Port 4

   B. The sample task uses port 42 to output a carrier wave. The following explains the block diagram of port 4.

   - Port control register 4 (PCR4)
     Sets input/output for port 4. When PCR4 = 0xFC, PCR4 sets P42 as the output port.
   - Port data register 4 (PDR4)
     Sets data to be stored into output port P42. Using timer F interrupt, PDR4 toggles P42 to output the 38-kHz carrier wave.

4. This paragraph explains the functions supported by port 5.

   A. Figure 3.3 shows the block diagram of the key input circuit supported by port 5.

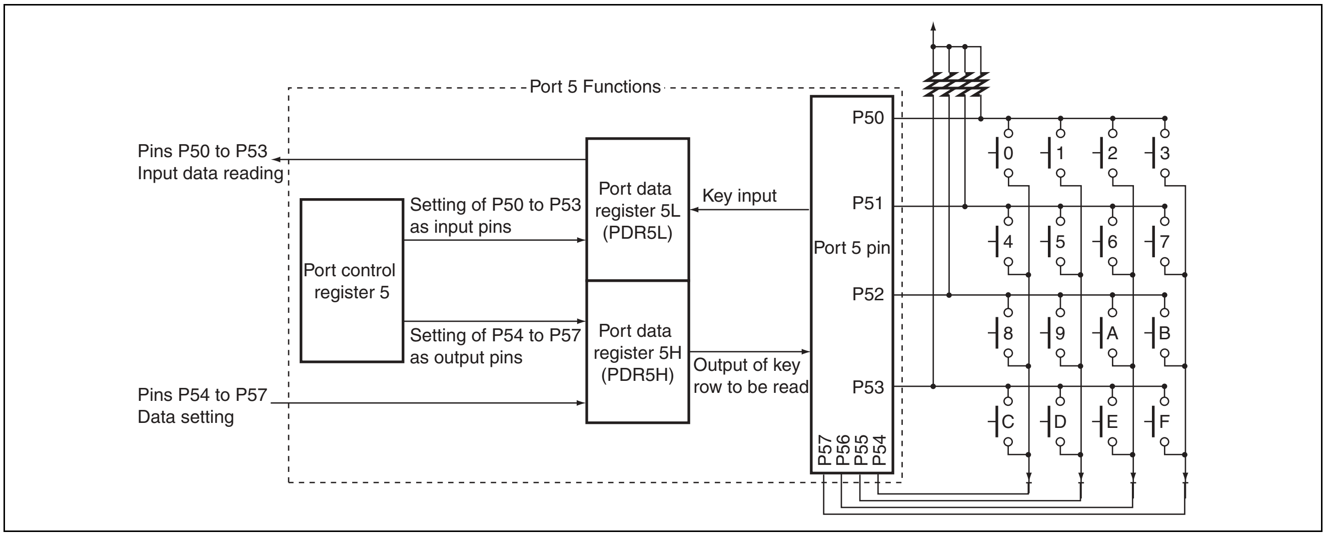


Figure 3.3 Block Diagram of Key Input Circuit Supported by Port 5

   B. The sample task uses port 5 for key input. The following explains the block diagram of the key input circuit.

   - Port control register 5 (PCR5)
     Sets input/output for port 5. When PCR5 = 0xF0, PCR5 sets P54 to P57 as the output ports, and P50 to P53 as the input ports.
   - Port data register 5H (PDR5H)
     Uses the upper four bits of PDR5 as PDR5H to set data to be stored into output ports P54 to P57. Out of P54 to P57, the status of a key row set at the low level is reflected in P50 to P53.
   - Port data register 5L (PDR5L)
     Uses the lower four bits of PDR5 as PDR5L to reflect the values of input ports P50 to P53 in PDR5L. The value stored in PDR5L provides the status of a key row selected by PDR5H.

5. This paragraph explains the timer C automatic reloading function.

   A. Figure 3.4 shows the block diagram of the timer C automatic reloading function.

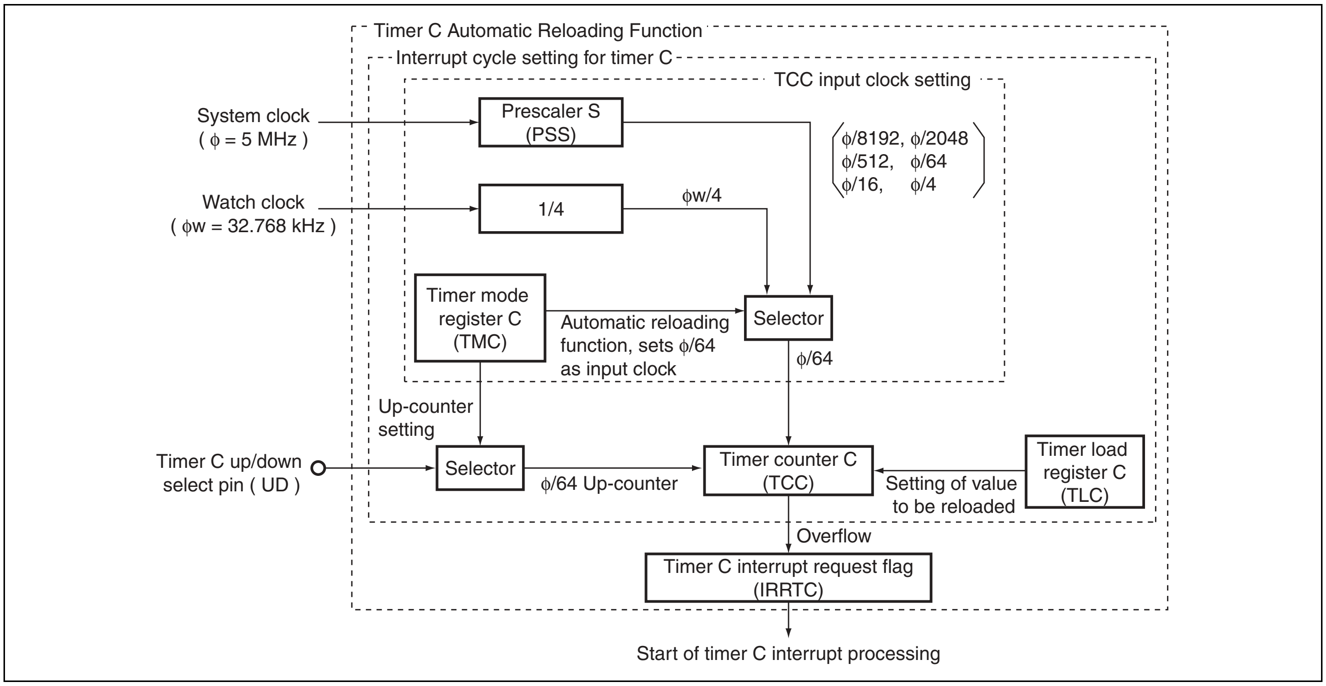


Figure 3.4 Block Diagram of Timer C Automatic Reloading Function

   B. The following explains the block diagram of the timer C automatic reloading function.

- Timer mode register C (TMC)

  This is an eight-bit readable/writable register that selects the automatic reloading function, controls increment/decrement of the timer counter C (TCC), and selects an input clock. For control over the increment/decrement of TCC, it selects hardware control by UD pin input, or the up/down counter by software control.

- Timer counter C (TCC)

  This is an eight-bit readable counter that is counted up or down by an internal input clock or an external event. As the input clocks, the system clock divided by 8192, 2048, 512, 64, 16, or 4, the watch clock divided by 4, and an external clock can be selected. The sample task sets TCC as the up-counter, and selects the system clock divided by 64 as the TCC input clock.

- Timer load register C (TLC)

  This is an eight-bit write-only register that sets a value to be reloaded to TCC.

- Timer C interrupt request flag (IRRTC)

  This flag is set to 1 by a TCC overflow. A timer C interrupt is accepted with the IRRTC at 1, the timer C interrupt enable (IENTC) for the interrupt enable register 2 (IENR2) set at 1, and bit I in the condition code register (CCR) cleared to 0. Then, timer C interrupt processing starts.

6. This paragraph explains the timer F output compare function.

   A. Figure 3.5 shows the block diagram of the timer F output compare function.

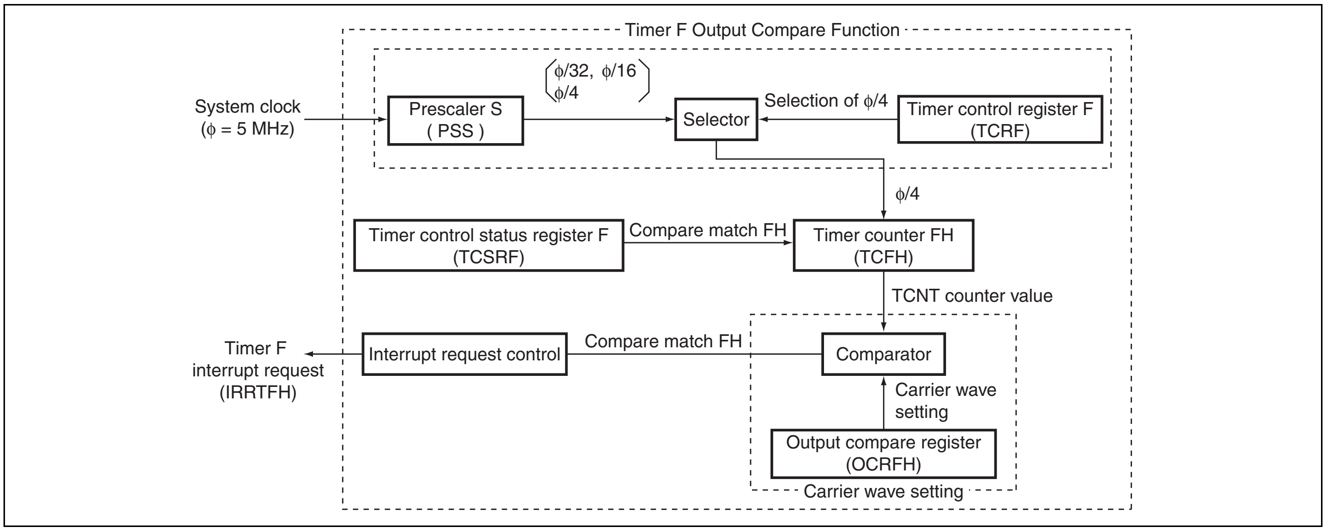


Figure 3.5 Block Diagram of Timer F Output Compare Function

   B. The following explains the block diagram of the timer F output compare function.

   - Timer counter FH (TCFH)
     This is an eight-bit readable/writable up-counter that is incremented by an input internal/external clock. Four internal clocks obtained by dividing ϕ, and one external clock can be selected as the input clock.
   - Timer control register F (TCRF)
     This is an eight-bit writable register that sets the TCFH input clock. The sample task uses ϕ divided by four as the TCFH input clock.
   - Timer control status register F (TCSRF)
     This is an eight-bit register that selects counter clearing, sets the compare match flag and timer overflow flag, and controls enabling an interrupt request resulting from an overflow. The sample task enables TCFH clearing by a compare match FH, and disables an interrupt resulting from a timer FH overflow.
   - Output compare register FH (OCRFH)
     This is an eight-bit readable/writable register. The contents of OCRFH are always compared with that of TCFH. If they match, the compare match flag H (CMFH) in TCSRF is set to 1. Then, a compare match FH is generated, requesting a CPU interrupt.

7. This paragraph explains the timer G input capture function.

A. Figure 3.6 shows the block diagram of the timer G input capture function.

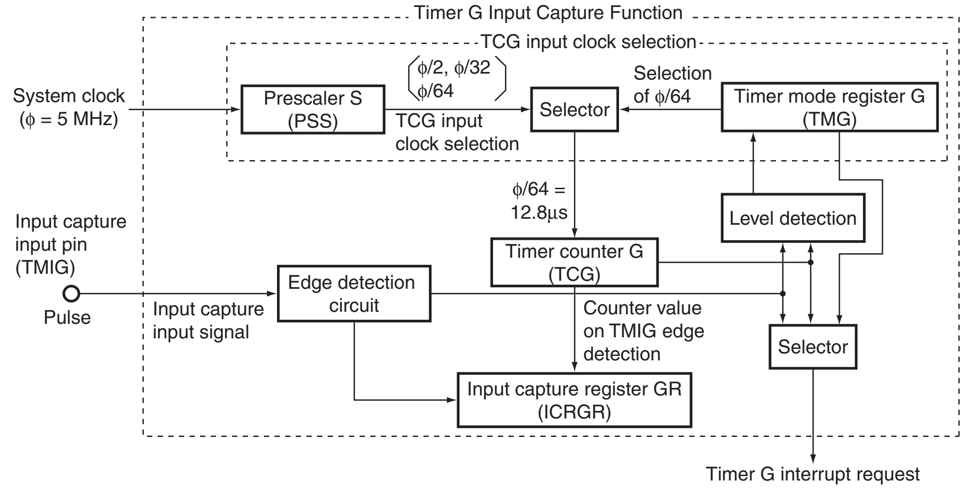


Figure 3.6 Block Diagram of Timer G Input Capture Function

B. The following explains the timer G input capture function.

- Timer counter G (TCG)
  This is an eight-bit unreadable/unwritable up-counter that is incremented by an input internal/external clock. As the input clocks, the system clock divided by 2, 32, and 64, and an external clock can be selected. The sample task selects the system clock divided by 64 as the TCG input clock.
- Timer mode register G (TMG)
  This is an eight-bit readable/writable register that selects the TCG input clock, counter clearing, and the edge of an interrupt request by an input capture input signal, controls enabling/disabling an interrupt request resulting from an overflow, and displays the overflow flag.
- Input capture register GR (ICRGR)
  This is an eight-bit read-only register that transfers the TCG value to ICRGR upon detection of the rising edge of the input capture input signal. When the IRRTG bit in IRR2 is set to 1, a CPU interrupt is requested.
- Input capture input pin (TMIG)
  Receives an output value from the infrared light reception device, measuring the low and high periods.

8. This paragraph explains the LCD controller/driver.

A. Figure 3.7 shows the block diagram of the LCD controller/driver.

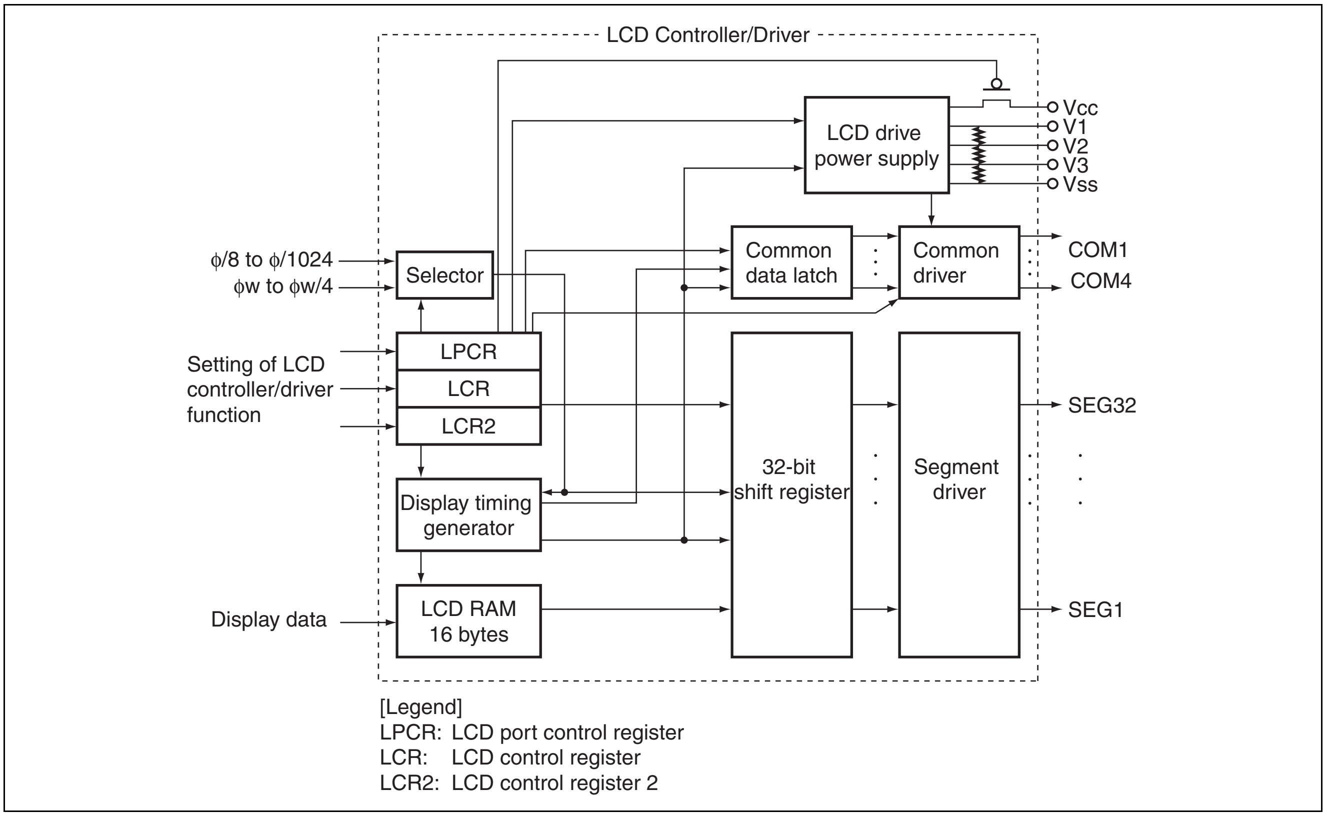


Figure 3.7 Block Diagram of LCD Controller/Driver

B. The following explains the functions supported by the LCD controller/driver.

- LCD port control register (LPCR)

  This is an eight-bit readable/writable register that selects a duty cycle, LCD driver, and pin functions.
- LCD control register (LCR)

  This is an eight-bit readable/writable register that turns on or off the LCD drive power supply, controls the start of the display function and display data, and selects a frame frequency.
- LCD control register 2 (LCR2)

  This is an eight-bit readable/writable register that controls waveform A or B selection, and selects a clock for the triple step-up circuit, a drive power supply, and a duty cycle in a period when the split-resistor for the power supply is connected to the power supply circuit.
- Segment output pins (SEG1 to SEG32)

  These pins are used to drive the LCD segment. All pins are shared with ports and can be set in a programmable way.
- Common output pins (COM1 to COM4)

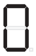
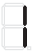
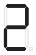
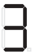
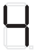
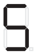
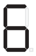
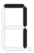
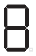
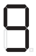
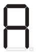
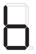
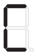
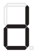
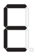
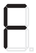
  These are the common drive pins for the LCD. In static or 1/2 duty cycle mode, pins can be used in parallel.
- LCD power supply pins (V1, V2, V3)

  These pins are used to connect an external bypass capacitor or to use an external power supply circuit.
- LCD RAM

  Sets display data. The relationship between the LCD RAM and the display segment depends on the duty cycle. Automatic displays are started by setting a register group needed for them, followed by writing data into the section for a duty cycle by the same instruction as for the normal RAM and turning on them. The word/byte access instructions can be used to set the RAM.

C. Table 3.3 lists SEG21 and SEG22 displays in the three-digit eight-segment LCD and sample display data.

Table 3.3 Sample Display Data

			Display data								
Symbol	Display	Address	Binary data								Hexadecimal data
0		0xF746	1	1	0	1	0	1	1	1	0xD7
1		0xF746	0	0	0	0	0	1	1	0	0x06
2		0xF746	1	1	1	0	0	0	1	1	0xE3
3		0xF746	1	0	1	0	0	1	1	1	0xA7
4		0xF746	0	0	1	1	0	1	1	0	0x36
5		0xF746	1	0	1	1	0	1	0	1	0xB5
6		0xF746	1	1	1	1	0	1	0	1	0xF5
7		0xF746	0	0	0	0	0	1	1	1	0x07
8		0xF746	1	1	1	1	0	1	1	1	0xF7
9		0xF746	1	0	1	1	0	1	1	1	0xB7
A		0xF746	0	1	1	1	0	1	1	1	0x77
B		0xF746	1	1	1	1	0	1	0	0	0xF4
C		0xF746	1	1	0	1	0	0	0	1	0xD1
D		0xF746	1	1	1	0	0	1	1	0	0xE6
E		0xF746	1	1	1	1	0	0	0	1	0xF1
F		0xF746	0	1	1	1	0	0	0	1	0x71

9. Table 3.4 lists the functions assigned for the sample task to carry out infrared remote control processing.

Table 3.4 Functions Assigned on Transmit Side

Function	Function Allocation
PSS	13-bit counter that receives the system clock as the input.
TMC	Sets the timer C automatic reloading register function and input clock.
TCC	Counter for timer C
TLC	Sets the duration for outputting/terminating the carrier wave.
TCRF	Sets the input clock for timer FH.
TCSRF	Sets TCFH clearing by compare match.
TCFH	Counter for timer FH
OCRFH	Generates timing for the carrier wave.
PCR4	Sets P42 as the output port.
PDR4 (P42)	Carrier wave output pin
PMR5	Selects a pin function for port 5.
PDR5	Key switch input pin
PUCR5	Turns off the pull-up MOS for port 5.
PCR5	Sets input/output for port 5.
TMG	Clears TCG, and sets the input clock and an interrupt edge.
TCG	Counter for timer G
ICRGR	Stores data received by the infrared light reception device.
LPCR	Selects a duty cycle for the LCD, and a segment pin.
LCR	Turns on and off the LCD, and sets a frame frequency.
LCR2	Selects waveform A or B for the LCD.
LCDRAM	Stores LCD display data.
TMIG	Receives the output value of the infrared light reception device as the input.
NCS	Turns on and off the noise elimination function for an input capture input signal.
IENTG	Enables an interrupt request by an input capture input signal.
IENTFH	Enables an interrupt request by timer FH compare match.
IENTC	Enables an interrupt request resulting from timer C overflow.
IRRTG	Interrupt flag by an input capture input signal
IRRTFH	Interrupt flag by timer FH compare match
IRRTC	Interrupt flag resulting from timer C overflow

4. Principle of Operation

1. Figure 4.1 shows the flowchart for the infrared remote controller. Following that flowchart, infrared remote control processing is carried out.

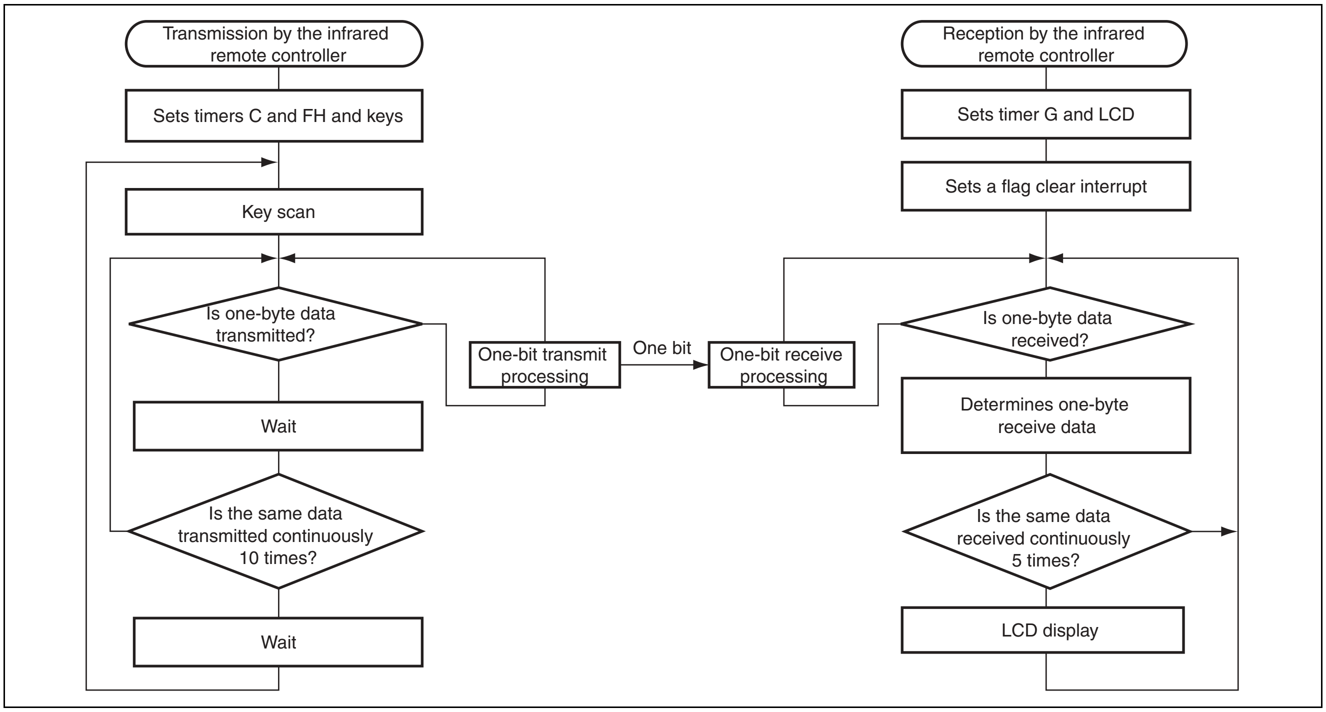

Transmission by the infrared remote controller
Sets timers C and FH and keys
Key scan
Is one-byte data transmitted?
Wait
Is the same data transmitted continuously 10 times?
Wait
One-bit transmit processing
One bit
One-bit receive processing
Reception by the infrared remote controller
Sets timer G and LCD
Sets a flag clear interrupt
Is one-byte data received?
Determines one-byte receive data
Is the same data received continuously 5 times?
LCD display

Figure 4.1 Flowchart for Infrared Remote Controller

2. Discrimination between 0 and 1
   The length of a carrier wave is used to discriminate between 0 and 1.

Transmit Bit	Carrier Wave Output Period
0	380 µs
1	770 µs

The carrier wave termination period during one-bit data transmission is 600 µs, being constant.

- Timer C setting
  Timer C input clock: ϕ/64 = 78.1 kHz
  Timer C count time: 1 / 78.1 kHz = 12.8 µs
- Transmit bit 0 (380 µs) setting
  Timer C count needed for 380 µs
  380 µs / 12.8 µs = 29.6 = 30
  TLC setting
  (For the up-counter) 256 – 30 = 226
- Transmit bit 1 (770 µs) setting
  Timer C count needed for 770 µs
  770 µs / 12.8 µs = 60.1 = 61
  TLC setting
  (For the up-counter) 256 – 61 = 195
- Setting the carrier wave termination period during one-bit data transmission (600 µs)
  Timer C count needed for 600 µs
  600 µs / 12.8 µs = 46.8 = 47
  TLC setting
  (For the up-counter) 256 – 47 = 209

3. One-byte data determination
   The transmit side transmits the same data 10 times. If it has been received continuously five times, the data is stored as the correct value.
4. One-byte transmit interval
   The carrier wave termination period during one-byte data transmission is 2 ms.
   If the carrier wave termination period exceeds 2 ms, the receive side identifies the start bit of the data.
   — Setting the carrier wave termination period during one-byte data transmission (2 ms)
   Timer C count needed for 2 ms
   2000 µs / 12.8 µs = 156.2 = 157
   TLC setting
   (For the up-counter) 256 – 157 = 99

5. Discrimination between retransmit and new data on receive side
   If the carrier wave termination period exceeds 3.2 ms, the receive side identifies new data.

5. Description of Software

5.1 Modules

Table 5.1 lists the modules used for the sample task.

Table 5.1 Description of Modules

Module	Label	Function
Main routine	main	Sets an interrupt, initializes the LCD, receives one byte of infrared data for data determination, and displays data on the LCD.
Initialization of remote controller transmission	sendir_init	Initializes transmit processing by the remote controller.
One-byte infrared transmission	sendir	Converts infrared data by bit.
Key scan	keyscan	Identifies a selection key, and transmits one-byte data 10 times by infrared.
Port 5L reading	keyread	Returns the contents of P50 to P53.
Timer C interrupt	tcint	Generates a timer C interrupt to set the flag.
Timer F interrupt	tfint	Generates a timer FH interrupt to toggle P42 for carrier wave generation.
Timer G interrupt	tgint	Starts and measures input capture, and sets the flag.
LCD initialization	lcd_init	Initializes the LCD RAM.

5.2 Arguments

Table 5.2 lists the arguments used for the sample task.

Table 5.2 Description of Arguments

Argument	Function	Used in	Data Length	Input/ Output
sdt	One-byte transmit data	sendir	One byte	Input

5.3 Internal Registers

Table 5.3 lists the internal registers used for the sample task.

Table 5.3 Description of Internal Registers

Register Name		Function	Address	Setting
TMC	TMC7	Timer mode register C (automatic reloading function selection) When TMC7 = 0, sets the timer C function as the interval function. When TMC7 = 1, sets the timer C function as the automatic reloading function.	0xFFB4 Bit 7	1

	TMC6
	TMC5
		Timer mode register C
		(counter increment/decrement control)
		When TMC6 = 0 and TMC5 = 0, TCC increments.
		When TMC6 = 0 and TMC5 = 1, TCC decrements.
		When TMC6 = 1 and TMC5 = x, TCC is controlled by hardware according to UD pin input.
		Note: x = Don't care
			0xFFB4
			Bit 6
			Bit 5
				TMC6 = 0
				TMC5 = 0

	TMC2
	TMC1
	TMC0
		Timer mode register C (clock selection)
		When TMC2 = 0, TMC1 = 1, and TMC0 = 1, TCC counts at φ/64.
			0xFFB4
			Bit 2
			Bit 1
			Bit 0
				TMC2 = 0
				TMC1 = 1
				TMC0 = 1

TCC		Timer counter C This is an eight-bit up-counter that receives the system clock divided by 16 as the input. TCC is loaded with TLC settings upon an overflow.	0xFFB5	0x00
TLC		Timer load register C With a value set in TLC, TCC starts to count up from TLC settings. It is loaded with TLC settings upon an overflow.	0xFFB5	—

TCRF	CKSH2
	CKSH1
	CKSH0
		Timer control register F (clock selection H)
		When CKSH2 = 1, CKSH1 = 1, and CKSH0 = 0, TCFH counts at φ/4.
			0xFFB6
			Bit 6
			Bit 5
			Bit 4
				CKSH2 = 1
				CKSH1 = 1
				CKSH0 = 0

TCSRF	TFOVFH	Timer control status register F (timer overflow flag H) When TFOVFH = 0, TCF has not overflowed. When TFOVFH = 1, TCF has overflowed.	0xFFB7 Bit 7	0
	CMFH	Timer control status register F (compare match flag H) When CMFH = 0, a compare match F has not occurred. When CMFH = 1, a compare match F has occurred.	0xFFB7 Bit 6	0
	OVIEH	Timer control status register F (timer overflow interrupt enable H) When OVIEH = 0, disables an interrupt request resulting from TCF overflow. When OVIEH = 1, enables an interrupt request resulting from TCF overflow.	0xFFB7 Bit 5	0
	CCLRH	Timer control status register F (counter clear H) When CCLRH = 0, disables TCFH clearing by compare match. When CCLRH = 1, enables TCFH clearing by compare match.	0xFFB7 Bit 4	1

Register		Function	Address	Setting
TCFH		Eight-bit timer counter FH Eight-bit up-counter that receives ϕ/4 as the input	0xFFB8	0x00
OCRFH		Output compare register FH Compared with TCFH. If the OCRFH value matches the TCFH one, the CMFH in TCSRF is set to 1.	0xFFBA	0x80
PMR5		Port mode register 5 (P5n/$\overline{\text{WKPn}}$/SEGn+1 pin function selection) Pin function with pin SEGn+1 not used When PMR5 = 0x00, pin P5n/$\overline{\text{WKPn}}$/SEGn+1 functions as the P5n input/output pin.	0xFFCC	0x00
PDR4	P42	Port data register 4 (P42) When P42 = 0, sets pin P42 at the low level. When P42 = 1, sets pin P42 at the high level.	0xFFD7 Bit 2	—
PDR5	PDR5H	Port data register 5 (P54 to P57) The upper four bits of PDR5 provide PDR5H to set data to be stored into output ports P54 to P57. Out of the data in P54 to P57, the status of a key row set at the low level is reflected in P50 to P53. When PDR5 = 0xE0, selects the key row (0, 4) in P54. When PDR5 = 0xD0, selects the key row (1, 5) in P55. When PDR5 = 0xB0, selects the key row (2, 6) in P56. When PDR5 = 0x70, selects the key row (3, 7) in P57.	0xFFD8 Bits 4 to 7	—
	PDR5L	Port 7 register (P50 to P53) The lower four bits of PDR5 provide PDR5L to reflect the values of input ports P50 to P53.	0xFFD8 Bits 0 to 3	—
PUCR5		Port pull-up control register 5 When PUCR5 = 0x00, turns off the pull-up MOS.	0xFFE2	0x00
PCR4		Port control register 4 When PCR4 = 0xFC, sets P42 as the output port.	0xFFE8	0xFC
PCR5		Port control register 5 When PCR5 = 0xF0, sets P54 to P57 as the output ports, and P50 to P53 as the input ports.	0xFFE8	0xF0
TMG	TGOVFH	Timer mode register G (timer overflow flag H) When TGOVFH = 0, the input capture input signal is at the high level, and TCG has not overflowed. When TGOVFH = 1, the input capture input signal is at the high level, and TCG has overflowed.	0xFFBC Bit 7	0
	TGOVFL	Timer mode register G (timer overflow flag L) When TGOVFL = 0, the input capture input signal is at the low level, and TCG has not overflowed. When TGOVFL = 1, the input capture input signal is at the low level, and TCG has overflowed.	0xFFBC Bit 6	0
	OVIE	Timer mode register G (timer overflow interrupt enable) When OVIE = 0, disables an interrupt request resulting from TCG overflow. When OVIE = 1, enables an interrupt request resulting from TCG overflow.	0xFFBC Bit 5	0

Register		Function	Address	Setting
TMG	IIEGS	Timer mode register G (input capture interrupt edge selection) When IIEGS = 0, generates an interrupt at the rising edge of an input capture input signal. When IIEGS = 1, generates an interrupt at the falling edge of an input capture input signal.	0xFFBC Bit 4	0

	CCLR1
	CCLR0
		Timer mode register G (counter clear 1 and 0)
		When CCLR1 = 1 and CCLR0 = 0, clears TCG at the rising edge of an input capture input signal.
			0xFFBC
			Bit 3
			Bit 2
				CCLR1 = 1
				CCLR0 = 0

	CKS1
	CKS0
		Timer mode register G (clock selection 1 and 0)
		When CKS1 = 0 and CKS0 = 0, sets the TCG input count at ϕ/64.
			0xFFBC
			Bit 1
			Bit 0
				CKS1 = 0
				CKS0 = 0

TCG		Timer counter G This is an eight-bit unreadable/unwritable register that increments by an input clock. Upon detection of the rising edge of an input capture input signal, the TCG value is transferred to the input capture register GR (ICRGR).	—	—
ICRGR		Input capture register GR This is an eight-bit read-only register. Upon detection of the rising edge of an input capture input signal, the TCG value is transferred.	0xFFBE	—

LPCR	DTS1
	DTS0
		LCD port control register (duty cycle selection 1 and 0)
		A combination of DTS1 and DTS0 selects the static cycle or a duty cycle of 1/4 to 1/2.
		When DTS1 = 1 and DTS0 = 1, selects a duty cycle of 1/4.
			0xFFC0
			Bit 7
			Bit 6
				DTS1 = 1
				DTS0 = 1

	CMX	LCD port control register (common function selection) Selects whether to output the same waveform from multiple common pins not used at a certain duty cycle in order to enhance common drive capability. When CMX = 0, does not output the same waveform from multiple common pins not used at a duty cycle. When CMX = 1, outputs the same waveform from multiple common pins not used at a duty cycle.	0xFFC0 Bit 5	0

	SGS3
	SGS2
	SGS1
	SGS0
		LCD port control register
		(segment driver selection 0 to 3)
		Selects a segment driver to be used.
		When SGS3 = 1, SGS2 = 0, SGS1 = 1, and SGS0 = 1, pins SEG13 to SEG32 function as the segment drivers, while pins SEG1 to SEG12 function as the ports.
			0xFFC0
			Bit 3
			Bit 2
			Bit 1
			Bit 0
				SGS3 = 1
				SGS2 = 0
				SGS1 = 1
				SGS0 = 1

Register		Function	Address	Setting
LCR	PSW	LCD control register (LCD power supply split-resistor connection control) If the LCD displays no data in power-down mode or if an external power supply is used, the split-resistor for the LCD power supply can be disconnected from Vcc. When ACT = 0 or in standby mode, the split-resistor for the LCD power supply is disconnected from Vcc regardless of this bit. When PSW = 0, disconnects the split-resistor for the LCD power supply from Vcc. When PSW = 1, connects the split-resistor for the LCD power supply to Vcc.	0xFFC1 Bit 6	0
	ACT	LCD control resistor (display function start) Selects whether to use the LCD controller/driver. Clearing this bit to 0 terminates the operation of the LCD controller/driver. Furthermore, the LCD drive power supply is turned off regardless of the PSW value. The contents of the register are, however, retained. When ACT = 0, the LCD controller/driver terminates. When ACT = 1, the LCD controller/driver operates.	0xFFC1 Bit 5	1
	DISP	LCD control register (display data control) Selects whether to display the contents of the LCD RAM or display blank data regardless of those contents. When DISP = 0, displays blank data. When DISP = 1, displays LCDRAM data.	0xFFC1 Bit 4	1

LCR	CKS3
	CKS2
	CKS1
	CKS0
		LCD control register (frame frequency selection 0 to 3)
		Selects a clock to be used and frame frequency.
		When CKS3 = 1, CKS2 = 1, CKS1 = 1, and CKS0 = 0, selects ϕ/128 as the clock to be used.
			0xFFC1
			Bit 3
			Bit 2
			Bit 1
			Bit 0
				CKS3 = 1
				CKS2 = 1
				CKS1 = 1
				CKS0 = 0

LCR2	LCDAB	LCD control register 2 (waveform A or B selection control) Selects whether to use waveform A or B to drive the LCD. When LCDAB = 0, uses waveform A to drive the LCD. When LCDAB = 1, uses waveform B to drive the LCD.	0xFFC2 Bit 7	0
LCDRAM		LCDRAM Sets display data for the LCD.	0xF740 to 0xF74F	—
PMR1	TMIG	Port mode register 1 (P13/TMIG pin function selection) When TMIG = 0, pin P13/TMIG functions as the P13 input/output pin. When TMIG = 1, pin P13/TMIG functions as the TMIG input pin.	0xFFC8 Bit 3	1
PMR2	NCS	Port mode register 2 (TMIG noise canceler selection) When NCS = 0, cancels the noise elimination function for an input capture input signal. When NCS = 1, activates the noise elimination function for an input capture input signal.	0xFFC9 Bit 1	0

Register		Function	Address	Setting
IENR2	IENTG	Interrupt enable register 2 (timer G interrupt enable) Controls enabling/disabling of a timer G interrupt request. When IENTG = 0, disables a timer G interrupt request. When IENTG = 1, enables a timer G interrupt request.	0xFFF4 Bit 4	1
	IENTFH	Interrupt enable register 2 (timer FH interrupt enable) When IENTFH = 0, disables a timer FH interrupt request. When IENTFH = 1, enables a timer FH interrupt request.	0xFFF4 Bit 3	1
	IENTC	Interrupt enable register 2 (timer C interrupt enable) When IENTC = 0, disables a timer C interrupt request. When IENTC = 1, enables a timer C interrupt request.	0xFFF4 Bit 1	1
IRR2	IRRTG	Interrupt request register 2 (timer G interrupt request flag) Reflects whether a timer G interrupt is requested. When IRRTG = 0, a timer G interrupt is not requested. When IRRTG = 1, a timer G interrupt is requested.	0xFFF7 Bit 4	0
	IRRTFH	Interrupt request register 2 (timer FH interrupt request flag) When IRRTFH = 0, a timer FH interrupt is not requested. When IRRTFH = 1, a timer FH interrupt is requested.	0xFFF7 Bit 3	0
IRR2	IRRTC	Interrupt request register 2 (timer C interrupt request flag) When IRRTC = 0, a timer C interrupt is not requested. When IRRTC = 1, a timer C interrupt is requested.	0xFFF7 Bit 1	0

5.4 Description of RAM

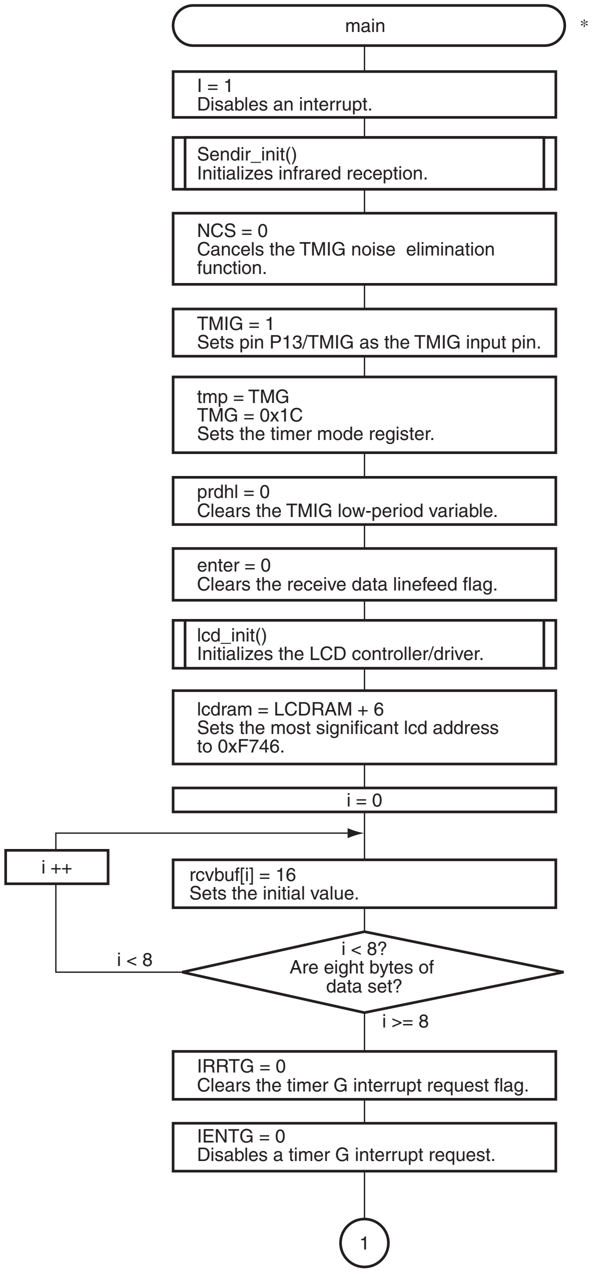
Table 5.4 lists the RAM used for the sample task.

Table 5.4 Description of RAM

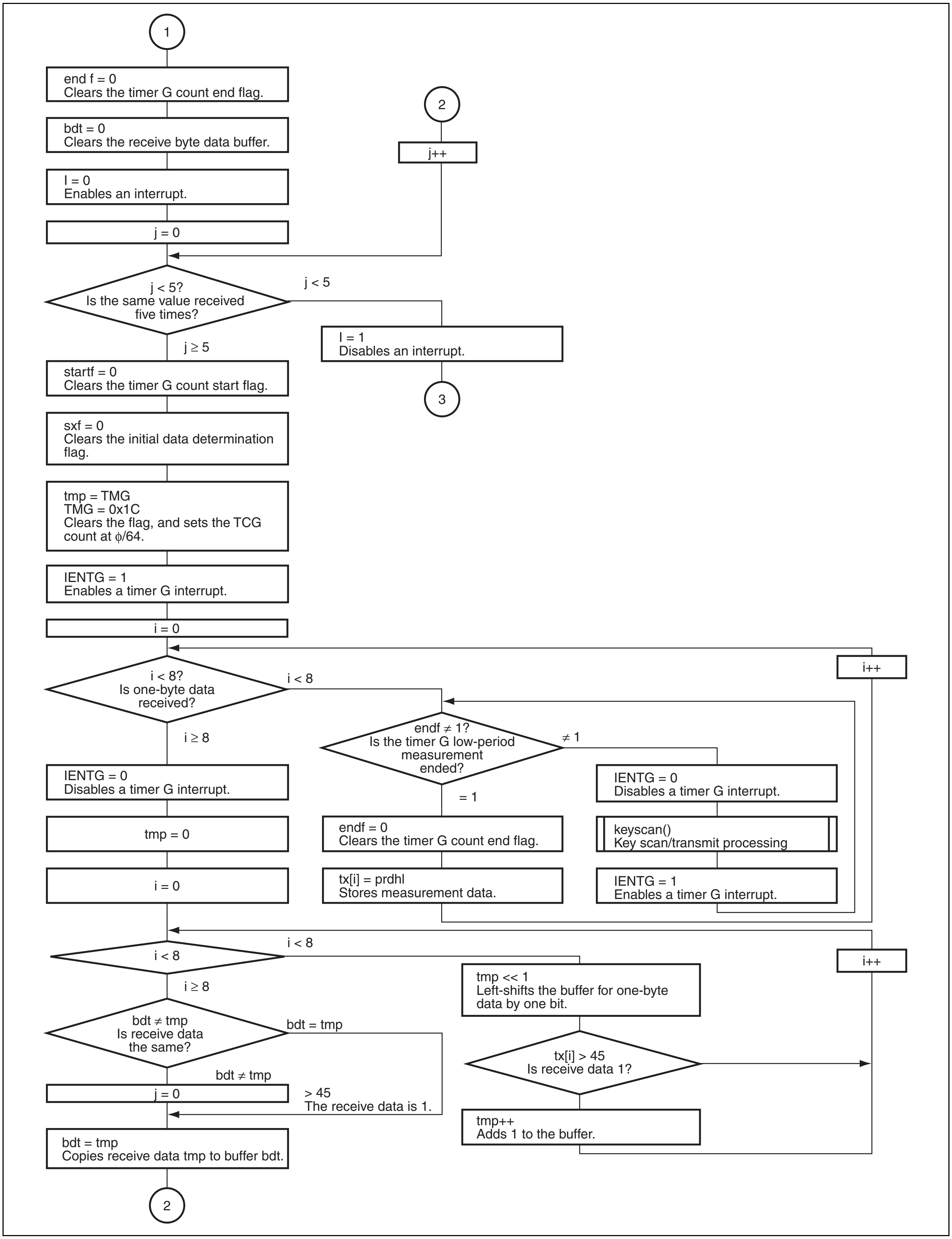
Label	Function	Address	Used in
tcflg	Determines a timer C interrupt.	1 byte	main, tcint, sendir
prdhl	Stores timer G measurement result.	1 byte	main, tgint
sxf	Determines whether the data is new or the same.	1 byte	main, tgint
enter	Determines whether to update the LCD.	1 byte	main, tgint
startf	Flag to determine whether the second timer G interrupt has been received	1 byte	main, tgint
endf	Flag to determine whether cycle measurement has been ended	1 byte	main, tgint

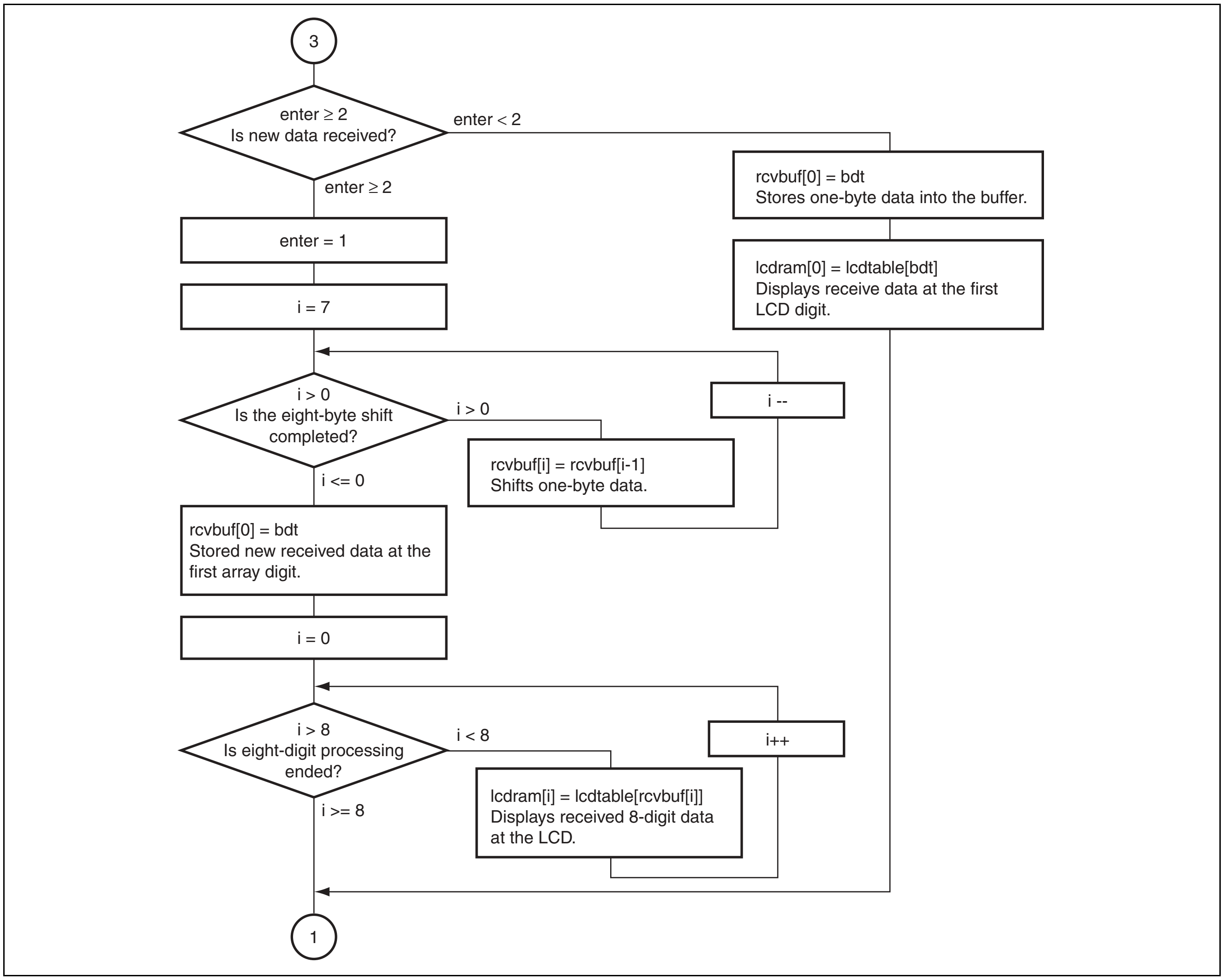
6. Flowchart

1. Main routine

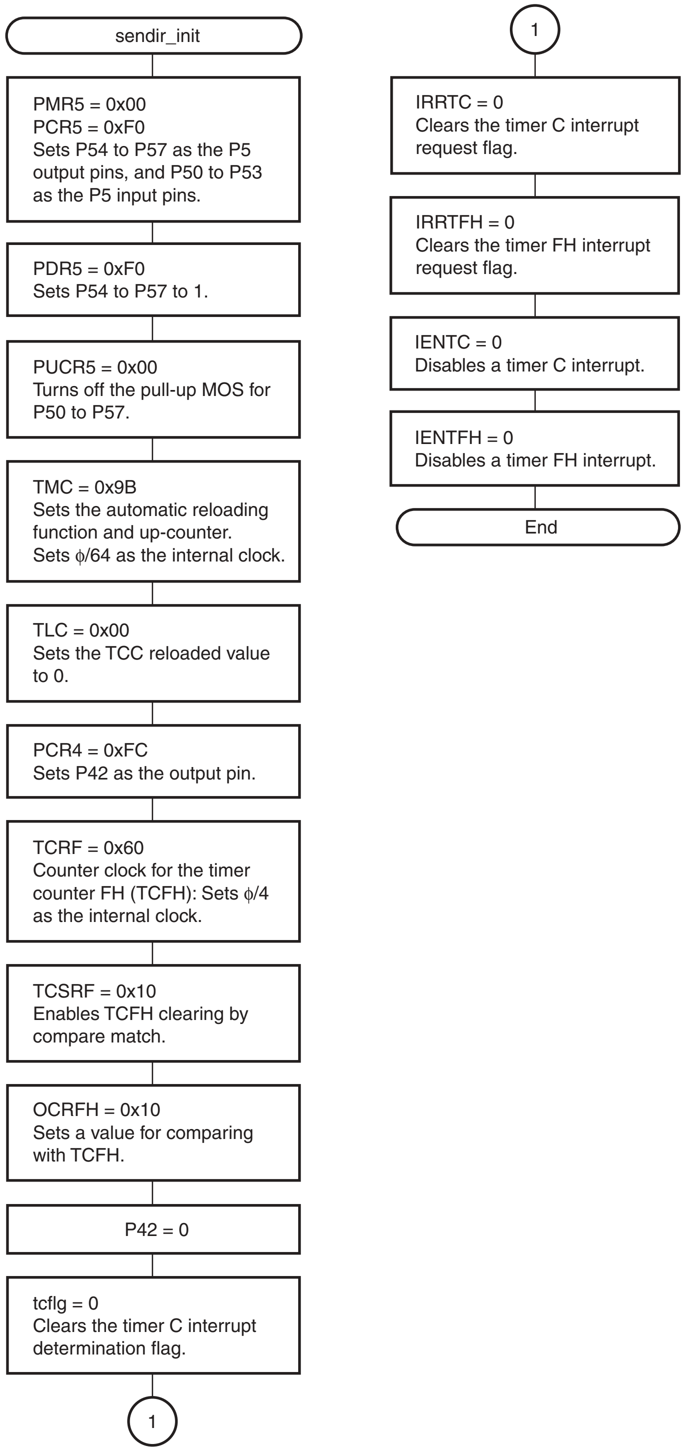

main
*
I = 1
Disables an interrupt.
Sendir_init()
Initializes infrared reception.
NCS = 0
Cancels the TMIG noise elimination function.
TMIG = 1
Sets pin P13/TMIG as the TMIG input pin.
tmp = TMG
TMG = 0x1C
Sets the timer mode register.
prdhl = 0
Clears the TMIG low-period variable.
enter = 0
Clears the receive data linefeed flag.
lcd_init()
Initializes the LCD controller/driver.
lcdram = LCDRAM + 6
Sets the most significant lcd address to 0xF746.
i = 0
i ++
rcvbuf[i] = 16
Sets the initial value.
i < 8
i < 8?
Are eight bytes of data set?
i >= 8
IRRTG = 0
Clears the timer G interrupt request flag.
IENTG = 0
Disables a timer G interrupt request.
1

Note: * In the sample task, the stack pointer is set in INIT.SRC (assembly language).


1
end f = 0
Clears the timer G count end flag.
bdt = 0
Clears the receive byte data buffer.
I = 0
Enables an interrupt.
j = 0
2
j++
j < 5?
Is the same value received
five times?
j < 5
I = 1
Disables an interrupt.
3
j ≥ 5
startf = 0
Clears the timer G count start flag.
sxf = 0
Clears the initial data determination
flag.
tmp = TMG
TMG = 0x1C
Clears the flag, and sets the TCG
count at φ/64.
IENTG = 1
Enables a timer G interrupt.
i = 0
i++
i < 8?
Is one-byte data
received?
i < 8
endf ≠ 1?
Is the timer G low-period
measurement
ended?
≠ 1
IENTG = 0
Disables a timer G interrupt.
keyscan()
Key scan/transmit processing
IENTG = 1
Enables a timer G interrupt.
= 1
endf = 0
Clears the timer G count end flag.
tx[i] = prdhl
Stores measurement data.
i ≥ 8
IENTG = 0
Disables a timer G interrupt.
tmp = 0
i = 0
i < 8
i < 8
i++
tmp << 1
Left-shifts the buffer for one-byte
data by one bit.
tx[i] > 45
Is receive data 1?
tmp++
Adds 1 to the buffer.
i ≥ 8
bdt ≠ tmp
Is receive data
the same?
bdt = tmp
bdt ≠ tmp
j = 0
> 45
The receive data is 1.
bdt = tmp
Copies receive data tmp to buffer bdt.
2


3
enter ≥ 2
Is new data received?
enter < 2
rcvbuf[0] = bdt
Stores one-byte data into the buffer.
enter ≥ 2
enter = 1
lcdram[0] = lcdtable[bdt]
Displays receive data at the first LCD digit.
i = 7
i > 0
Is the eight-byte shift completed?
i > 0
i --
rcvbuf[i] = rcvbuf[i-1]
Shifts one-byte data.
i <= 0
rcvbuf[0] = bdt
Stored new received data at the first array digit.
i = 0
i > 8
Is eight-digit processing ended?
i < 8
i++
lcdram[i] = lcdtable[rcvbuf[i]]
Displays received 8-digit data at the LCD.
i >= 8
1

2. Remote control transmit initialization



PMR5 = 0x00
PCR5 = 0xF0
Sets P54 to P57 as the P5 output pins, and P50 to P53 as the P5 input pins.

PDR5 = 0xF0
Sets P54 to P57 to 1.

PUCR5 = 0x00
Turns off the pull-up MOS for P50 to P57.

TMC = 0x9B
Sets the automatic reloading function and up-counter.
Sets φ/64 as the internal clock.

TLC = 0x00
Sets the TCC reloaded value to 0.

PCR4 = 0xFC
Sets P42 as the output pin.

TCRF = 0x60
Counter clock for the timer counter FH (TCFH): Sets φ/4 as the internal clock.

TCSRF = 0x10
Enables TCFH clearing by compare match.

OCRFH = 0x10
Sets a value for comparing with TCFH.

P42 = 0

tcflg = 0
Clears the timer C interrupt determination flag.

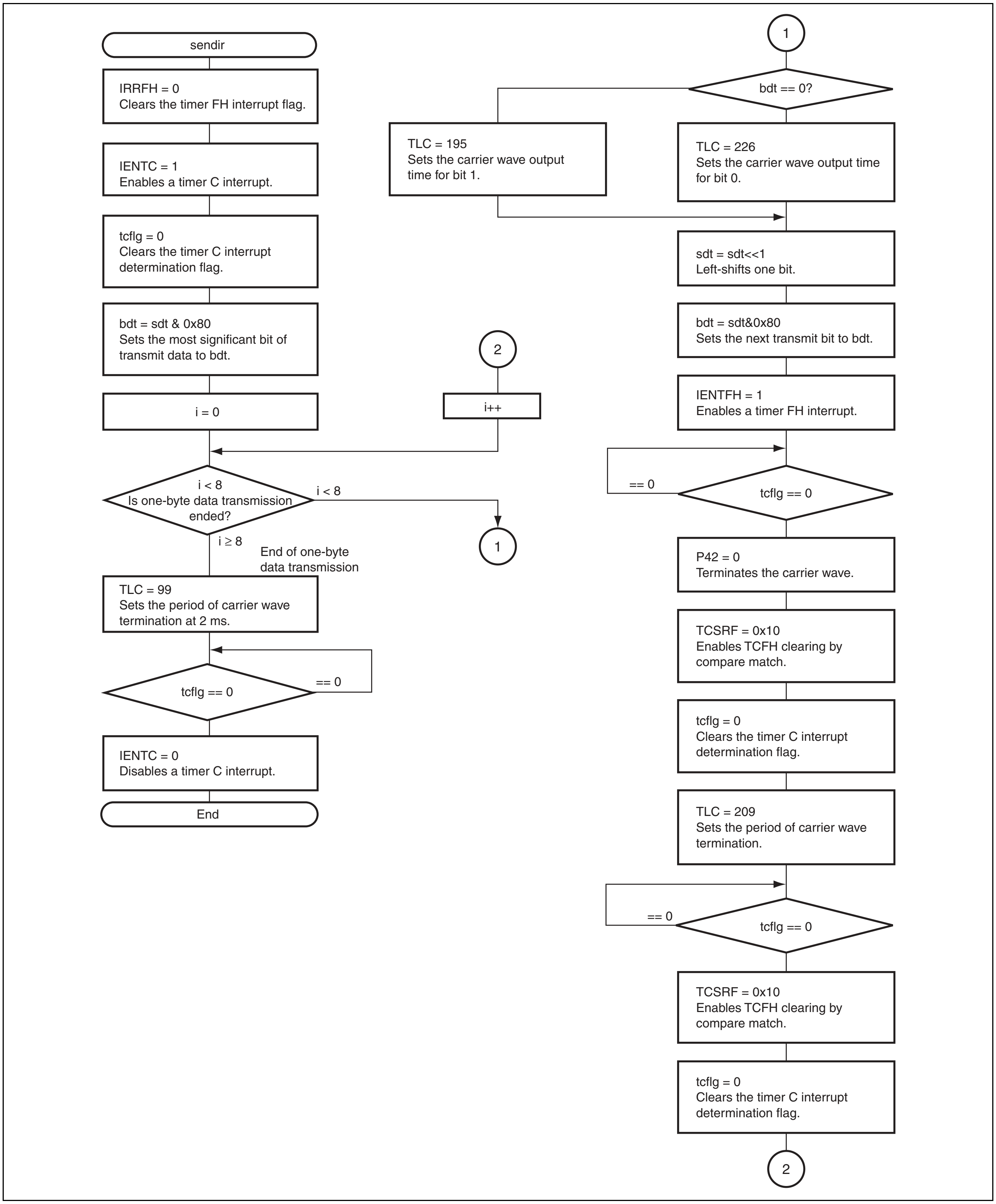
IRRTC = 0
Clears the timer C interrupt request flag.

IRRTFH = 0
Clears the timer FH interrupt request flag.

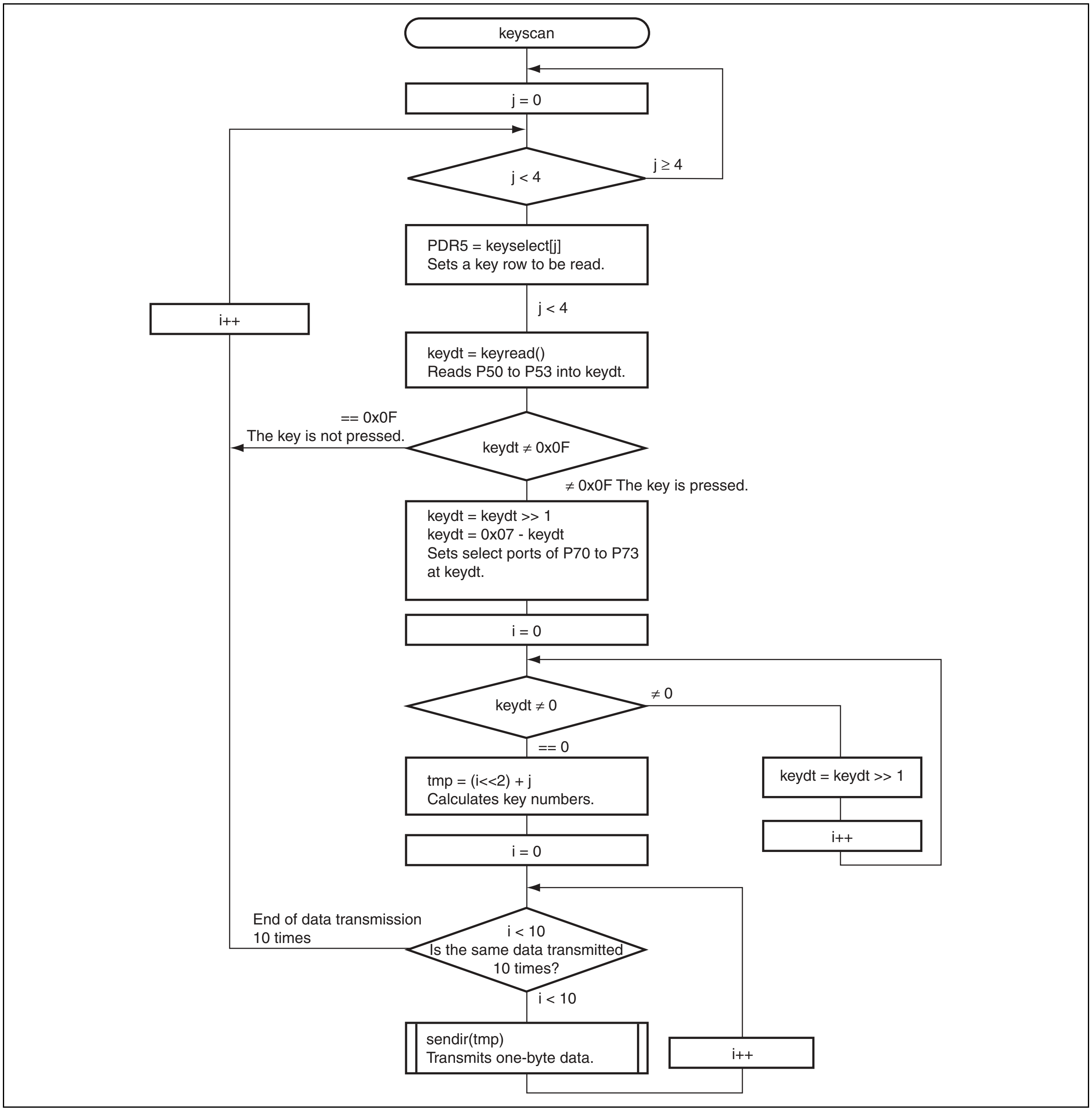
IENTC = 0
Disables a timer C interrupt.

IENTFH = 0
Disables a timer FH interrupt.

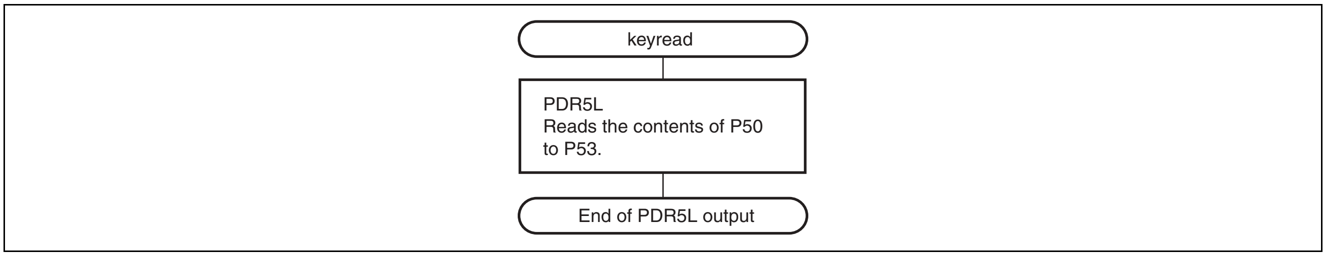
3. Remote control transmit processing


sendir
IRRFH = 0
Clears the timer FH interrupt flag.
IENTC = 1
Enables a timer C interrupt.
tcflg = 0
Clears the timer C interrupt determination flag.
bdt = sdt & 0x80
Sets the most significant bit of transmit data to bdt.
i = 0
2
i++
i < 8
Is one-byte data transmission ended?
i < 8
1
i ≥ 8
End of one-byte data transmission
TLC = 99
Sets the period of carrier wave termination at 2 ms.
tcflg == 0
== 0
IENTC = 0
Disables a timer C interrupt.
End
1
bdt == 0?
TLC = 195
Sets the carrier wave output time for bit 1.
TLC = 226
Sets the carrier wave output time for bit 0.
sdt = sdt<<1
Left-shifts one bit.
bdt = sdt&0x80
Sets the next transmit bit to bdt.
IENTFH = 1
Enables a timer FH interrupt.
== 0
tcflg == 0
P42 = 0
Terminates the carrier wave.
TCSRF = 0x10
Enables TCFH clearing by compare match.
tcflg = 0
Clears the timer C interrupt determination flag.
TLC = 209
Sets the period of carrier wave termination.
== 0
tcflg == 0
TCSRF = 0x10
Enables TCFH clearing by compare match.
tcflg = 0
Clears the timer C interrupt determination flag.
2

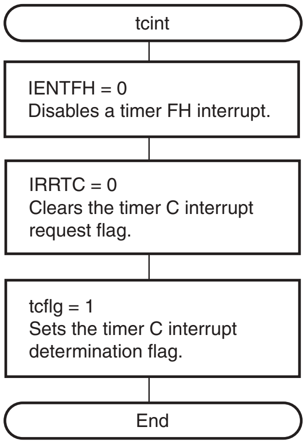
4. Key scan



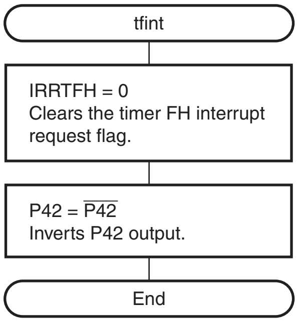
5. Port 5L reading



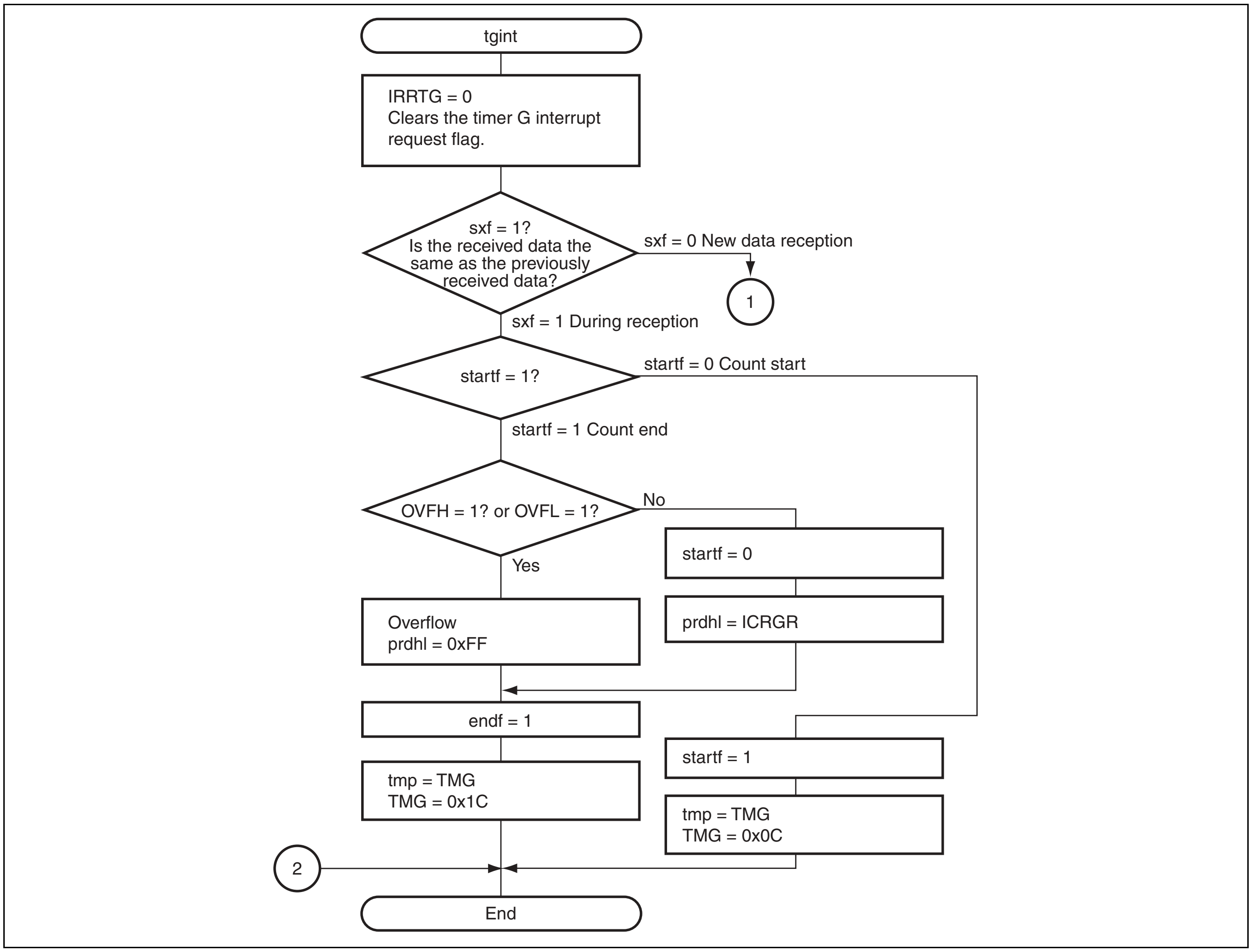
6. Timer C interrupt

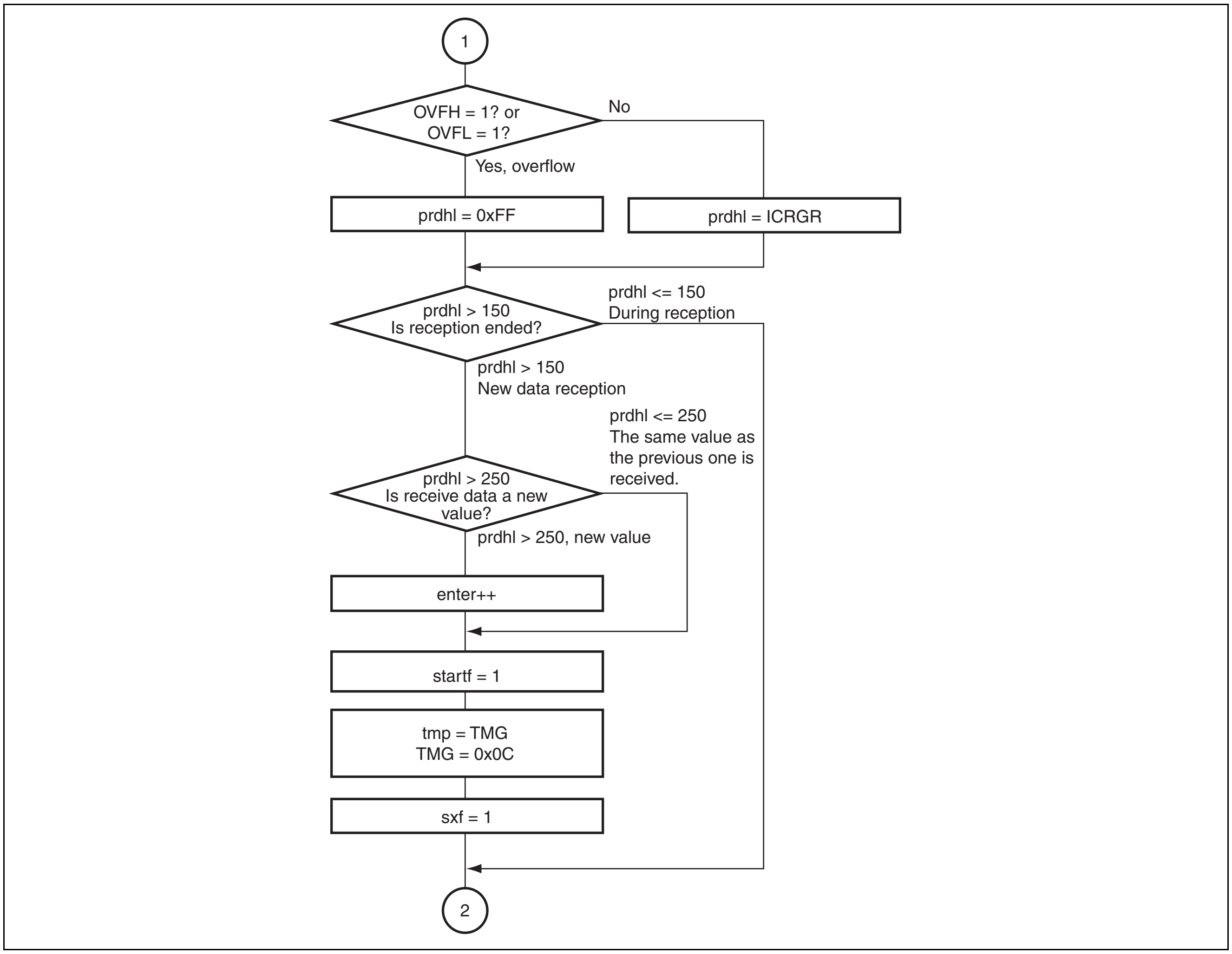

tcint
IENTFH = 0
Disables a timer FH interrupt.
IRRTC = 0
Clears the timer C interrupt request flag.
tcflg = 1
Sets the timer C interrupt determination flag.
End

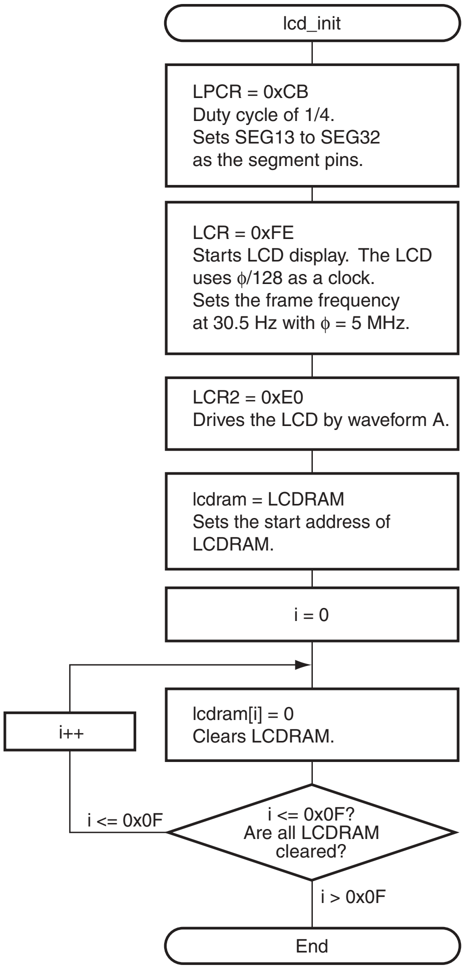
7. Timer F interrupt


tfint
IRRTFH = 0
Clears the timer FH interrupt request flag.
P42 = $\overline{\text{P42}}$
Inverts P42 output.
End

8. Timer G interrupt




1
OVFH = 1? or
OVFL = 1?
No
Yes, overflow
prdhl = 0xFF
prdhl = ICRGR
prdhl > 150
Is reception ended?
prdhl <= 150
During reception
prdhl > 150
New data reception
prdhl <= 250
The same value as
the previous one is
received.
prdhl > 250
Is receive data a new
value?
prdhl > 250, new value
enter++
startf = 1
tmp = TMG
TMG = 0x0C
sxf = 1
2


lcd_init
LPCR = 0xCB
Duty cycle of 1/4.
Sets SEG13 to SEG32
as the segment pins.
LCR = 0xFE
Starts LCD display. The LCD
uses ϕ/128 as a clock.
Sets the frame frequency
at 30.5 Hz with ϕ = 5 MHz.
LCR2 = 0xE0
Drives the LCD by waveform A.
lcdram = LCDRAM
Sets the start address of
LCDRAM.
i = 0
lcdram[i] = 0
Clears LCDRAM.
i++
i <= 0x0F
i <= 0x0F?
Are all LCDRAM
cleared?
i > 0x0F
End

7. Program Listing

```
INIT.SRC (Program list)
```

```
       .export  _INIT
       .import  _main
   ;
       .section P,CODE
   _INIT:
       mov.w    #h'ff80,r7
       ldc.b    #b'10000000,ccr
       jmp @_main
   ;
       .end
```

```
/*************************************************************************************************************/
/*                                                                                                           */
/*      H8/300L Super Low Power Series                                                                       */
/*      -H8/38024 Series-                                                                                    */
/*      Application Note                                                                                     */
/*                                                                                                           */
/*      'Infrared radiation Send/Receive Function'                                                           */
/*                                                                                                           */
/*  Function                                                                                                 */
/*  :Timer C Auto-reload Timer                                                                               */
/*  :Timer FH 8-bit Timer                                                                                    */
/*  :Timer G Input capture Timer                                                                             */
/*                                                                                                           */
/*  External Clock : 10MHz                                                                                   */
/*  Internal Clock : 5MHz                                                                                    */
/*  Sub Clock      : 32.768kHz                                                                               */
/*                                                                                                           */
/*************************************************************************************************************/

#include    <machine.h>

/*************************************************************************************************************/
/*  Symbol Definition                                                                                        */
/*************************************************************************************************************/
struct BIT {
    unsigned char   b7:1;       /* bit7 */
    unsigned char   b6:1;       /* bit6 */
    unsigned char   b5:1;       /* bit5 */
    unsigned char   b4:1;       /* bit4 */
    unsigned char   b3:1;       /* bit3 */
    unsigned char   b2:1;       /* bit2 */
    unsigned char   b1:1;       /* bit1 */
    unsigned char   b0:1;       /* bit0 */
};

struct P4BIT {
    unsigned char   H:4;        /* bit7-bit4 */
    unsigned char   L:4;        /* bit3-bit0 */
};

#define     TMC         *(volatile unsigned char *)0xFFB4        /* Timer Mode Register C                    */
#define     TCC         *(volatile unsigned char *)0xFFB5        /* Timer Counter C                          */
#define     TLC         *(volatile unsigned char *)0xFFB5        /* Timer Load Register C                    */
#define     TCRF        *(volatile unsigned char *)0xFFB6        /* Timer Control Register F                 */
```

```
#define     TCRF_BIT    (*(struct BIT *)0xFFB6)                  /* Timer Control Register F                 */
#define     TOLH        TCRF_BIT.b7                              /* Toggle Output Level F                    */
#define     CKSH2       TCRF_BIT.b6                              /* Clock Select H2                          */
#define     CKSH1       TCRF_BIT.b5                              /* Clock Select H1                          */
#define     CKSH0       TCRF_BIT.b4                              /* Clock Select H0                          */
#define     TCSRF       *(volatile unsigned char *)0xFFB7        /* Timer Control Status Register F          */
#define     TCSRF_BIT   (*(struct BIT *)0xFFB7)                  /* Timer Control Status Register F          */
#define     TFOVFH      TCSRF_BIT.b7                             /* Timer Overflow Flag H                    */
#define     CMFH        TCSRF_BIT.b6                             /* Compare Match Flag H                     */
#define     OVIEH       TCSRF_BIT.b5                             /* Timer Overflow Interrupt Enable H        */
#define     CCLRH       TCSRF_BIT.b4                             /* Counter clear H                          */
#define     TCFH        *(volatile unsigned char *)0xFFB8        /* Timer Counter FH                         */
#define     OCRFH       *(volatile unsigned char *)0xFFBA        /* Output Compare Register FH               */
#define     PMR5        *(volatile unsigned char *)0xFFCC        /* Port mode register 5                     */
#define     PDR4_BIT    (*(struct BIT *)0xFFD7)                  /* Port data register 4                     */
#define     P42         PDR4_BIT.b2                              /* P42                                      */
#define     PDR5        *(volatile unsigned char *)0xFFD8        /* Port data register 5                     */
#define     PDR5_BIT    (*(struct P4BIT *)0xFFD8)                /* Port mode register 5                     */
#define     PDR5H       PDR5_BIT.H                               /* P57-P54                                  */
#define     PDR5L       PDR5_BIT.L                               /* P53-P50                                  */
#define     PUCR5       *(volatile unsigned char *)0xFFE2        /* Port pull-up control register 5          */
#define     PCR4        *(volatile unsigned char *)0xFFE7        /* Port control register 4                  */
#define     PCR5        *(volatile unsigned char *)0xFFE8        /* Port control register 5                  */
#define     LCDRAM      (volatile unsigned char *)0xF740         /* LCD RAM                                  */
#define     TMG         *(volatile unsigned char *)0xFFBC        /* Timer Mode Register G                    */
#define     TMG_BIT     (*(struct BIT *)0xFFBC)                  /* Timer Mode Register G                    */
#define     TGOVFH      TMG_BIT.b7                               /* Timer Overflow Flag H                    */
#define     TGOVFL      TMG_BIT.b6                               /* Timer Overflow Flag L                    */
#define     OVIE        TMG_BIT.b5                               /* Timer Overflow Interrupt Enable          */
#define     IIEGS       TMG_BIT.b4                               /* Input Caputure Interrupt Edge Select     */
#define     CCLR1       TMG_BIT.b3                               /* Counter Clear 1                          */
#define     CCLR0       TMG_BIT.b2                               /* Counter Clear 0                          */
#define     CKS1        TMG_BIT.b1                               /* Clock Select 1                           */
#define     CKS0        TMG_BIT.b0                               /* Clock Select 0                           */
#define     ICRGF       *(volatile unsigned char *)0xFFBD        /* Input Caputure Register GF               */
#define     ICRGR       *(volatile unsigned char *)0xFFBE        /* Input Caputure Register GR               */
#define     LPCR        *(volatile unsigned char *)0xFFC0        /* LCD Port Control Register                */
#define     LCR         *(volatile unsigned char *)0xFFC1        /* LCD Control Register                     */
#define     LCR2        *(volatile unsigned char *)0xFFC2        /* LCD Control Register 2                   */
#define     PMR1        *(volatile unsigned char *)0xFFC8        /* Port Mode Register 1                     */
#define     PMR1_BIT    (*(struct BIT *)0xFFC8)                  /* Port Mode Register 1                     */
#define     TMIG        PMR1_BIT.b3                              /* P13/TMIG Input Select                    */
#define     PMR2        *(volatile unsigned char *)0xFFC9        /* Port Mode Register 2                     */
#define     PMR2_BIT    (*(struct BIT *)0xFFC9)                  /* Port Mode Register 2                     */
#define     NCS         PMR2_BIT.b1                              /* TMIG noise canceler select               */
#define     IENR2       *(volatile unsigned char *)0xFFF4        /* Interrupt Enable Register 2              */
#define     IENR2_BIT   (*(struct BIT *)0xFFF4)                  /* Interrupt Enable Register 2              */
#define     IENTG       IENR2_BIT.b4                             /* Timer G Interrupt Enable                 */
#define     IENTFH      IENR2_BIT.b3                             /* Timer FH Interrupt Enable                */
#define     IENTC       IENR2_BIT.b1                             /* Timer C Interrupt Enable                 */
#define     IRR2_BIT    (*(struct BIT *)0xFFF7)                  /* Interrupt Request Register 2             */
#define     IRRTG       IRR2_BIT.b4                              /* Timer G Interrupt Request Flag           */
#define     IRRTFH      IRR2_BIT.b3                              /* Timer FH Interrupt Request Flag          */
#define     IRRTC       IRR2_BIT.b1                              /* Timer C Interrupt Request Flag           */

#pragma interrupt      (tcint)
#pragma interrupt      (tfint)
#pragma interrupt      (tgint)
```

```
/**************************************************************************************************************/
/*  Function define                                                                                           */
/**************************************************************************************************************/
extern void INIT( void );                                        /* SP Set                                    */
void main( void );
void sendir_init( void );
void sendir( unsigned char sdt );
void keyscan( void );
unsigned char keyread( void );
void tcint( void );
void tfint( void );
void tgint( void );
void lcd_init( void );

/**************************************************************************************************************/
/*  RAM define                                                                                                */
/**************************************************************************************************************/
volatile unsigned char  tcflg;                                   /* Timer C Interrupt Flag                    */
unsigned char   prdhl,sxf,enter;                                 /* Timer G Interrupt Flag                    */
unsigned char   startf;                                          /* Timer G Count Start Flag                  */
unsigned char   endf;                                            /* Timer G Count End Flag                    */

unsigned char keyselect[4] = {
    0xE0,
    0xD0,
    0xB0,
    0x70,
};

unsigned char lcdtable[17] = {                                   /* LCD Key Select Table                      */
    0x07,   /* 7 */
    0xF7,   /* 8 */
    0xB7,   /* 9 */
    0x06,   /* 1 */
    0xE3,   /* 2 */
    0xA7,   /* 3 */
    0x36,   /* 4 */
    0xB5,   /* 5 */
    0xF5,   /* 6 */
    0x77,   /* A */
    0xF4,   /* B */
    0xD1,   /* C */
    0xE6,   /* D */
    0xF1,   /* E */
    0x71,   /* F */
    0x00,   /*   */
};

/**************************************************************************************************************/
/*  Vector Address                                                                                            */
/**************************************************************************************************************/
#pragma section     V1                                           /* Vector Section Set                        */
void (*const VEC_TBL1[])(void) = {                               /* 0x0000 - 0x000F                           */
    INIT                                                         /* 0x0000 Reset Vector                       */
};
#pragma section     V2                                           /* Vector Section Set                        */
void (*const VEC_TBL2[])(void) = {
    tcint                                                        /* 0x001A Timer C Interrupt Vector           */
};
```

```
#pragma section     V3                                          /* Vector Section Set                       */
void (*const VEC_TBL3[])(void) = {
    tfint                                                       /* 0x001E Timer F Interrupt Vector          */
};
#pragma section     V4                                          /* Vector Section Set                       */
void (*const VEC_TBL4[])(void) = {
    tgint                                                       /* 0x0020 Timer G Interrupt Vector          */
};

#pragma section                                                 /* P                                        */
/************************************************************************************************************/
/*  Main Program                                                                                            */
/************************************************************************************************************/
void main( void )
{
    unsigned char i,j,tx[8],rcvbuf[8],tmp,bdt;
    unsigned char *lcdram;

    set_imask_ccr(1);                                           /* Interrupt Disable                        */

    sendir_init();

    NCS = 0;                                                    /* No noise cancellation circuit            */
    TMIG = 1;                                                   /* P13/TMIG input select                    */
    tmp = TMG;                                                  /* Dummy Read for Flag Clear                */
    TMG  = 0x1C;                                                /* TMG Set                                  */

    prdhl = 0;                                                  /* Caputure Data Ram Clear                  */
    enter = 0;                                                  /* Enter flag Clear                         */

lcd_init();                                                     /* Initialize LCD                           */
    lcdram = LCDRAM + 0x0006;                                   /* Set LCDRAM Address                       */
    for(i = 0; i < 8; i++){
        rcvbuf[i] = 16;
    }

    IRRTG = 0;                                                  /* Clear IRRTG                              */
    IENTG = 0;                                                  /* Timer G Interrupt Disable                */

    while(1){
        endf = 0;                                               /* Timer G Interrupt End Flag Clear         */
        bdt = 0;                                                /* Data Buffer Clear                        */

        set_imask_ccr(0);                                       /* Interrupt Enable                         */
        for(j = 0; j < 5; j++){
            startf = 0;                                         /* Timer G Interrupt Start Flag Clear       */
            sxf = 0;                                            /* Flag Clear                               */

            tmp = TMG;                                          /* Dummy Read for Flag Clear                */
            TMG  = 0x1C;                                        /* Timer Mode Register Set                  */
            IENTG = 1;                                          /* Timer G Interrupt Enable                 */
            for(i = 0; i < 8; i++){                             /* Receive 1byte?                           */
                while(endf != 1){                               /* endf = 1 ?                               */
                    IENTG = 0;                                  /* Timer G Interrupt Disable                */
                    keyscan();                                  /* Keyscan and output KeyNo to IR           */
                    IENTG = 1;                                  /* Timer G Interrupt Enable                 */
                }
                endf = 0;
                tx[i] = prdhl;                                  /* Save 1bit Receive                        */
            }
            IENTG = 0;                                           /* Timer G Interrupt Disable                 */

            tmp = 0;
            for(i = 0; i < 8; i++){                              /* Change 1byte Data                         */
                tmp = tmp<<1;
                if(tx[i] > 45){
                    tmp++;
                }
            }

            if(bdt != tmp){                                      /* Is Receive Data same past Data?           */
                j = 0;                                           /* Receive Data Error                        */
            }

            bdt = tmp;
        }
        set_imask_ccr(1);                                        /* Interrupt Disable                         */

        if(enter >= 2){                                          /* First Data?                               */
            enter = 1;                                           /* Renew Data                                */
            for(i = 7; i > 0; i--){
                rcvbuf[i] = rcvbuf[i-1];                         /* move a figure 1 place to the left         */
            }
            rcvbuf[0] = bdt;                                     /* Set Renew Data                            */

            for(i = 0; i < 8; i++){
                lcdram[i] = lcdtable[rcvbuf[i]];                 /* Copy Renew Data -> LCDRAM                 */
            }
        }
        else{
            rcvbuf[0] = bdt;                                     /* First Data                                */
            lcdram[0] = lcdtable[bdt];                           /* A/D Data 3 figures on LCD                 */
        }
    }
}

/**************************************************************************************************************/
/*  Infared radiation Send Initialize                                                                         */
/**************************************************************************************************************/
void sendir_init( void )
{
    PMR5 = 0x00;                                                 /* Pin function Select Port5                 */
    PCR5 = 0xF0;                                                 /* P57-54 Output,P53-50 Input Port           */
    PDR5 = 0xF0;                                                 /* P57-54 Port "1"set                        */
    PUCR5 = 0x00;                                                /* Port5 pull-up OFF                         */

    TMC = 0x9B;                                                  /* Select Auto-reload Timer                  */
    TLC = 0x00;                                                  /* Clear TCC                                 */
    PCR4 = 0xFC;                                                 /* Set P42 Output Pin                        */

    TCRF   = 0x60;                                               /* Select Timer FH, phi/4                    */
    TCSRF  = 0x10;                                               /* TCFH clearing by compare match            */
    OCRFH  = 0x10;                                               /* Set Interrupt time is 26us                */

    P42 = 0;                                                     /* P42 Output Low level                      */
    tcflg = 0;

    IRRTC = 0;                                                   /* Clear IRRTC                               */
    IRRTFH = 0;                                                   /* Clear IRRTFH                              */
    IENTC = 0;                                                    /* Timer C Interrupt Disable                 */
    IENTFH = 0;                                                   /* Timer FH Interrupt Disable                */
}

/***************************************************************************************************************/
/*  Infared radiation Send                                                                                     */
/***************************************************************************************************************/
void sendir( unsigned char sdt )
{
    unsigned char bdt,i;

    IRRTFH = 0;                                                   /* Clear IRRTFH                              */
    IENTC = 1;                                                    /* Timer C Interrupt Enable                  */

    tcflg = 0;
    bdt = sdt&0x80;                                               /* Set Send top bit to bdt                   */
    for(i = 0; i < 8; i++){                                       /* Send 1 byte                               */
        if(bdt == 0)
            TLC = 226;                                            /* Set bit-0 output time                     */
        else
            TLC = 195;                                            /* Set bit-1 output time                     */

        sdt = sdt<<1;                                             /* Set Next bit to bdt                       */
        bdt = sdt&0x80;
        IENTFH = 1;                                               /* Timer FH Interrupt Enable                 */
        while(tcflg == 0);                                        /* 1-bit is Sending                          */
        P42 = 0;                                                  /* P42 is Low level                          */
        TCSRF  = 0x10;                                            /* Initialize Overflow Interrupt             */

        tcflg = 0;
        TLC = 209;                                                /* Set P42 Low level time                    */
        while(tcflg == 0);
        TCSRF = 0x10;                                             /* Initialize Overflow Interrupt             */
        tcflg = 0;
    }

    TLC = 99;                                                     /* Wait Over 2ms                             */
    while(tcflg == 0);

    IENTC = 0;                                                    /* Timer C Interrupt Disable                 */
}

/***************************************************************************************************************/
/*  KeyScan                                                                                                    */
/***************************************************************************************************************/
void keyscan( void )
{
    unsigned char tmp,i,j,keydt;

    for(j = 0; j < 4; j++){
        PDR5 = keyselect[j];                                      /* Set Key Select                            */

        keydt = keyread();
        if(keydt != 0x0F){                                        /* Touch Key?                                */
            keydt = keydt>>1;                                     /* What Key?                                 */
            keydt = 0x07 - keydt;
            for(i = 0; keydt != 0; i++){
                keydt = keydt>>1;
            }

            i = i<<2;
            tmp = i+j;                                            /* Set KeyNo -> tmp                         */

            for(i = 0; i < 10; i++){                              /* Same Data Output 10 time                 */
                sendir(tmp);                                      /* Send 1byte data to IR                    */
            }
        }
    }
}

/**************************************************************************************************************/
/*  KeyRead                                                                                                   */
/**************************************************************************************************************/
unsigned char keyread( void )
{
    return(PDR5L);
}

/**************************************************************************************************************/
/*  Timer C Interrupt                                                                                         */
/**************************************************************************************************************/
void tcint( void )
{
    IENTFH = 0;                                                   /* Timer FH Interrupt Disable               */
    IRRTC = 0;                                                    /* Clear IRRTC                              */
    tcflg = 1;                                                    /* Timer C Interrupt flag Set               */
}

/**************************************************************************************************************/
/*  Timer F Interrupt                                                                                         */
/**************************************************************************************************************/
void tfint( void )
{
    IRRTFH = 0;                                                   /* Clear IRRFH                              */
    P42 = ~P42;                                                   /* Toggle Output P42                        */
}

/**************************************************************************************************************/
/*  Timer G Interrupt                                                                                         */
/**************************************************************************************************************/
void tgint( void )
{
    unsigned char tmp;

    IRRTG = 0;                                                    /* Clear IRRTG                              */

    if(sxf == 1){
        if(startf == 1){
            if((TGOVFH == 1)|(TGOVFL == 1)){
                prdhl = 0xFF;
            }
            else{
                startf = 0;                                       /* Clear startf                             */
                prdhl =ICRGR;                                     /* Caputure Data Ramcopy                    */
            }
            endf = 1;                                             /* Set endf                                 */
            tmp = TMG;                                            /* Dummy Read for Flag Clear                */
            TMG = 0x1C;                                          /* Overflow Interrupt Disable               */
        }
        else{
            startf = 1;                                          /* Set startf                               */
            tmp = TMG;                                           /* Dummy Read for Flag Clear                */
            TMG = 0x0C;                                          /* Timer Mode Register Set                  */
        }
    }
    else{
        if((TGOVFH == 1)|(TGOVFL == 1)){
            prdhl = 0xFF;                                        /* Not Receive Data                         */
        }
        else{
            prdhl =ICRGF;                                        /* Caputure Data Ramcopy                    */
        }

        if(prdhl > 150){
            if(prdhl > 250){
                enter++;                                         /* First Data or Renew Data                 */
            }
            startf = 1;                                          /* Set startf                               */
            tmp = TMG;                                           /* Dummy Read for Flag Clear                */
            TMG  = 0x0C;                                         /* Timer Mode Register Set                  */
            sxf = 1;
        }
    }
}

/*************************************************************************************************************/
/*  LCD Initialize                                                                                           */
/*************************************************************************************************************/
void lcd_init( void )
{
    unsigned char i;
    unsigned char *lcdram;

    LPCR = 0xCB;                                                 /* 1/4 Duty ,Select SEG32-SEG13             */
    LCR  = 0xFE;                                                 /* LCD ON                                   */
    LCR2 = 0xE0;                                                 /* A waveform                               */

    lcdram = LCDRAM;                                             /* Set LCDRAM Address                       */
    for(i = 0; i <= 0x0F; i++){                                  /* Initialize LCD RAM                       */
        lcdram[i] = 0;
    }
}
```

Link address specifications

Section Name	Address
CV1	0x0000
CV2	0x001A
CV3	0x001E
CV4	0x0020
P, D	0x0100
B	0xFB80

Revision Record

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

Keep safety first in your circuit designs!

1. Renesas Technology Corp. puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage.
   Remember to give due consideration to safety when making your circuit designs, with appropriate measures such as (i) placement of substitutive, auxiliary circuits, (ii) use of nonflammable material or (iii) prevention against any malfunction or mishap.

Notes regarding these materials

1. These materials are intended as a reference to assist our customers in the selection of the Renesas Technology Corp. product best suited to the customer's application; they do not convey any license under any intellectual property rights, or any other rights, belonging to Renesas Technology Corp. or a third party.
2. Renesas Technology Corp. assumes no responsibility for any damage, or infringement of any third-party's rights, originating in the use of any product data, diagrams, charts, programs, algorithms, or circuit application examples contained in these materials.
3. All information contained in these materials, including product data, diagrams, charts, programs and algorithms represents information on products at the time of publication of these materials, and are subject to change by Renesas Technology Corp. without notice due to product improvements or other reasons. It is therefore recommended that customers contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor for the latest product information before purchasing a product listed herein.
   The information described here may contain technical inaccuracies or typographical errors. Renesas Technology Corp. assumes no responsibility for any damage, liability, or other loss rising from these inaccuracies or errors.
   Please also pay attention to information published by Renesas Technology Corp. by various means, including the Renesas Technology Corp. Semiconductor home page (http://www.renesas.com).
4. When using any or all of the information contained in these materials, including product data, diagrams, charts, programs, and algorithms, please be sure to evaluate all information as a total system before making a final decision on the applicability of the information and products. Renesas Technology Corp. assumes no responsibility for any damage, liability or other loss resulting from the information contained herein.
5. Renesas Technology Corp. semiconductors are not designed or manufactured for use in a device or system that is used under circumstances in which human life is potentially at stake. Please contact Renesas Technology Corp. or an authorized Renesas Technology Corp. product distributor when considering the use of a product contained herein for any specific purposes, such as apparatus or systems for transportation, vehicular, medical, aerospace, nuclear, or undersea repeater use.
6. The prior written approval of Renesas Technology Corp. is necessary to reprint or reproduce in whole or in part these materials.
7. If these products or technologies are subject to the Japanese export control restrictions, they must be exported under a license from the Japanese government and cannot be imported into a country other than the approved destination.
   Any diversion or reexport contrary to the export control laws and regulations of Japan and/or the country of destination is prohibited.
8. Please contact Renesas Technology Corp. for further details on these materials or the products contained therein.