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.

# M32C/85 Group

## Remote Control Reception

## 1. Abstract

This documents describes how to receive a remote control signal using the time measurement function of the intelligent I/O.

## 2. Introduction

The application example described in this document is applied to the following MCU and parameter(s):
MCU: M32C/85 Group

This program can be used with other M16C Family MCUs which have the same special function registers (SFRs) as the M32C/85 Group. Check the manual for any additions and modifications to functions. Careful evaluation is recommended before using this application note.

## 3. Remote Control Reception Overview

An infrared signal transmitted from the remote control is transmitted to the receiver at a certain frequency (carrier frequency). As the infrared signal is weakened through diffusion at the receiver, the output of the infrared receiving element must be amplified with a preamplifier. Also, passing though a bandpass filter (BPF) allows an accurate remote control signal to be obtained by extracting only the carrier waveform element and detecting and rectifying the waveform. In this case, the carrier frequency is set to 38 kHz.
Figure 3.1 shows the Block Diagram of Remote Control Reception Overview, Figure 3.2 shows the Internal Block Diagram of Infrared Remote Control Receive Module, and Figure 3.3 shows the Carrier Waveform.

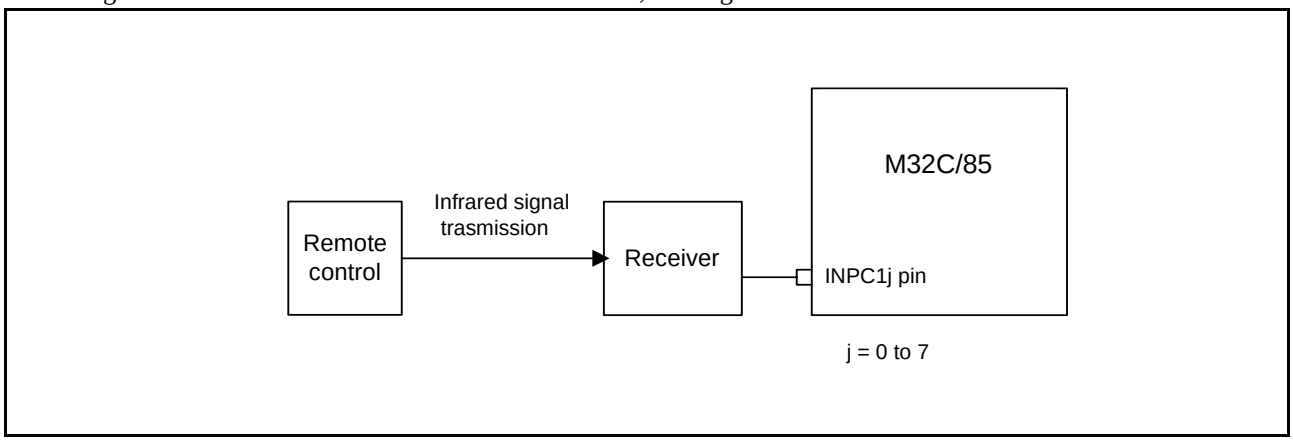



**Figure 3.1 Block Diagram of Remote Control Reception Overview**

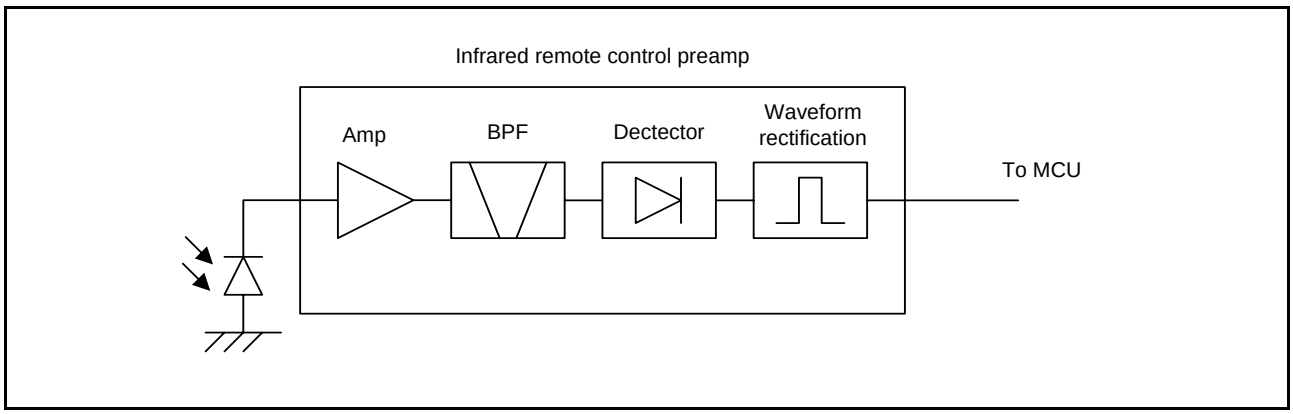
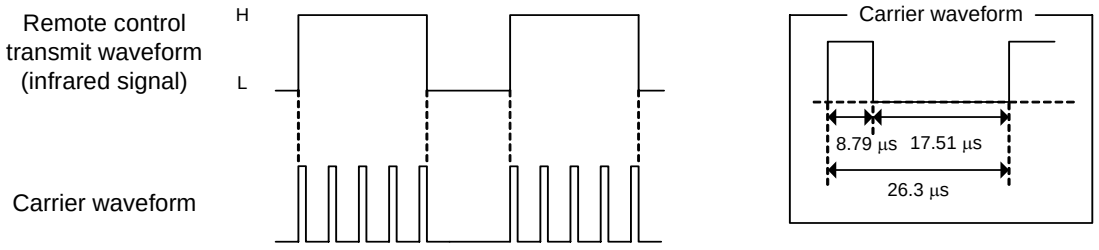



**Figure 3.2 Internal Block Diagram of Infrared Remote Control Receive Module**

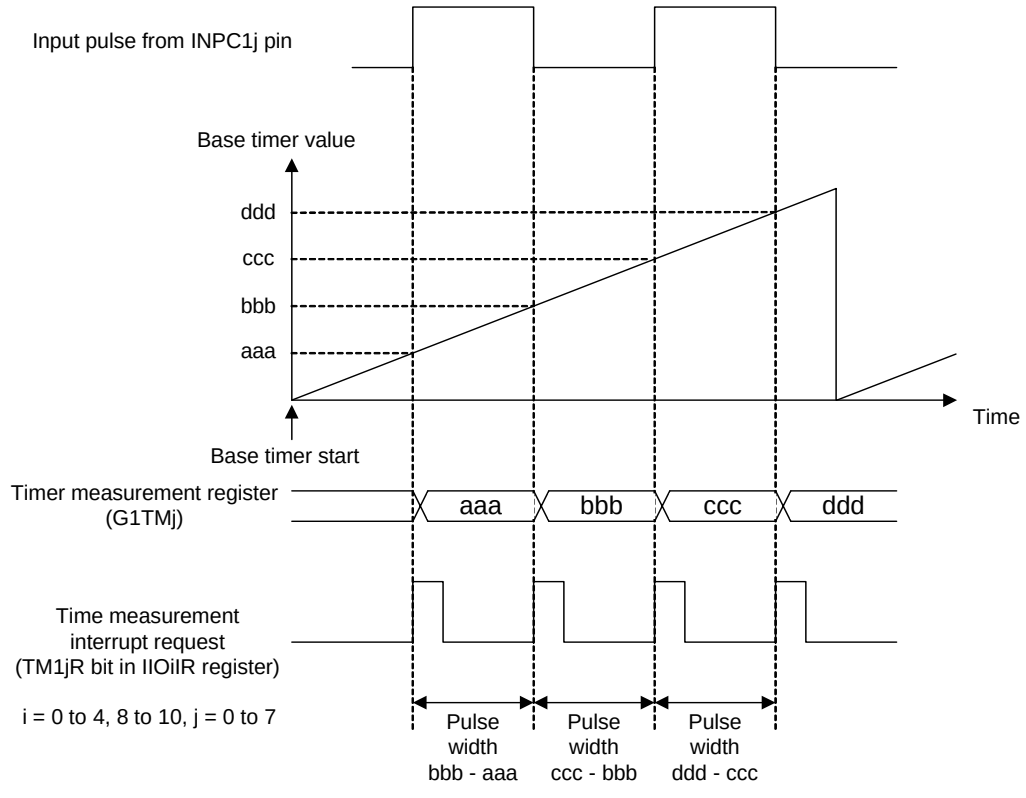



An infrared signal is not continuously transmitted throughout the entire “H” interval of the remote control transmit waveform. The interval of infrared transmission/no transmission is repeated at a certain frequency (carrier frequency).

**Figure 3.3 Carrier Waveform**

## 4. Time Measurement Function

The time measurement function of the intelligent I/O synchronizes with the external trigger input and stores the base timer value to the G1TMj register (j = 0 to 7). Figure 4.1 shows the Operational Timing of Time Measurement Function.




This applies under the following conditions:
- Time measurement trigger: Both edges
- No timer reset at the match between the G1PO0 register and the base timer
- Base timer value: Up count mode

**Figure 4.1 Operational Timing of Time Measurement Function**

# 5. Operational Description

Table 5.1 lists the Resource List.

**Table 5.1 Resource List**

| Resource | Channel | Setting Value | Application |
|---|---|---|---|
| Intelligent I/O | Time measurement 1 channel | Divide-by-62<br>Count source f1<br>Both edges trigger | Remote control transmit waveform time measurement |
| Timer | Timer A0 | 1 ms | One frame timer measurement (for timer A1 event count) |
| | Time A1 | 140 ms | One frame timer measurement |

## 5.1

- Intelligent I/O -

1. Set the base timer to divide-by-62 and the count source f1.
2. Set the timer measurement trigger for the base timer to both edges.
3. Setting the base timer start bit to 1 increments the base timer value.
4. The intelligent I/O time measurement function 0 interrupt request bit is set to 1 at the rising or falling edge of the INPC10 pin.
5. The base timer value (G1TM0) is stored at each interrupt.
6. The stored base timer value is converted to a pulse value and compared with the remote control data.

## 5.2

- Timer -

1. Use the timer to measure one frame from the leader code.
2. Set timer A0 to timer mode and timer A1 to event counter mode (timer A0 underflow counted).
3. Set the timer value to (7D00-1)h to set the timer A0 interrupt to a 1 ms interval.
4. Set the timer value to (8C-1)h to set the timer A1 interrupt to a 140 ms interval (one frame +30% error).
5. Set the timer A0 and timer A1 count start flags to 1 upon detecting the leader code and start the timer.
6. Stop timer A0 and timer A1 when the interrupt request bit is set to 1 by the timer A1 interrupt.

## 6. Remote Control Detection Performance

- For the first data, reception is determinded as complete if the following are received within 108 ms: the leader code, custom code (8 bits), inverted custom code (8 bits), data code (8 bits), inverted data code (8 bits), stop bit (1 bit), and one frame (interval including from the leader code to the frame space) of the frame space (interval of no infrared transmission).
- For the second or subsequent data, reception is determind as complete if the following are received within 108 ms: leader code, stop bit (1 bit), and one frame of the frame space.
- For each code, code recognition is determinded as complete if the error is within ±30% of the format value. The same applies to one frame if the error is within 108 ms +30 %.
- When the leader code is detected, detection takes place in the order of custom code, data code, stop bit, and frame space.
- If a receive error occurs on each code, the next rising or falling egde is determinded as the leader code (first data) and reception starts.
- When one frame (including the +30% error) or more has elasped after the leader code, if the frame space is being recognized in the received data, reception is determinded as complete.
- When the leader code after the frame space is detected within one frame (including the +30% error), the detected received data is recognized as the second or sebsequent data. (The first leader code may be received within one frame after the frame space as the +30% error is included.)
- Figure 6.1 shows the Remote Control Format Diagram.

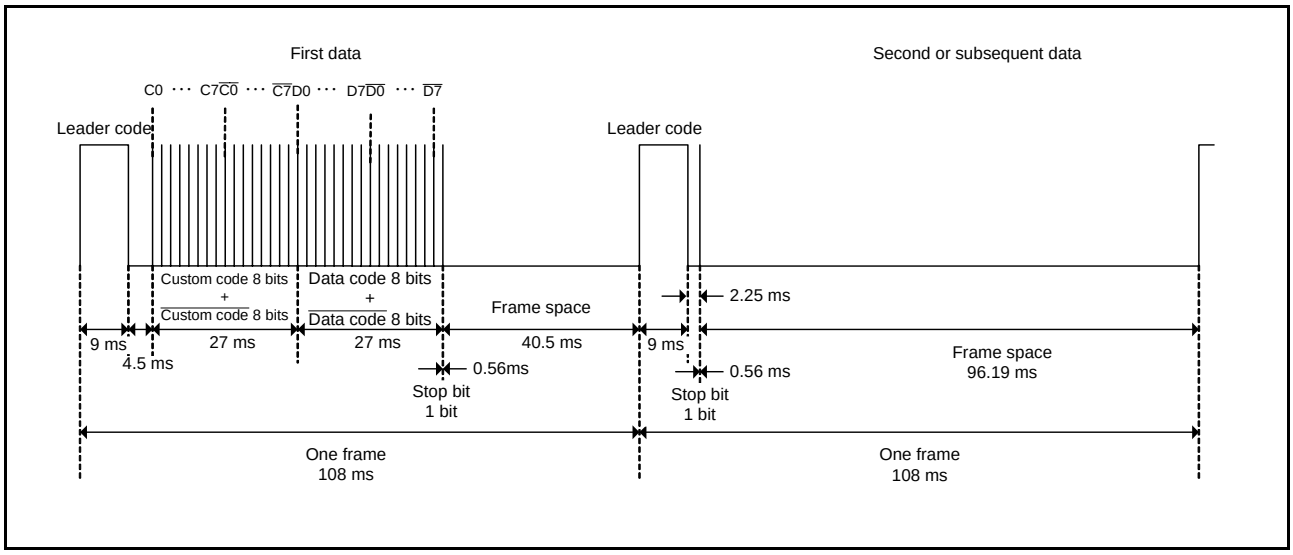



**Figure 6.1 Remote Control Format Diagram**

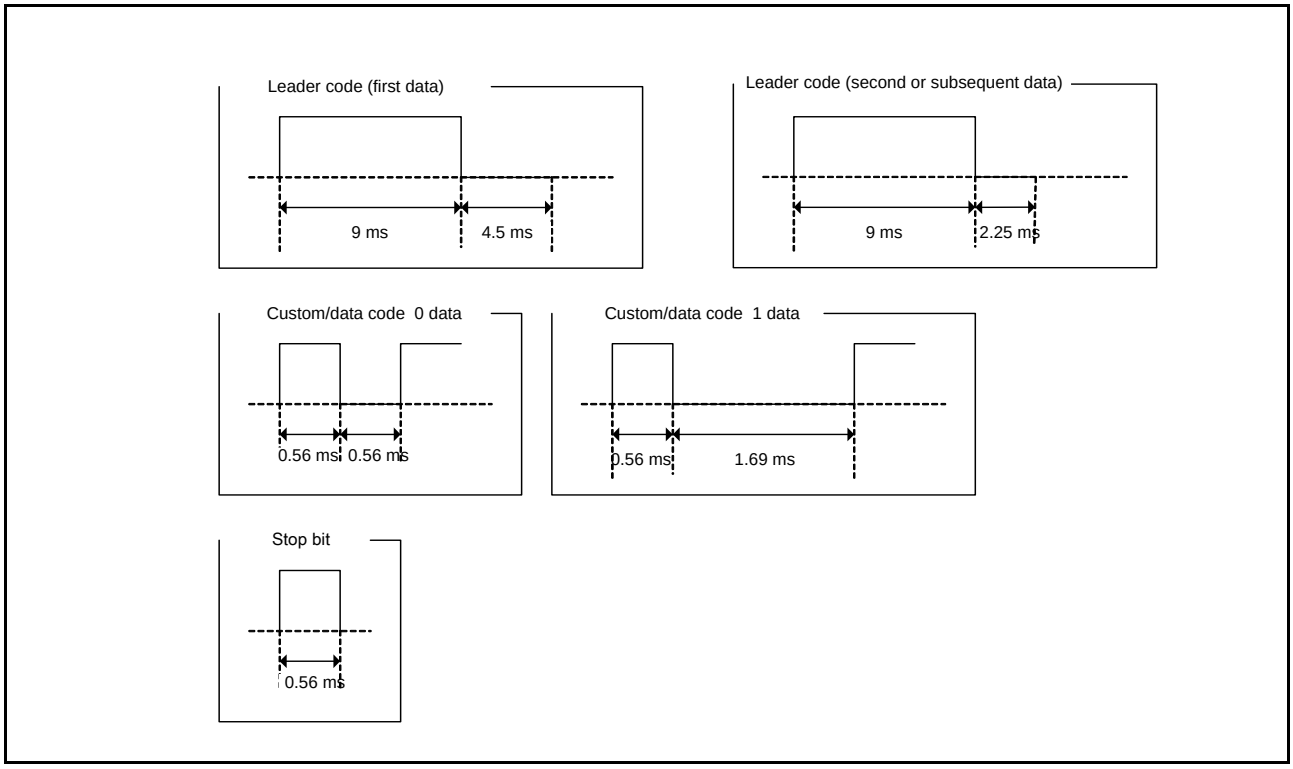



**Figure 6.2 Expanded Codes**

**Table 6.1 Recognition Range Value by Code**

| Code Name | Code Recognition Range Value |
|---|---|
| Leader code “H” | 6.3 ms to 11.69 ms |
| Leader code “L” | 3.15 ms to 5.84 ms |
| Custom code “H” | 0.393 ms to 0.726 ms |
| Custom code (0 data) “L” | 0.393 ms to 0.726 ms |
| Custom code (1 data) “L” | 1.183 ms to 2.195 ms |
| Data code “H” | 0.393 ms to 0.726 ms |
| Data code (0 data) “L” | 0.393 ms to 0.726 ms |
| Data code (1 data) “L” | 1.183 ms to 2.195 ms |
| Stop bit | 0.393 ms to 0.726 ms |
| Frame space | 28.35 ms to 52.64 ms |
| Leader code “H” (repeat) | 6.3 ms to 11.69 ms |
| Leader code “L” (repeat) | 1.575 ms to 2.923 ms |
| Stop bit (repeat) | 0.393 ms to 0.726 ms |
| Frame space (repeat) | 67.33 ms to 125.04 ms |

# 7. Software Description

## 7.1 Function Description

Table 7.1 lists the Function Description.

**Table 7.1 Function Description**

| Function Name | Label Name | Features |
|---|---|---|
| Main processing | main | Used register setting, used RAM initialization, interrupt enable, received data decision, timer decision |
| Received data setting processing | rcv_data | Received data setting process upon received data interrupt detection |
| Time over setting processing | time_over | Setup processing after one-frame time elapsed upon leader code detection |
| Pulse value setting processing | set_pulse_value | Pulse value setting |
| Received data decision processing | check_code | Received data decision/setting |
| Received data range check processing | cop_pulse | Received data range decision |
| Inverted data code decision processing | judge_reversing_code | Inverted data code decision |
| Inverted data code buffer setting processing | set_reversubg_code | Inverted data code buffer setting |
| Inverted data code comparison processing | cop_reversing_code | Inverted data code comparison |

## 7.2 Register Description

Table 7.2 lists the Register Description.

**Table 7.2 Register Description**

| Register Name | | Address | Setting Value | Feature |
|---|---|---|---|---|
| G1BCR0 | Base timer control register 10 | 0122h | 78h<br>7Bh | • Clock stop<br>• Bit 15 overflow, divide-by-62, f1 |
| G1BCR1 | Base timer control register 11 | 0123h | 00h<br>10h | • Up count mode<br>• Base timer reset/count start<br>• No base timer reset at the “L” input to the INT0 or INT1 pin<br>• No base timer reset at the match between the G1PO0 register and the base timer |
| G1TMCR0 | Time measurement control register 10 | 0118h | 03h | • No digital filter<br>• The time measurement trigger is selected as both edges. |
| G1FS | Function select register 1 | 0127h | 01h | Time measurement function for channel 0 |
| G1FE | Function enable register 1 | 0126h | 01h | Functional operation for channel 0 |
| G1TM0 | Time measurement register 10 | 0100h | - | A base timer value is stored at every time measurement timing. |
| IIO3IC | Interrupt control register | 0097h | 00h | Interrupt request clear |
| IIO3IR | Interrupt request register | 00A3h | 00h | Intelligent I/O<br>Time measurement function 0 interrupt request clear |
| IIO3IE | Interrupt enable register | 00B3h | 01h<br>05h | • Intelligent I/O time measurement function 0 interrupt enable<br>• An interrupt request is used for an interrupt.<br>*To use a interrupt request for an interrupt, set bit 0 to 1 before setting other bits to 1. |
| PS1 | Function select register A1 | 03B1h | 00h | Port P73 as an input port |
| PD7 | Port P7 direction register | 03C3h | 00h | Port P73 as an input mode |
| TABSR | Count start flag | 0340h | 00h<br>03h | Timer A0, A1 count stop/start |
| TA0MR | Timer A0 mode register | 0356h | 00h | • Count source f1<br>• No gate function<br>• Timer mode |
| TA0 | Timer A0 register | 0346h | 7CFFh | Set to 1 ms. |
| TA1MR | Timer A1 mode register | 0357h | 01h | • Reload type<br>• Event counter mode |
| TA1 | Timer A1 register | 0348h | 008Bh | Set to 140 ms (108 ms +30%). |
| TRGSR | Trigger select register | 0343h | 02h | The timer A1 event trigger selected as the timer TA0 underflow. |
| TA0IC | Interrupt control register | 006Ch | 00h | Priority level 0 |
| TA1IC | Interrupt control register | 008Ch | 00h | Priority level 0 |

## 7.3 RAM Description

Table 7.3 lists the RAM Description.

**Table 7.3 RAM Description**

| RAM Name | Feature | Data Length | Function Used |
|---|---|---|---|
| Rcv_mode | Receive mode | 1 byte | main, rcv_data,<br>time_over, check_code |
| Rcv_bit_cnt | Number of received custom/data code bit | 1 byte | main, rcv_data, check_code |
| Rcv_data_cnt | Number of received data | 1 byte | main, rcv_data,<br>time_over, check_code |
| pulse[100] | Received data buffer<br>Storage in hexdecimal | 200 bytes | time_over, set_pulse_value, check_code |
| pulse_cnt | Received data buffer position | 1 byte | main, rcv_data, time_over,<br>set_pulse_value, check_code |
| code_low_cnt | Received data code “L” counter | 1 byte | main, rcv_data, judge_reversing_code, |
| Rev_cnt | Inverted data counter | 1 byte | main, rcv_data, judge_reversing_code,<br>set_reversing_code, cmp_reversing_code |
| Rev_pulse[8] | Inverted data code confirmation buffer<br>If the code is not inverted data, it is stored to the following:<br>If 0 data: 0xF1<br>If 1 data: 0xF0 | 10 bytes | set_reversing_code, cmp_reversing_code |
| old_tr | Timer measurement capture value for next comparison | 2 bytes | set_pulse_value |

## 7.4 ROM Description

Table 7.4 lists the ROM Description.

**Table 7.4 ROM Description**

| ROM Name | Feature | Data Length | Function Used |
|---|---|---|---|
| cmp_tbl[14][2] | Received code compare table<br>• [*][0]: Format value for each interval, [*][1]: Format value ±30%<br>• [*][0]-[*][1] to [*][0]+[*][1]: Code recognition range value<br>• If an error is ±30% of a format value, recognition is OK.<br>• These range values are set as the frequency of 32 MHz divided by 62.<br><br>[0][*]: Leader code “H” interval (6.3 ms to 11.69 ms)<br>[1][*]: Leader code “L” interval (3.15 ms to 5.84 ms)<br>[2][*]: Custom code “H” interval (0.393 ms to 0.726 ms)<br>[3][*]: Custom code (0 data) “L” interval (0.393 ms to 0.726 ms)<br>[4][*]: Custom code (1 data) “L” interval (1.183 ms to 2.195 ms)<br>[5][*]: Data code “H” interval (0.393 ms to 0.726 ms)<br>[6][*]: Data code (0 data) “L” interval (0.393 ms to 0.726 ms)<br>[7][*]: Data code (1 data) “L” interval (1.183 ms to 2.195 ms)<br>[8][*]: Stop bit interval (0.393 ms to 0.726 ms)<br>[9][*]: Frame space interval (28.35 ms to 52.64 ms)<br>[10][*]: Leader code “H” interval (repeat) (6.3 ms to 11.69 ms)<br>[11][*]: Leader code “L” interval (repeat) (1.575 ms to 2.923 ms)<br>[12][*]: Stop bit interval (repeat) (0.393 ms to 0.726 ms)<br>[13][*]: Frame space interval (repeat) (67.33 ms to 125.04 ms) | 56 bytes | check_code |

## 8. Setup Procedure

In this sample program, the CPU clock is set to 32 MHz - 8 MHz multiplied by 4.

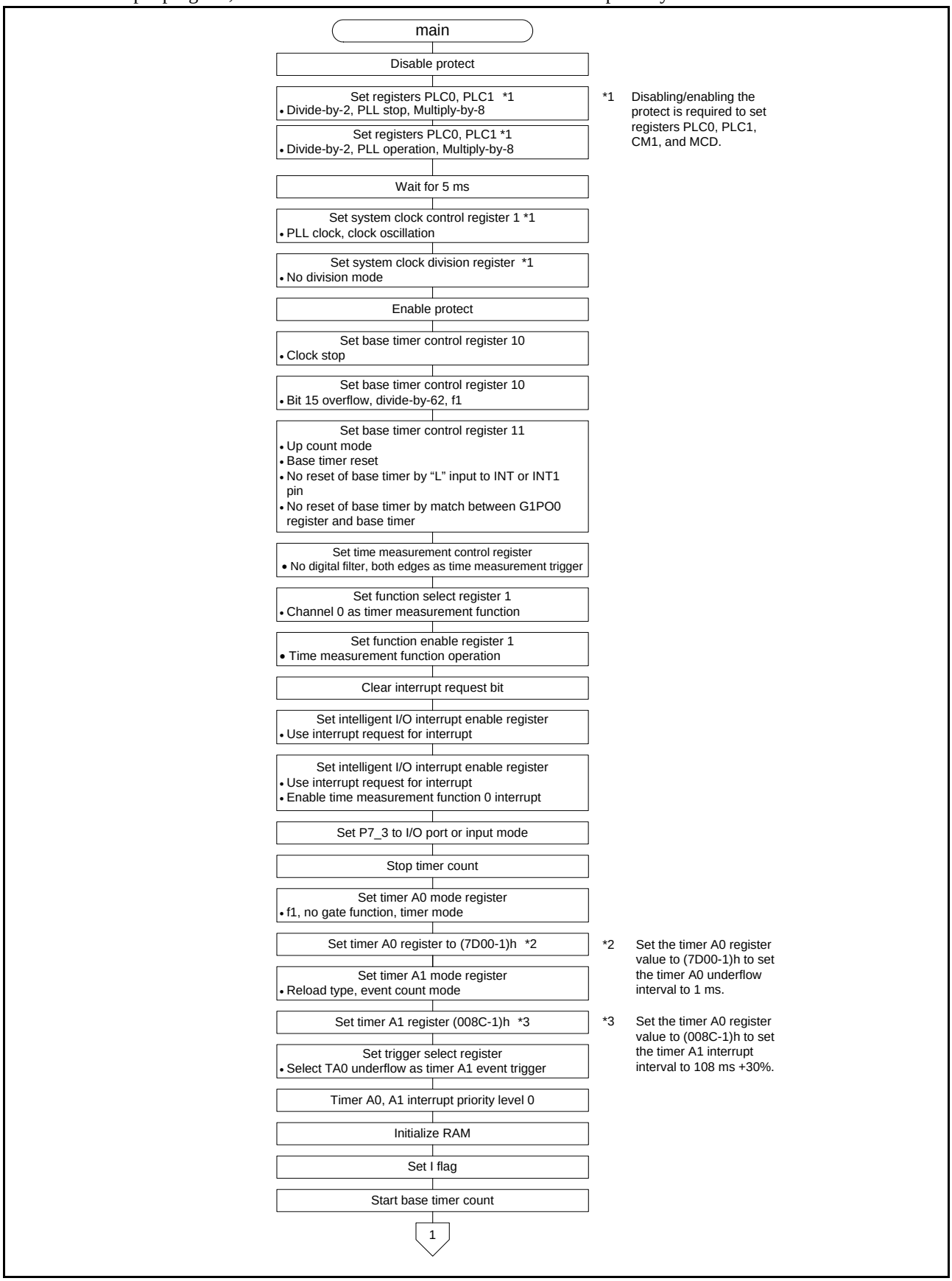



**Figure 8.1 Flowchart (main)**

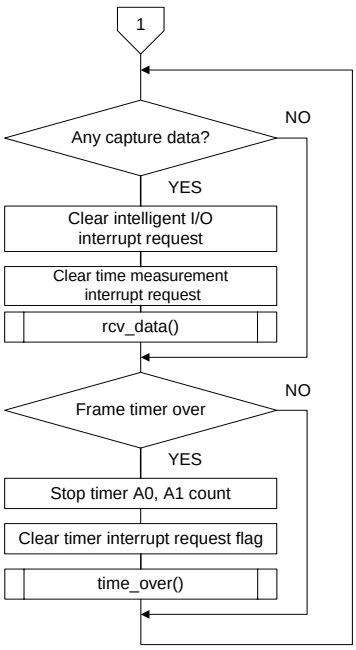



**Figure 8.2 Flowchart (main)**

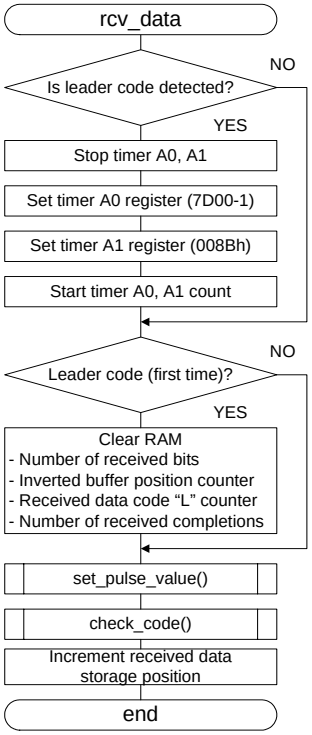



**Figure 8.3 Flowchart (rcv_data)**

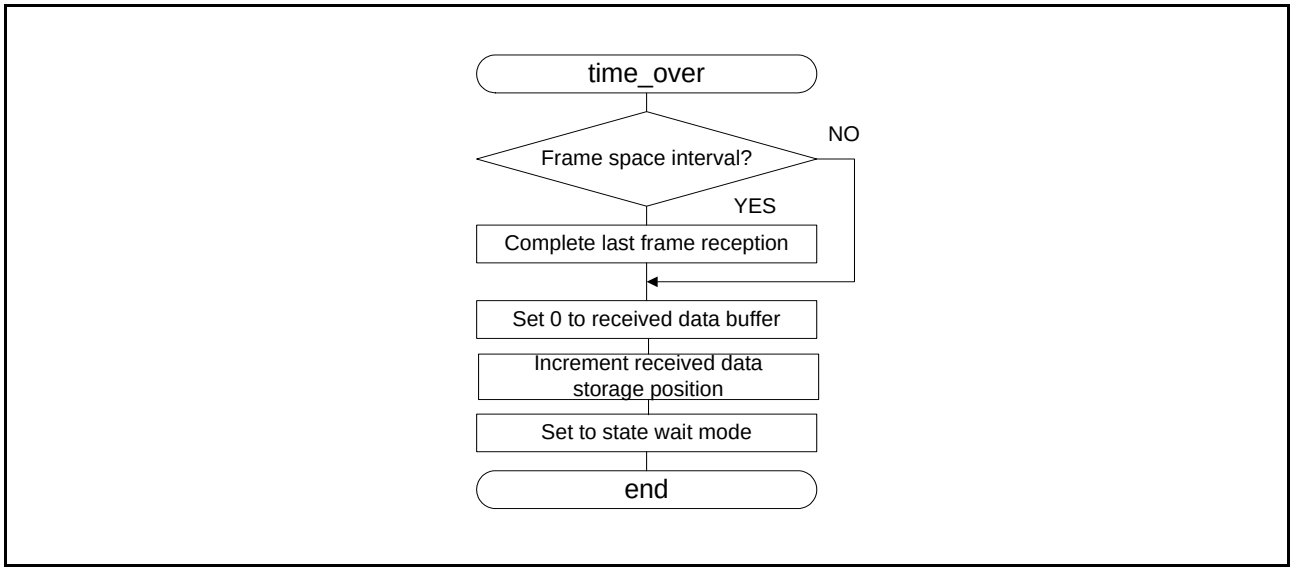



**Figure 8.4 Flowchart (time_over)**

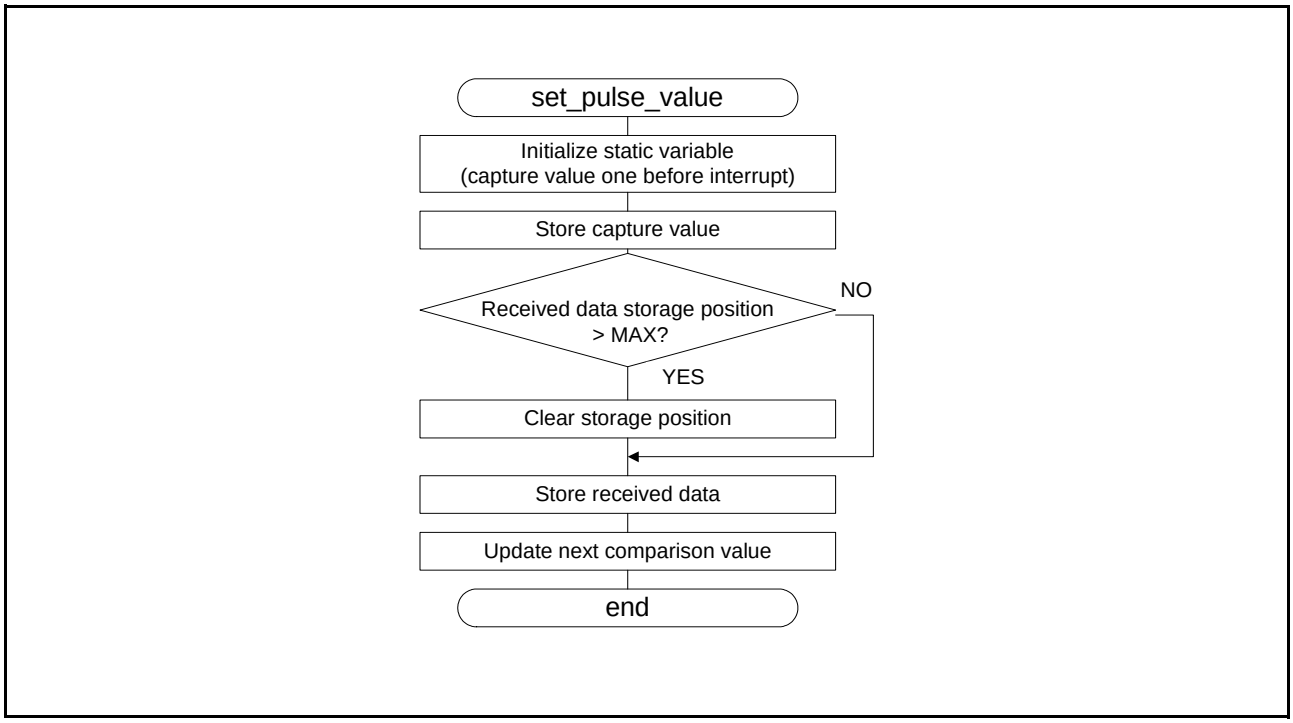



**Figure 8.5 Flowchart (set_pulse_value)**

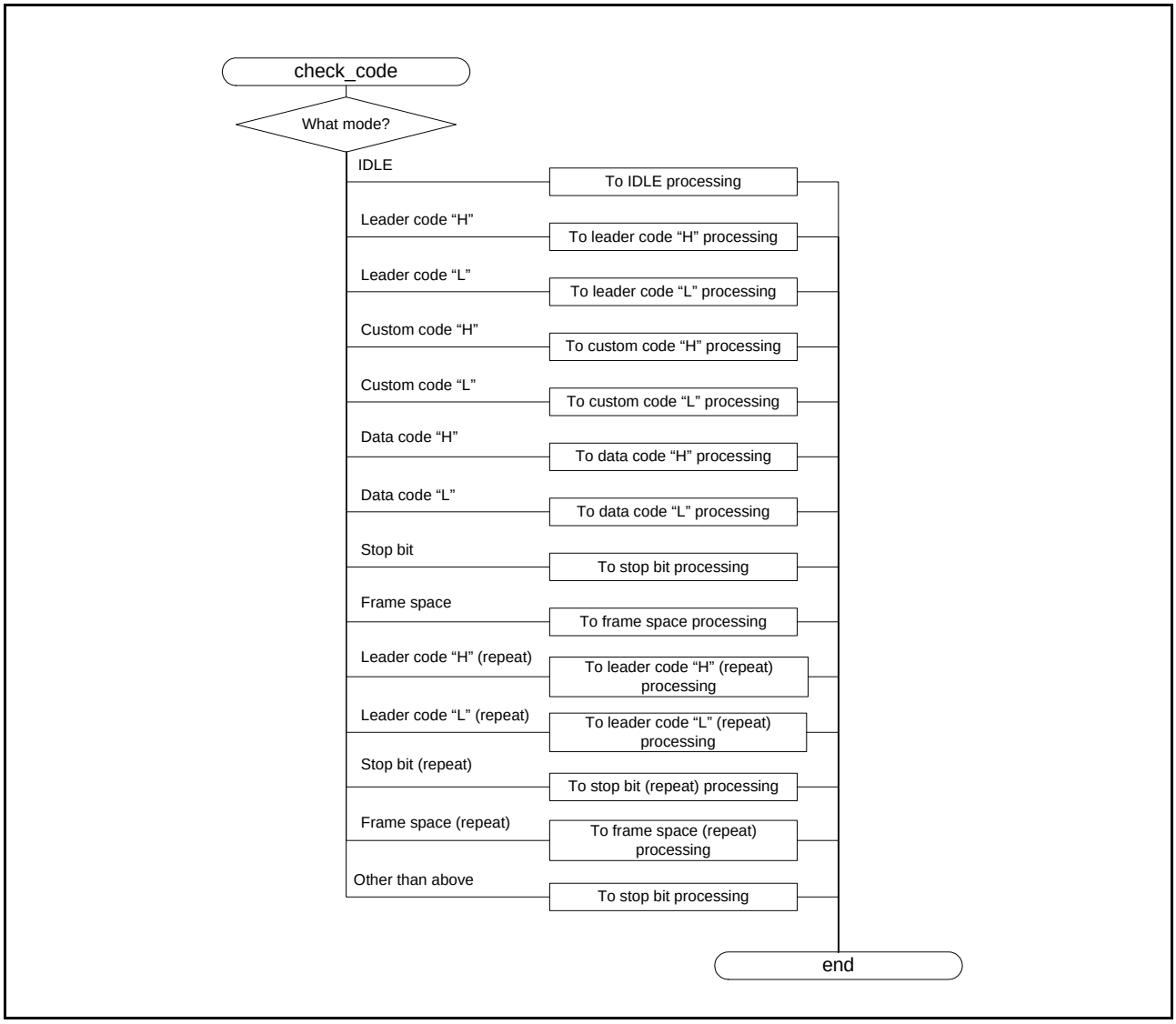



**Figure 8.6 Flowchart (check_code)**

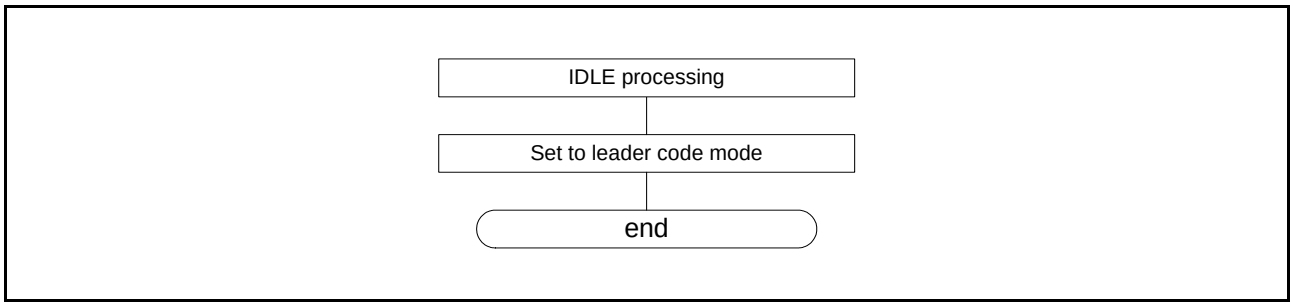



**Figure 8.7 Flowchart (check_code/IDLE)**

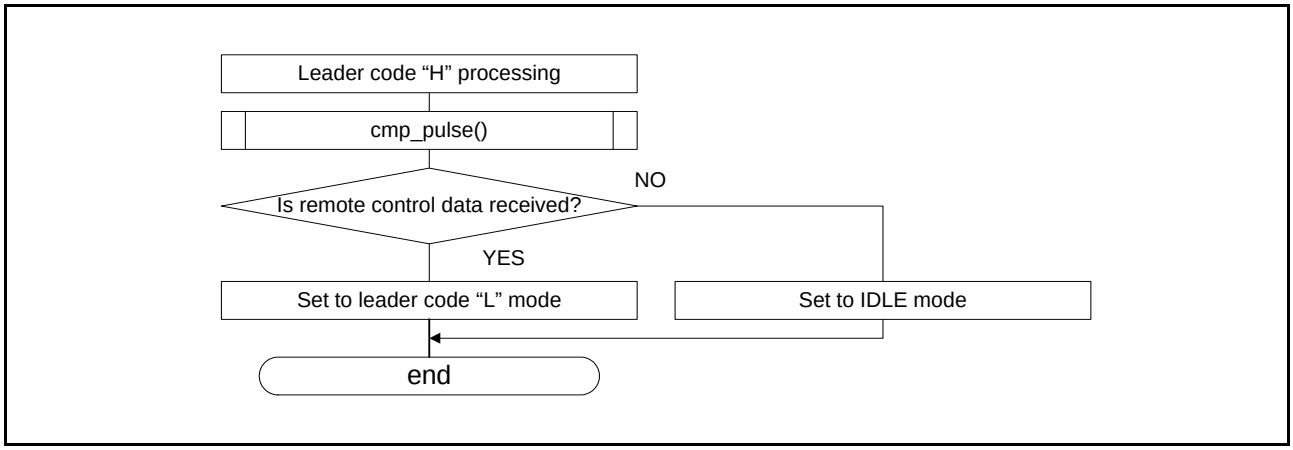



**Figure 8.8 Flowchart (check_code/leader code “H”)**

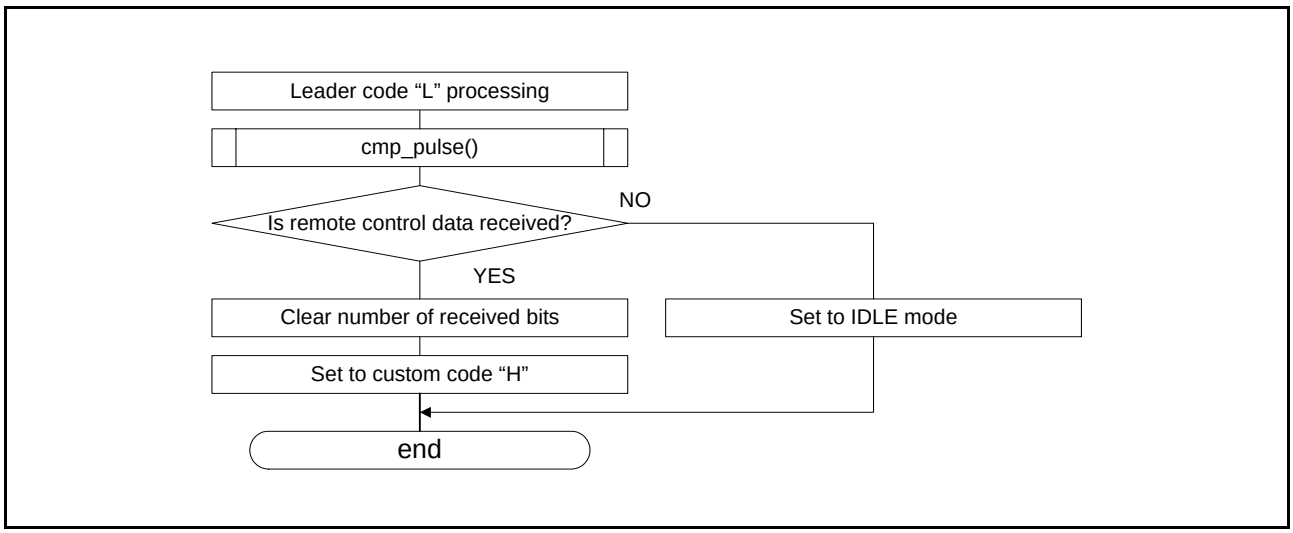



**Figure 8.9 Flowchart (check_code/leader code “L”)**

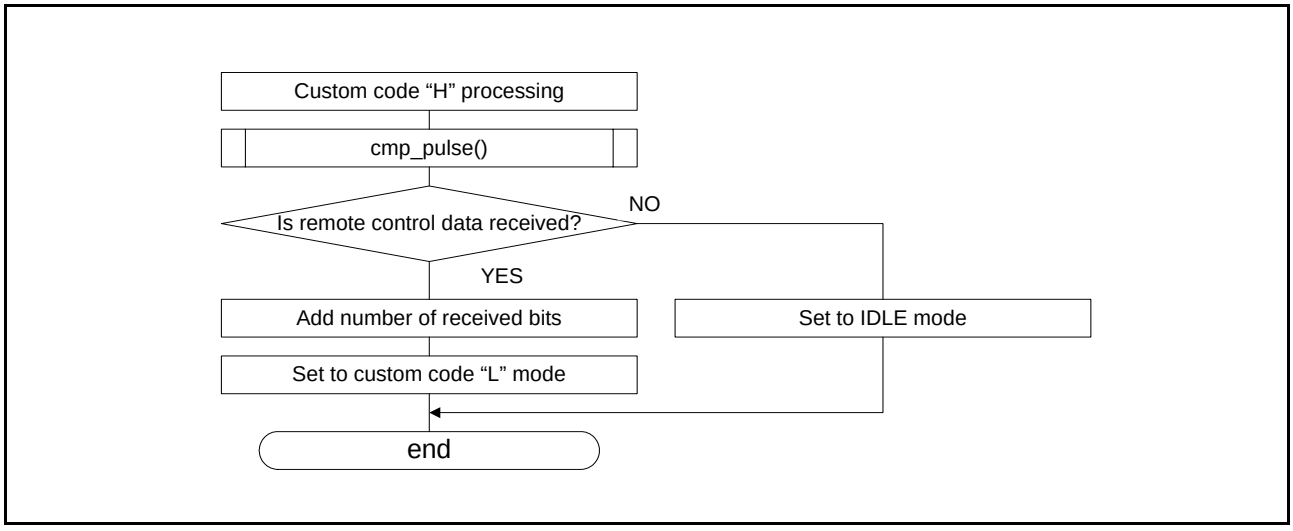



**Figure 8.10 Flowchart (check_code/custom code “H”)**

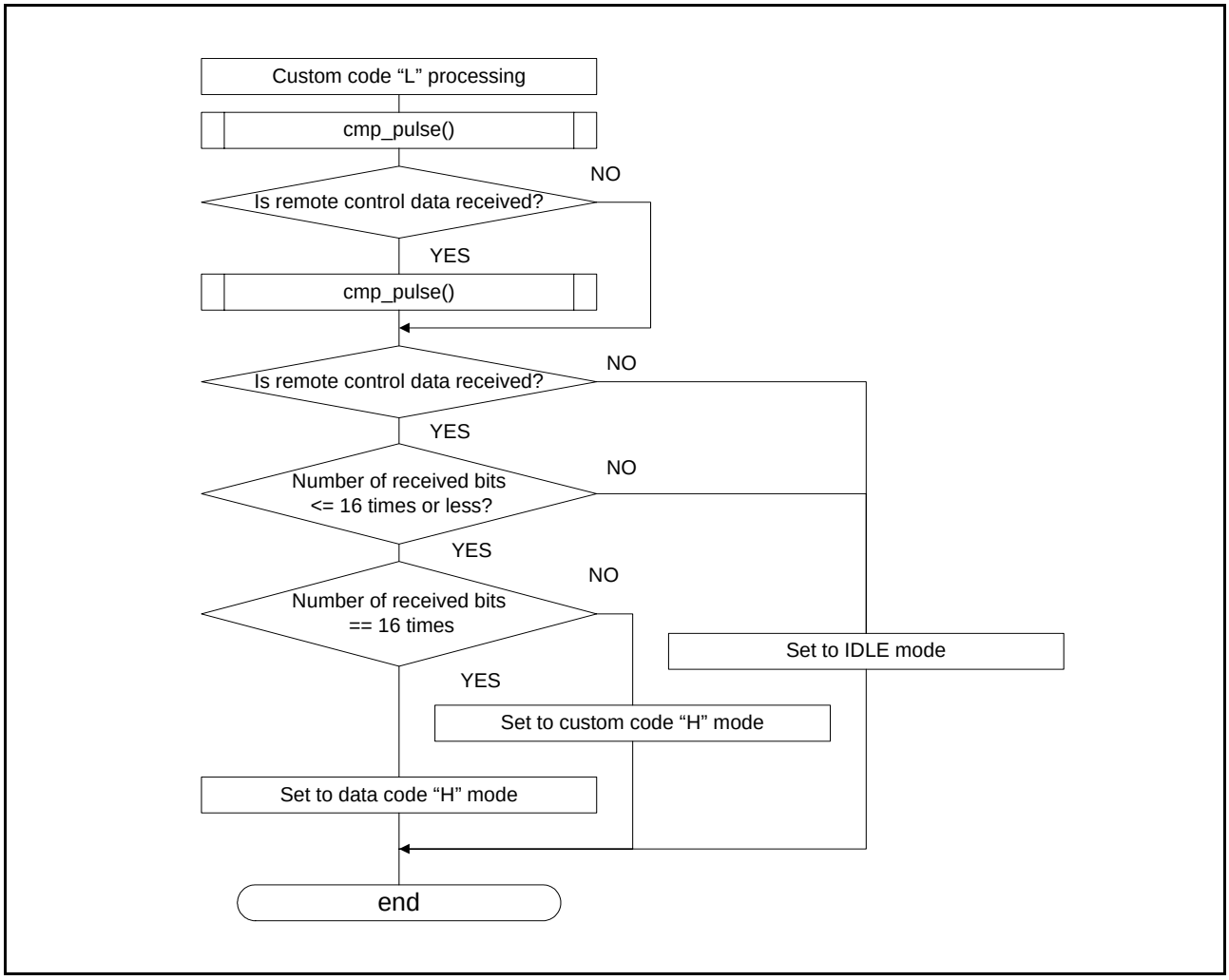



**Figure 8.11 Flowchart (check_code/custom code “L”)**

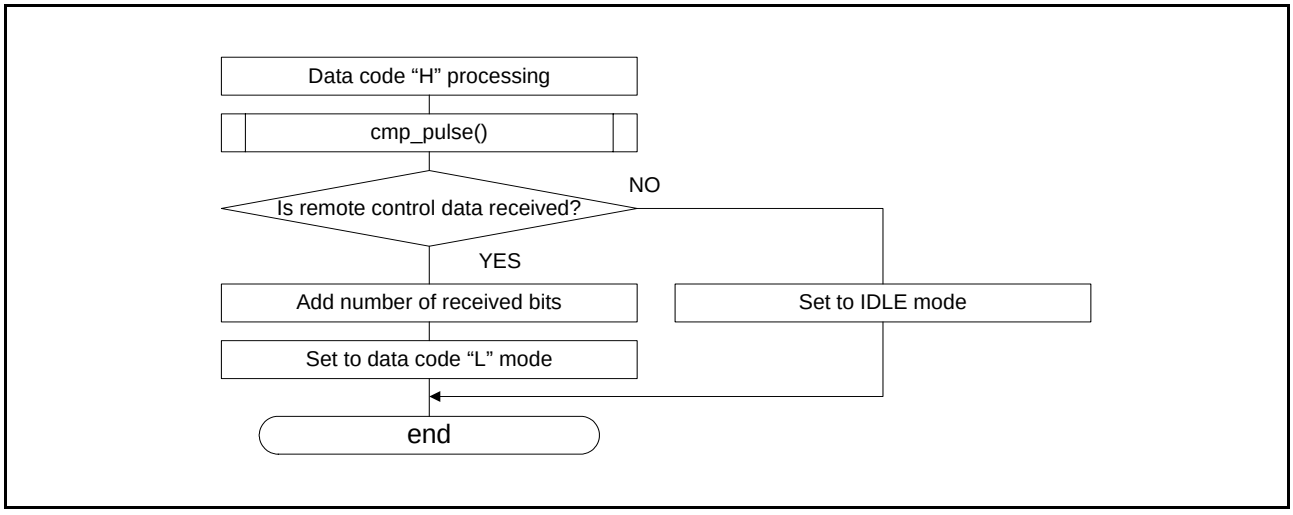



**Figure 8.12 Flowchart (check_code/data code “H”)**

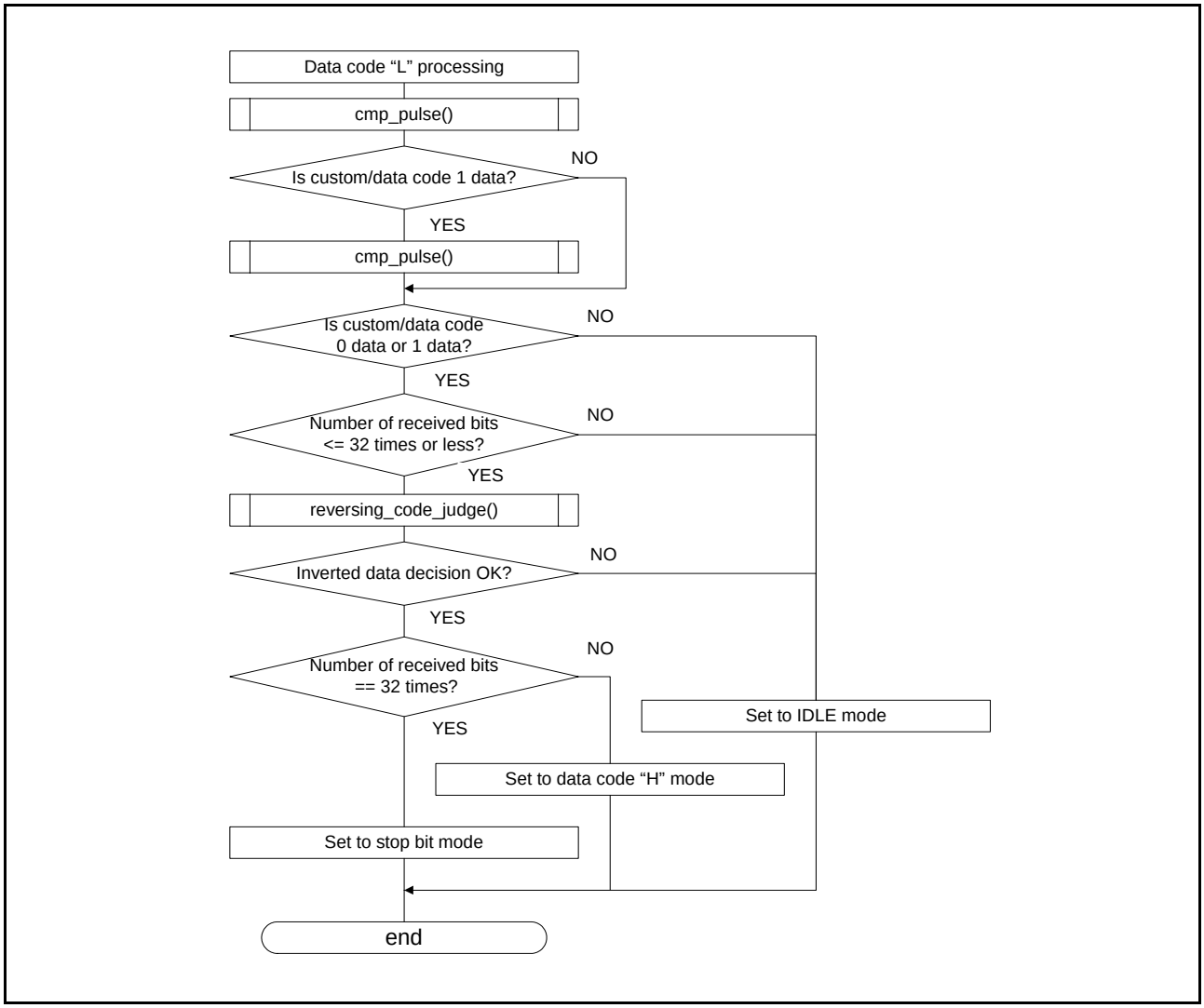



**Figure 8.13 Flowchart (check_code/data code “L”)**

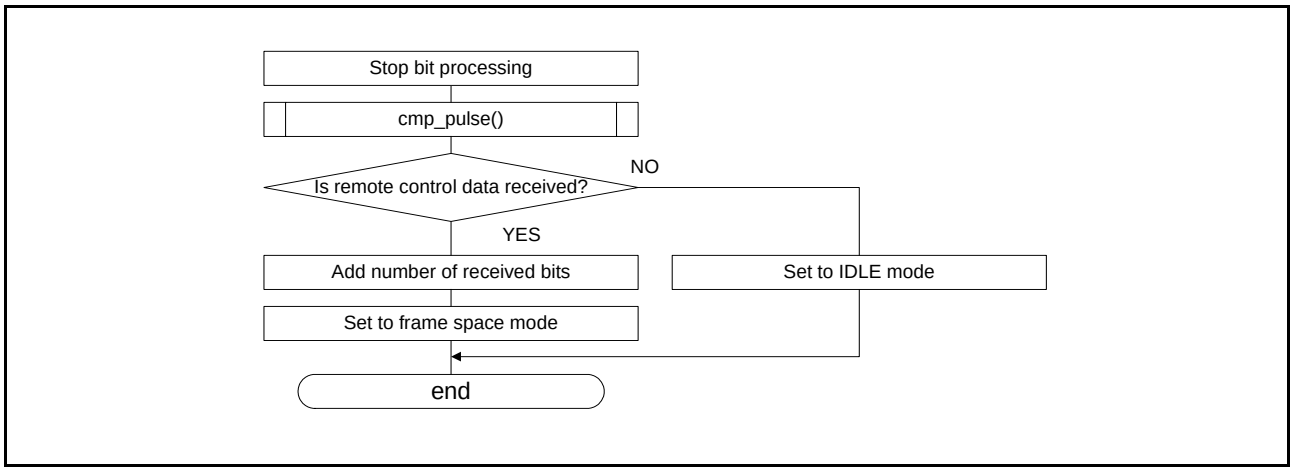



**Figure 8.14 Flowchart (check_code/stop bit)**

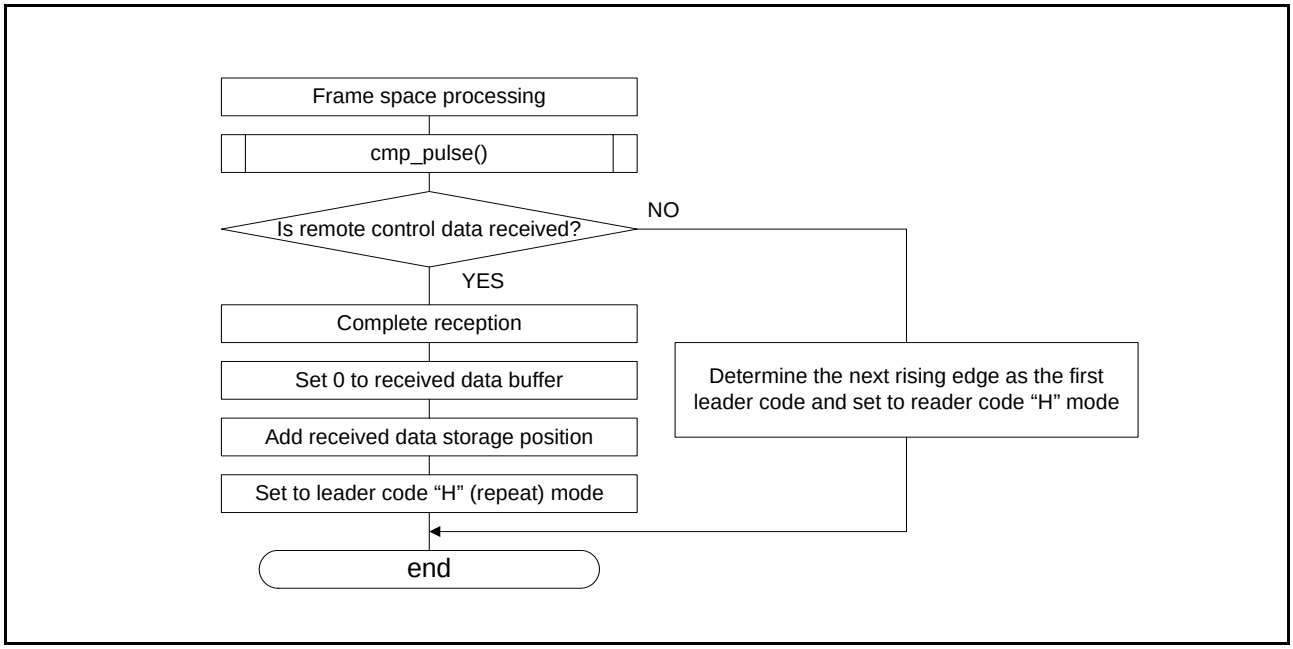



**Figure 8.15 Flowchart (check_code/frame space)**

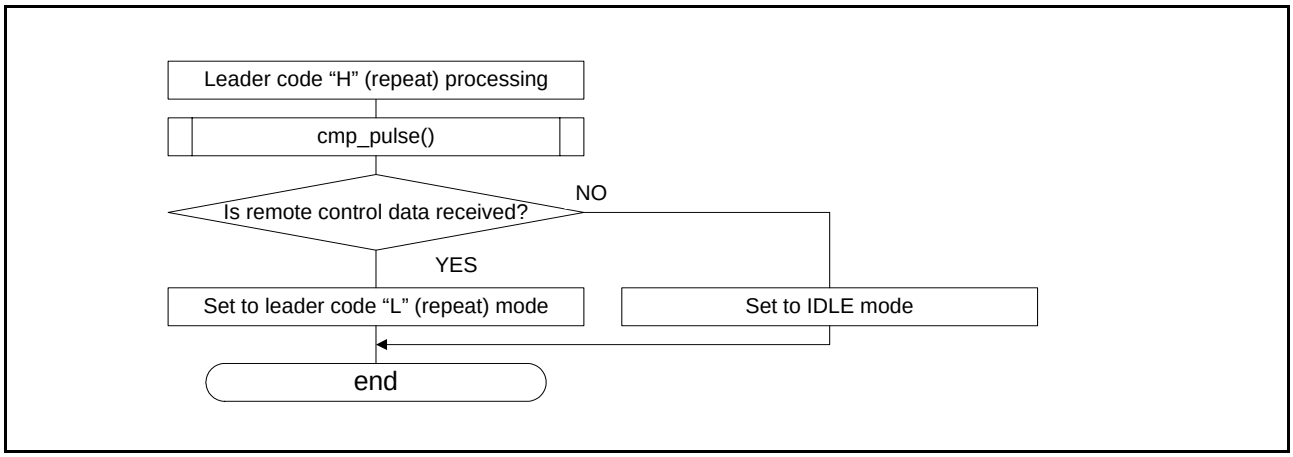



**Figure 8.16 Flowchart (check_code/leader code “H” (repeat))**

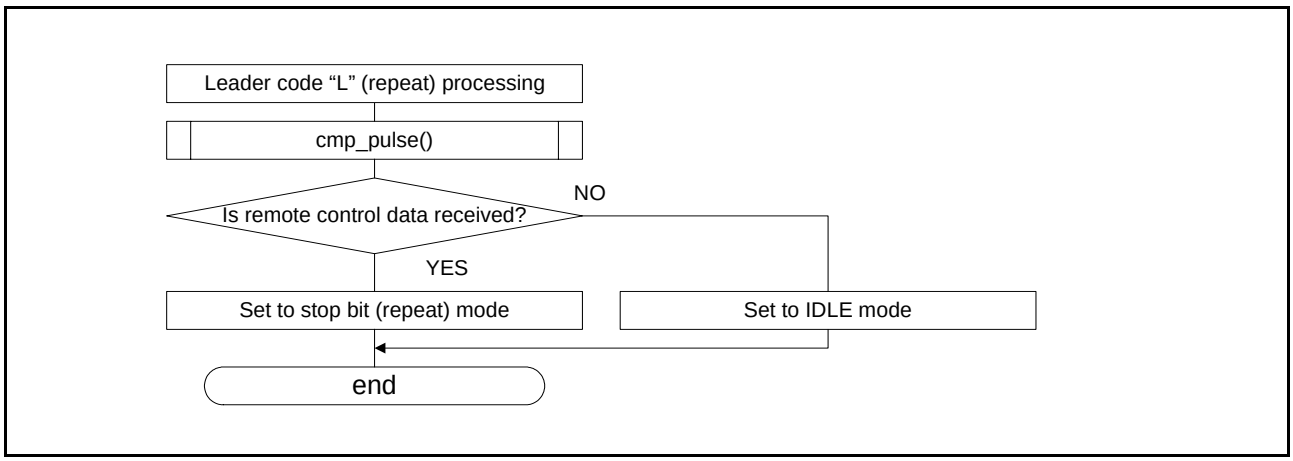



**Figure 8.17 Flowchart (check_code/leader code “L” (repeat))**

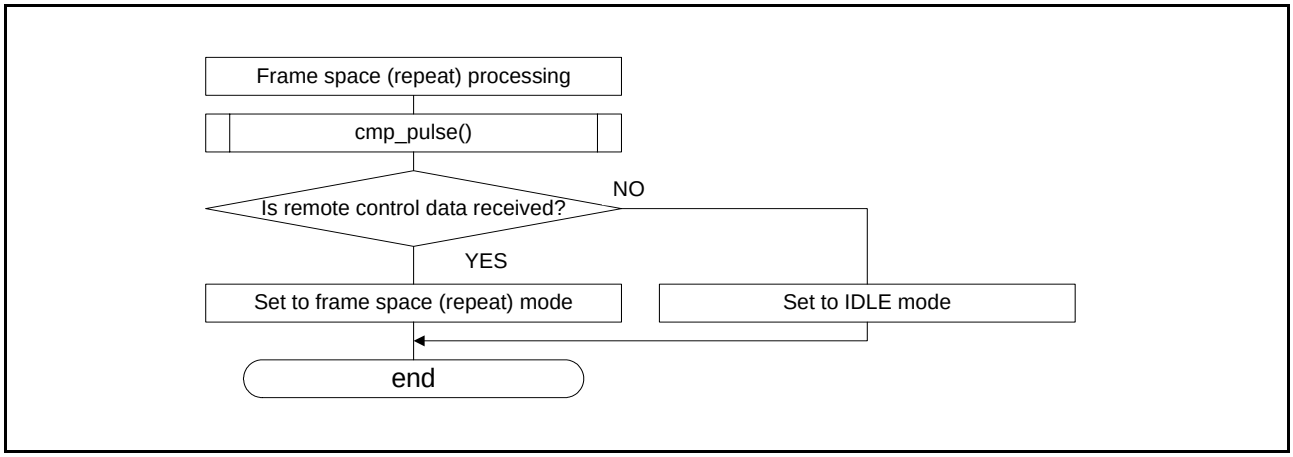



**Figure 8.18 Flowchart (check_code/stop bit (repeat))**

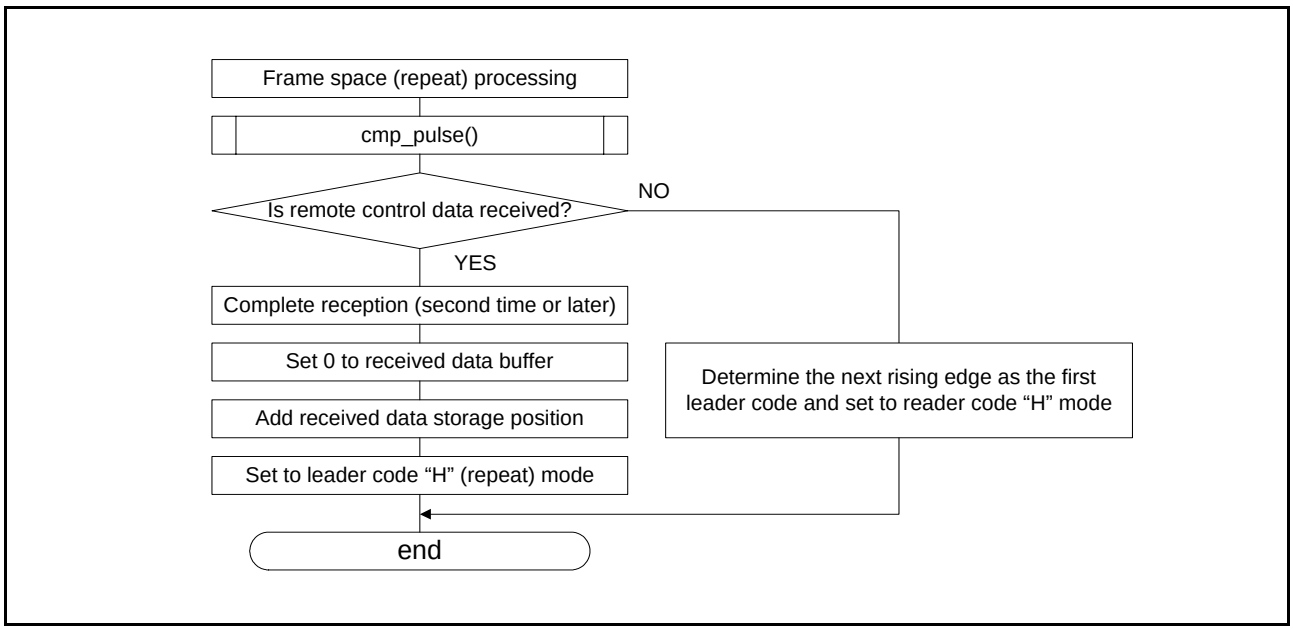



**Figure 8.19 Flowchart (check_code/frame space (repeat))**

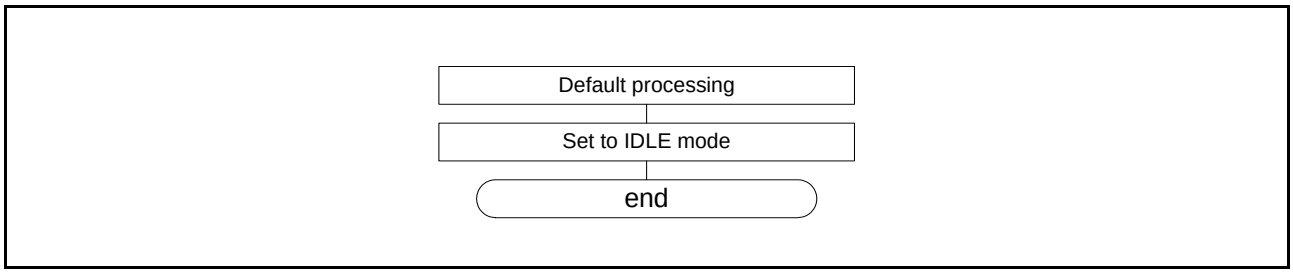



**Figure 8.20 Flowchart (check_code/default)**

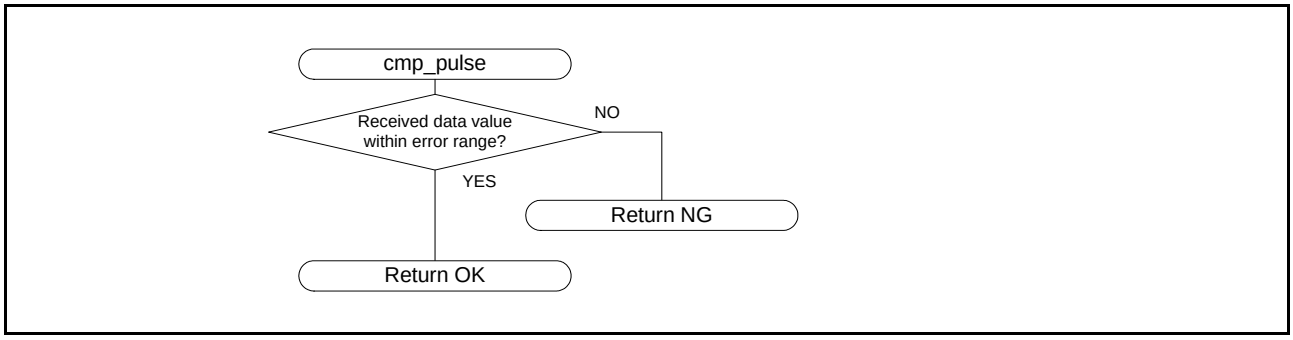



**Figure 8.21 Flowchart (cmp_pulse)**

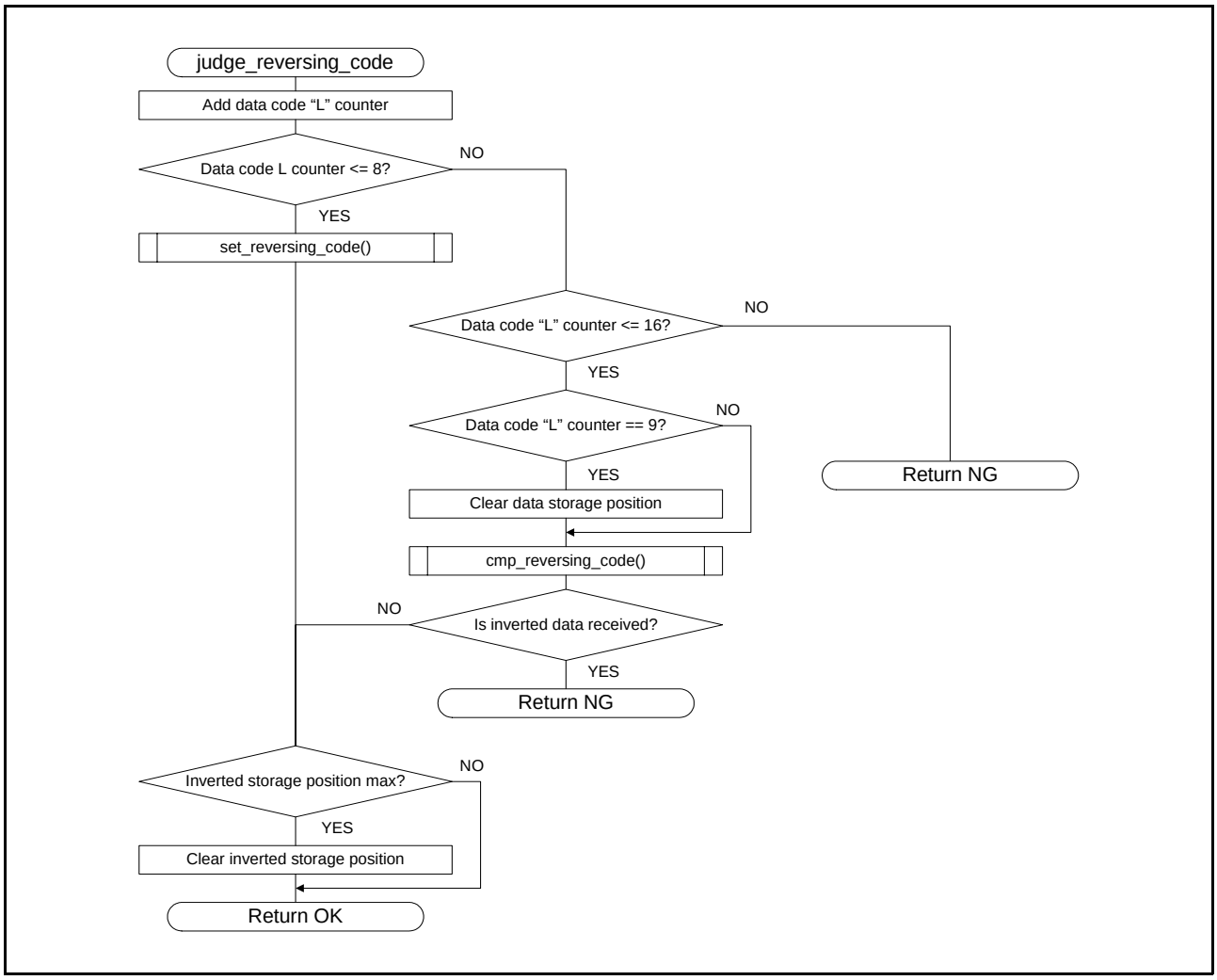



**Figure 8.22 Flowchart (judge_reversing_code)**

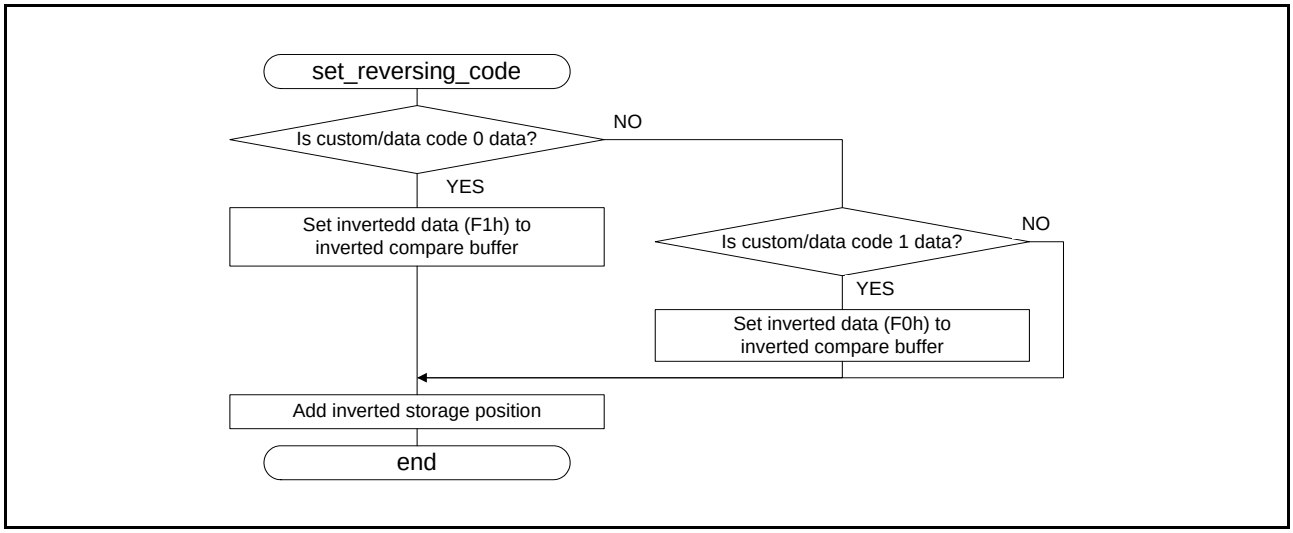



**Figure 8.23 Flowchart (set_reversing_code)**

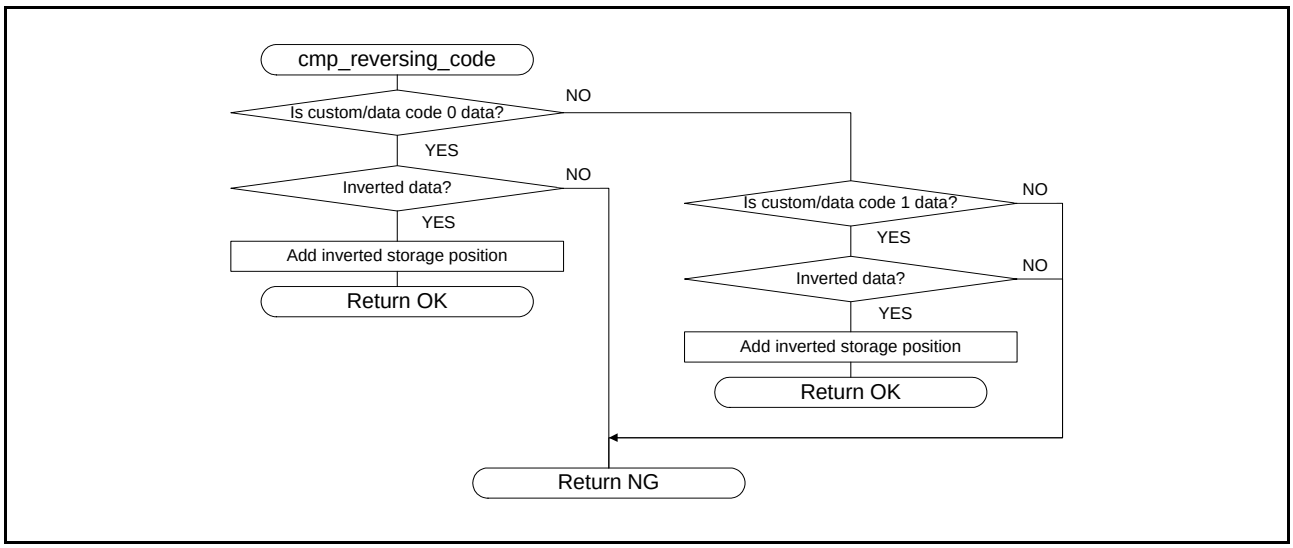



**Figure 8.24 Flowchart (cmp_reversing_code)**

## 9. Sample Programming Example

In this sample program, the CPU clock is set to 32 MHz - 8 MHz multiplied by 4.

```
/***********************************************************************/
/*                                                                     */
/*      M32C/85 Program Collection                                     */
/*                                                                     */
/*      FILE NAME   :   rjj05b0859_src.c                               */
/*      CPU         :   M32C/85 Group                                  */
/*      FUNCTION    :   Remote control reception                       */
/*      HISTORY     :   2005.07.30   Ver 1.00                          */
/*                                                                     */
/*      Copyright(C)2005, Renesas Technology Corp.                     */
/*      Copyright(C)2005, Renesas Solutions Corp.                      */
/*      All rights reserved.                                           */
/*                                                                     */
/***********************************************************************/

/***********************************************************************/
/*      Include                                                        */
/***********************************************************************/
#include    "sfr32c85.h"

/***********************************************************************/
/*      Prototype declaration                                          */
/***********************************************************************/
void main(void);
void rcv_data(void);
void time_over(void);
void set_pulse_value(void);
void check_code(void);
unsigned char cmp_pulse(unsigned short, unsigned short, unsigned short);
unsigned char judge_reversing_code(unsigned char, unsigned char);
void set_reversing_code(unsigned char, unsigned char);
unsigned char cmp_reversing_code(unsigned char, unsigned char);

/***********************************************************************/
/*      DEFINE                                                         */
/***********************************************************************/
#define OK                  0       /* OK */
#define NG                  1       /* NG */

#define IDLE                0       /* Receiving mode : IDLE                */
#define LEADER_CODE_H       1       /* Receiving mode : Leader code "H"     */
#define LEADER_CODE_L       2       /* Receiving mode : Leader code "L"     */
```

```
#define   CUSTM_CODE_H          3       /* Receiving mode : Custm code "H"              */
#define   CUSTM_CODE_L          4       /* Receiving mode : Custm code "L"              */
#define   DATA_CODE_H           5       /* Receiving mode : Data code "H"               */
#define   DATA_CODE_L           6       /* Receiving mode : Data code "L"               */
#define   STOP_BIT              7       /* Receiving mode : Stop bit                    */
#define   FRAMESPACE            8       /* Receiving mode : Frame space                 */
#define   RE_LEADER_CODE_H      9       /* Receiving mode : Leader code "H"  (repeat)   */
#define   RE_LEADER_CODE_L      10      /* Receiving mode : Leader code "L"  (repeat)   */
#define   RE_STOP_BIT           11      /* Receiving mode : Stop bit         (repeat)   */
#define   RE_FRAMESPACE         12      /* Receiving mode : Frame space      (repeat)   */

#define   LEADER_CODE_H_POS     0   /* Range data table position : Leader code "H"              */
#define   LEADER_CODE_L_POS     1   /* Range data table position : Leader code "L"              */
#define   CUSTM_H_POS           2   /* Range data table position : Custm code "H"               */
#define   CUSTM_0_L_POS         3   /* Range data table position : Custm code "L"( 0 DATA )     */
#define   CUSTM_1_L_POS         4   /* Range data table position : Custm code "L"( 1 DATA )     */
#define   DATA_H_POS            5   /* Range data table position : Data code "H"                */
#define   DATA_0_L_POS          6   /* Range data table position : Data code "L"( 0 DATA )      */
#define   DATA_1_L_POS          7   /* Range data table position : Data code "L"( 1 DATA )      */
#define   STOP_BIT_POS          8   /* Range data table position : Stop bit                     */
#define   FRAMESPACE_POS        9   /* Range data table position : Frame space                  */
#define   RE_LEADER_CODE_H_POS  10  /* Range data table position : Leader code "H" (repeat)     */
#define   RE_LEADER_CODE_L_POS  11  /* Range data table position : Leader code "L" (repeat)     */
#define   RE_STOP_BIT_POS       12  /* Range data table position : Stop bit        (repeat)     */
#define   RE_FRAMESPACE_POS     13  /* Range data table position : Frame space     (repeat)     */

#define   PULSE_MAX             100     /* Base timer storage maximum position          */
#define   REV_PULSE_MAX         10      /* Reversing buffer storage maximum position    */

#define   CUSTM_MAX_BIT_CNT     16      /* Number of custom code reception bits MAX     */
#define   DATA_MAX_8_BIT_CNT    8       /* Number of data code reception bits MAX       */
#define   DATA_MAX_LOW_BIT_CNT  16      /* Number of data code "L" maximum receptions   */
#define   RCV_COMP_BIT_CNT      32      /* Number of reception completion bits          */

#define   PLL_WAIT_1MS          5       /* PLL wait time 5ms                            */
#define   PLL_WAIT_CNT          100     /* PLL wait time 1ms                            */

/******************************************************************************/
/*          RAM                                                               */
/******************************************************************************/
unsigned char   rcv_mode;                       /* Receiving mode                       */
unsigned char   pulse_cnt;                      /* Pulse storage position               */
unsigned short pulse[PULSE_MAX];                /* Pulse storage buffer                 */
unsigned char   rcv_bit_cnt;                    /* Number of reception bits             */
unsigned char   rcv_data_cnt;                   /* Number of receive data completion    */
unsigned char   rev_pulse[REV_PULSE_MAX];       /* Buffer for reversing data confirmation */
unsigned char   rev_cnt;                        /* Buffer position for reversing data confirmation */
```

```
unsigned char    code_low_cnt;                  /* Reception code "L" counter            */

/************************************************************************************/
/*        ROM                                                                       */
/************************************************************************************/
const unsigned short cmp_tbl[14][2] =
                                                /* Receive data comparison table */
{

    /* When assuming that 62 dividing 32MHz (one cycle = 1.9375µs)                  */
    /* [*][0]: Format value of each section                                         */
    /* [*][1]: Error range value to format value(±30%)                              */
    /* [*][0]-[*][1] to [*][0]+[*}[1] : Code recognition range value                */

    {4645, 1393},    /* Range value of leader code "H"          ( 6.3 ms to 11.69 ms )    */
    {2322, 696},     /* Range value of leader code "L"          ( 3.15 ms to 5.84 ms )    */
    {289, 86},       /* Range value of custm code "H"           ( 0.393 ms to 0.726 ms )  */
    {289, 86},       /* Range value of custm code "L"( 0 DATA ) ( 0.393 ms to 0.726 ms )  */
    {872, 261},      /* Range value of custm code "L"( 1 DATA ) ( 1.183 ms to 2.195 ms )  */
    {289, 86},       /* Range value of data code "H"            ( 0.393 ms to 0.726 ms )  */
    {289, 86},       /* Range value of data code "L"( 0 DATA )  ( 0.393 ms to 0.726 ms )  */
    {872,261},       /* Range value of data code "L"( 1 DATA )  ( 1.183 ms to 2.195 ms )  */
    {289, 86},       /* Range value of stop bit                 ( 0.393 ms to 0.726 ms )  */
    {20903, 6270},   /* Range value of Frame space              ( 28.35 ms to 52.64 ms )  */
    {4645, 1393},    /* Range value of leader code "H"(repeat)  ( 6.3 ms to 11.69 ms )    */
    {1161, 348},     /* Range value of leader code "L"(repeat)  ( 1.575 ms to 2.923 ms )  */
    {289, 86},       /* Range value of stop bit         (repeat)( 0.393 ms to 0.726 ms )  */
    {49646, 14893}   /* Range value of Frame space      (repeat)( 67.33 ms to 125.04 ms ) */

};

/************************************************************************************/
/*        Pragma                                                                    */
/************************************************************************************/
/*""FUNC COMMENT""*****************************************************************
* ID                : 1.0
*-----------------------------------------------------------------------------------
* Include           : "sfr32c85.h"
*-----------------------------------------------------------------------------------
* Declaration       : void main(void)
*-----------------------------------------------------------------------------------
* Function          : main
*-----------------------------------------------------------------------------------
* Arguments         : Nothing
*-----------------------------------------------------------------------------------
* Returns           : Nothing
*-----------------------------------------------------------------------------------
```

```
* Input             : Nothing
* Output            : unsigned char rcv_mode        : Receiving mode
*                   : unsigned char pulse_cnt       : Pulse storage position
*                   : unsigned char rcv_data_cnt    : Number of receive data completion
*                   : unsigned char rcv_bit_cnt     : Number of reception bits
*                   : unsigned char rev_cnt         : Buffer position for
*                   :                                 reversing data confirmation
*                   : unsigned char code_low_cnt    : Reception code "L" counter
*-------------------------------------------------------------------------------
* Call functions    : rcv_data()
*                   : time_over()
*-------------------------------------------------------------------------------
* Note              :
*-------------------------------------------------------------------------------
* History           : 2005.07.30    Ver 1.00
*""FUNC COMMENT END""*****************************************************************/
void main(void)
{

    unsigned char i;      /* PLL wait count */
    unsigned char j;      /* PLL wait count */


    /* PLL clock is made CPU source. 8MHz -> 32MHz */
    prcr = 0x01;          /* Protect Register */
    /* 00000001B */
    /*          +---------- Protect Bit 0                                    */
    /*                      Enables writing to CM0, CM1, CM2,                */
    /*                                          MCD, PLC0, PLC1 registers    */
    /*                      0: Write disable                                 */
    /*                      1: Write enable                                  */

    plc = 0x0254;
    plc = 0x02D4;
    /* PLL Control Register 0 */
    /* 01010100B */
    /* |||| +++---------- Programmable counter select bit      */
    /* ||||                    1 0 0: Multiply-by-8            */
    /* |||+-------------- Set to "1"                           */
    /* ||+--------------- Set to "0"                           */
    /* |+---------------- Set to "1"                           */
    /* +----------------- Operation enable bit                 */
    /*                      0: PLL is Off                      */
    /*                      1: PLL is On                       */
    /* PLL Control Register 1 */
    /* 00000010B */
    /* ||| |||+---------- Set to "0"                           */
```

```
/* ||| ||+---------- Set to "1"                    */
/* ||| |+---------- PLL clock division switch bit  */
/* ||| |                    0: Divide-by-2        */
/* ||| |                    1: Divide-by-3        */
/* ||| +---------- Set to "0"                     */
/* +++---------- Set to "0"                       */


/* 5 ms wait */
for(i=0;i<PLL_WAIT_1MS;i++){
      for(j=0;j<PLL_WAIT_CNT;j++){
      }
}

cm1 = 0xA0;             /* System Clock Control Register 1 */
/* 10100000B */
/* |||||||+---------- All clock stop control bit      */
/* |||||||                  0: Clock oscillates     */
/* |||++++---------- Set to "0"                     */
/* ||+---------- Set to "1"                         */
/* |+---------- Set to "0"                          */
/* +---------- CPU clock select bit 1               */
/*                          1: PLL clock            */

mcd = 0x12;             /* Main Clock Division Register */
/* 00010010B */
/*      +++++---------- Main clock division select bit                    */
/*                          1 0 0 1 0: Divide-by-1(no division) mode */

prcr = 0x00;            /* Protect Register */
/* 00000000B */
/*           +---------- Protect Bit 0                                          */
/*                          Enables writing to CM0, CM1, CM2,                  */
/*                                              MCD, PLC0, PLC1 registers      */
/*                          0: Write disable                                   */
/*                          1: Write enable                                    */

g1bcr0 = 0x78;          /* Base Timer Control Register 10 */
/* 01111000B */
/* ||||||++---------- Count source select bit                                  */
/* ||||||                  00: Clock stops                                    */
/* |+++++---------- Count source divide ratio select bit                      */
/* |                       11110: Divide-by-62                                */
/* +---------- Base timer interrupt select bit                                */
/*                          0: Bit 15 overflows                                */

g1bcr0 = 0x7B;          /* Base Timer Control Register 10 */
```

```
/* 01111011B */
/* ||||||++---------- Count source select bit                    */
/* ||||||                    11: f1                              */
/* |+++++--------- Count source divide ratio select bit          */
/* |                         11110: Divide-by-62                 */
/* +----------------- Base timer interrupt select bit            */
/*                           0: Bit 15 overflows                 */

g1bcr1 = 0x00;          /* Base Timer Control Register 11*/
/* 00000000B */
/*   ||| |+---------- Base timer reset cause select bit 1                  */
/*   ||| |                   0: The base timer is not reset by             */
/*   ||| |                                 matching with the G1PO0 register */
/*   ||| +------------ Base timer reset cause select bit 2                 */
/*   |||                   0: The base timer is not reset by               */
/*   |||                           applying "L" to the INT0 or INT1 pin    */
/*   ||+-------------- Base timer start bit                                */
/*   ||                    0: Base timer is reset                          */
/*   ++---------------- Counter increment / Decrement control bit          */
/*                         00: Counter increment mode                      */

g1tmcr0 = 0x03;         /* Time Measurement Control Register 10 */
/* 00000011B */
/* ||||||++---------- Time measurement trigger select bit               */
/* ||||||                    11: Both edges                             */
/* ||||++------------ Digital filter function select bit                */
/* ||||                      00: No digital filter                      */
/* |||+--------------- Gate function select bit                         */
/* |||                       0: Gate function is not used              */
/* ||+---------------- Gate function clear select bit                   */
/* ||                        0: Not cleared                            */
/* |+----------------- Gate function clear bit                          */
/* |                         0: The gate is cleared by                 */
/* |                                 setting the GSC bit to "1"        */
/* |                           Set all bits 7 to 4 in the G1TMCR0 to   */
/* |                                   G1TMCR5 registers to "0".       */
/* +------------------ Prescaler function select bit                    */
/*                           0: Not used                               */
/*                             Set all bits 7 to 4 in the G1TMCR0 to   */
/*                                     G1TMCR5 registers to "0".       */

g1fs = 0x01;            /* Function Select Register 1 */
/* 00000001B */
/*           +---------- Channel 0 Time Measurement /                  */
/*                                 Waveform Generating Function Select Bit */
/*                         1: Selects the time measurement function    */
```

```
g1fe = 0x01;            /* Function Enable Register 1 */
/* 00000001B */
/*           +---------- Channel 0 Function Enable Bit                */
/*                                 1 : Enables functions for channel 0 */

iio3ir = 0x00;          /* Interrupt Request Register */
/* 00000000B */

iio3ie = 0x01;          /* Interrupt Enable Register */
/* 00000001B */
/*           +---------- Interrupt request select bit */
/*                                 1 : Interrupt request is used for interrupt */

iio3ie = 0x05;          /* Interrupt Enable Register */
/* 00000101B */
/*         | +---------- Interrupt request select bit                          */
/*         |             1 : Interrupt request is used for interrupt           */
/*         +------------ Intelligent I/O Time Measurement 3 Interrupt Enabled   */
/*                       1 : Enables an interrupt by bit 2 in IIO0IR register  */

ps1 = 0x00;             /* Function Select Register A1 */
/* 00000000B */
/* |||||||+---------- Port P70 output function select bit       */
/* |||||||           0: I/O port                                */
/* ||||||+---------- Port P71 output function select bit       */
/* ||||||            0: I/O port                                */
/* |||||+---------- Port P72 output function select bit        */
/* |||||             0: I/O port                                */
/* ||||+---------- Port P73 output function select bit         */
/* ||||              0: I/O port                                */
/* |||+---------- Port P74 output function select bit          */
/* |||               0: I/O port                                */
/* ||+---------- Port P75 output function select bit           */
/* ||                0: I/O port                                */
/* |+---------- Port P76 output function select bit            */
/* |                 0: I/O port                                */
/* +---------- Port P77 output function select bit            */
/*                   0: I/O port                                */

pd7 = 0x00;             /* Port P7 Direction Register */
/* 00000000B */
/* |||||||+---------- Port P70 direction bit                          */
/* |||||||           0 : Input mode (Functions as input port)       */
/* ||||||+---------- Port P71 direction bit                          */
/* ||||||            0 : Input mode (Functions as input port)       */
/* |||||+---------- Port P72 direction bit                           */
/* |||||             0 : Input mode (Functions as input port)       */
```

```
/* ||||+------------ Port P73 direction bit     */
/* ||||                          0: Input mode (Functions as input port)    */
/* |||+------------- Port P74 direction bit     */
/* |||                           0: Input mode (Functions as input port)    */
/* ||+-------------- Port P75 direction bit     */
/* ||                            0: Input mode (Functions as input port)    */
/* |+--------------- Port P76 direction bit     */
/* |                             0: Input mode (Functions as input port)    */
/* +---------------- Port P77 direction bit     */
/*                               0: Input mode (Functions as input port)    */

tabsr = 0x00;           /* Count Start Flag */
/* 00000000B */
/*          |+---------- Timer A0 count start flag         */
/*          |                0: Stops counting         */
/*          +----------- Timer A1 count start flag         */
/*                           0: Stops counting         */

ta0mr = 0x00;               /* Timer A0 Mode Register */
/* 00000000B */
/* ||||||++---------- Operating mode select bit                          */
/* ||||||                    0 0: Timer mode                          */
/* |||||+------------ Set to "0"       */
/* |||++------------- Gate function select bit                          */
/* |||                       0 X: Gate function disabled          */
/* |||                  (TAiIN pin is a programmable I/O pin)       */
/* ||+--------------- Set to "0" in timer mode                          */
/* ++---------------                0 0: f1                          */

ta0 = 0x7CFF;               /* Timer A0 Register (7D00-1) */

ta1mr = 0x01;               /* Timer A1 Mode Register */
/* 00000001B */
/* ||||||++---------- Operating mode select bit                            */
/* ||||||                    0 1: Event counter mode                  */
/* |||||+------------ Set to "0"                                        */
/* ||||+------------- Count polarity select bit                         */
/* ||||                      0: Counts falling edges of               */
/* ||||                                 an external signal            */
/* |||+-------------- Increment/Decrement switching source select bit  */
/* |||                       0: UDF registser setting                 */
/* ||+--------------- Set to "0" in event counter mode                  */
/* |+---------------- Count operation type select bit                   */
/* |                         0: Reloading                             */
/* +----------------- Set to "0"                                       */

ta1 = 0x008B;               /* Timer A1 Register (8C-1)*/
```

```
trgsr = 0x02;                /* Trigger Select Register */
/* 00000010B */
/* ||||||++---------- Timer A1 event/trigger select bit                */
/* ||||||                    1 0: Selects the TA0 overflows             */
/* ||||++------------ Timer A2 event/trigger select bit                */
/* ||||                      0 0: Selects an input to the TA2IN pin     */
/* ||++--------------- Timer A3 event/trigger select bit                */
/* ||                        0 0: Selects an input to the TA3IN pin     */
/* ++----------------- Timer A4 event/trigger select bit                */
/*                           0 0: Selects an input to the TA4IN pin     */

ta0ic = 0x00;                /* Interrupt Control Register */
/* 00000000B */
/*       |+++---------- Interrupt priority level select bit             */
/*       |                   0 0 0: Level 0 (interrupt disabled)        */
/*       +-------------- Interrupt request bit                          */
/*                           0: No interrupt requested                  */

ta1ic = 0x00;                /* Interrupt Control Register */
/* 00000000B */
/*       |+++---------- Interrupt priority level select bit             */
/*       |                   0 0 0: Level 0 (interrupt disabled)        */
/*       +-------------- Interrupt request bit                          */
/*                           0: No interrupt requested                  */

/* RAM Initial */
rcv_mode = IDLE;
pulse_cnt = 0;
rcv_data_cnt = 0;
rcv_bit_cnt = 0;
rev_cnt = 0;
code_low_cnt = 0;

asm("FSET I");          /* "I" flag */

bts_g1bcr1 = 1;         /* Base timer start of counting */

while(1){
                        /* Data reception waiting */
    if(tm10r == 1){
        /* When there is an external data reception */
        iio3ic = 0x00;          /* Interrupt request bit clear          */
        iio3ir = 0x00;          /* Interrupt Request Register clear     */
        rcv_data();             /* rcv_data function                    */

    }
```

```
            if(ir_ta1ic == 1){
                /* When time passes by one frame      */
                tabsr = 0x00;           /* timer A0, A1 stop                */
                ir_ta1ic = 0;           /* Interrupt request flag clear     */
                time_over();            /* time_over function               */

            }
        }
}


/*""FUNC COMMENT""*****************************************************************
* ID                    : 1.1
*-----------------------------------------------------------------------------
* Include               : "sfr32c85.h"
*-----------------------------------------------------------------------------
* Declaration           : void rcv_data(void)
*-----------------------------------------------------------------------------
* Function              : Receive data setting
*                       : when receive data interrupt is detected
*-----------------------------------------------------------------------------
* Arguments             : Nothing
*-----------------------------------------------------------------------------
* Returns               : Nothing
*-----------------------------------------------------------------------------
* Input                 : unsigned char rcv_mode      : Receiving mode
* Output                : unsigned char pulse_cnt     : Pulse storage position
*                       : unsigned char rcv_bit_cnt   : Number of reception bits
*                       : unsigned char rev_cnt       : Buffer position for
*                       :                               reversing data confirmation
*                       : unsigned char code_low_cnt : Reception code "L" counter
*-----------------------------------------------------------------------------
* Call functions        : set_pulse_value()
*                       : check_code()
*-----------------------------------------------------------------------------
* Note                  :
*-----------------------------------------------------------------------------
* History               : 2005.07.30    Ver 1.00
*""FUNC COMMENT END""*************************************************************/
void rcv_data(void)
{

    if( (rcv_mode == IDLE) || (rcv_mode == FRAMESPACE) || (rcv_mode == RE_FRAMESPACE)){
        /* When Leader code "H" detects */
        tabsr = 0x00;           /* timer A0, A1 stop            */
        ta0 = 0x7CFF;           /* 1 ms     (7D00-1)h           */
        ta1 = 0x008B;           /* 140 ms (Underflow timing)    */
```

```
            tabsr = 0x03;           /* timer A0, A1 start           */
        }

        if(rcv_mode == LEADER_CODE_H){
            /* When Leader code "H" (Data of the first time)            */
            rcv_bit_cnt = 0;        /* Number of reception bits clear                       */
            rev_cnt = 0;            /* Buffer position for reversing data confirmation clear */
            code_low_cnt = 0;       /* Reception code "L" counter clear                     */
            rcv_data_cnt = 0;       /* Number of receive data completion clear              */
        }

        set_pulse_value();          /* set_pulse_value function     */
        check_code();               /* check_code function          */

        pulse_cnt++;                /* Pulse storage position add */

}


/*""FUNC COMMENT""*****************************************************************************
* ID                    : 1.2
*-------------------------------------------------------------------------------
* Include               : Nothing
*-------------------------------------------------------------------------------
* Declaration           : void time_over(void)
*-------------------------------------------------------------------------------
* Function              : Processing after it passes of one frame
*                       :
*-------------------------------------------------------------------------------
* Arguments             : Nothing
*-------------------------------------------------------------------------------
* Returns               : Nothing
*-------------------------------------------------------------------------------
* Input                 : unsigned char rcv_mode      : Receiving mode
* Output                : unsigned char rcv_data_cnt  : Number of receive data completion
*                       : unsigned short pulse[]      : Pulse storage buffer
*                       : unsigned char pulse_cnt     : Pulse storage position
*                       : unsigned char rcv_mode      : Receiving mode
*-------------------------------------------------------------------------------
* Call functions        : Nothing
*-------------------------------------------------------------------------------
* Note                  :
*-------------------------------------------------------------------------------
* History               : 2005.07.30    Ver 1.00
*""FUNC COMMENT END""*************************************************************/
void time_over(void)
{
```

```
    if( (rcv_mode == FRAMESPACE) || (rcv_mode == RE_FRAMESPACE) ){
        /* When the frame space passes the fixed time */
        rcv_data_cnt++;             /* Number of receive data completion add */

    }

    pulse[pulse_cnt] = 0;           /* Stopper setting */
    pulse_cnt++;                    /* Pulse storage position add   */

    rcv_mode = IDLE;                /* mode clear */

}


/*""FUNC COMMENT""*****************************************************************
* ID                    : 1.3
*--------------------------------------------------------------------------------
* Include               : "sfr32c85.h"
*--------------------------------------------------------------------------------
* Declaration           : void set_pulse_value(void)
*--------------------------------------------------------------------------------
* Function              : The pulse value is set
*                       :
*--------------------------------------------------------------------------------
* Arguments             : Nothing
*--------------------------------------------------------------------------------
* Returns               : Nothing
*--------------------------------------------------------------------------------
* Input                 : unsigned char pulse_cnt   : Pulse storage position
* Output                : unsigned short pulse[]    : Pulse storage buffer
*                       : unsigned char pulse_cnt   : Pulse storage position
*--------------------------------------------------------------------------------
* Call functions        : Nothing
*--------------------------------------------------------------------------------
* Note                  :
*--------------------------------------------------------------------------------
* History               : 2005.07.30    Ver 1.00
*""FUNC COMMENT END""*************************************************************/
void set_pulse_value(void)
{

    unsigned short now_tr;                  /* Present capture value */


    static unsigned short old_tr = 0;       /* The previous capture value of one */
```

```
        now_tr = (unsigned short)g1tm0;        /* capture value storage */

        if(pulse_cnt >= PULSE_MAX){
            /* When the storage position is larger than the MAX value */
            pulse_cnt = 0;                      /* storage position clear */
        }

        pulse[pulse_cnt] = now_tr - old_tr; /* pulse value storage */

        old_tr = now_tr;                        /* The present capture value is set for   */
                                                /* the previous capture value of one      */

}


/*""FUNC COMMENT""*****************************************************************
* ID                    : 1.4
*--------------------------------------------------------------------------------
* Include               : Nothing
*--------------------------------------------------------------------------------
* Declaration           : void check_code(void)
*--------------------------------------------------------------------------------
* Function              : Data code check function according to mode
*                       :
*--------------------------------------------------------------------------------
* Arguments             : Nothing
*--------------------------------------------------------------------------------
* Returns               : Nothing
*--------------------------------------------------------------------------------
* Input                 : unsigned char rcv_mode      : Receiving mode
*                       : unsigned short pulse[]      : Pulse storage buffer
*                       : unsigned char pulse_cnt     : Pulse storage position
*                       : unsigned char rcv_bit_cnt   : Number of reception bits
* Output                : unsigned char rcv_mode      : Receiving mode
*                       : unsigned char rcv_bit_cnt   : Number of reception bits
*                       : unsigned short pulse[]      : Pulse storage buffer
*                       : unsigned char rcv_data_cnt  : Number of receive data completion
*--------------------------------------------------------------------------------
* Call functions        : cmp_pulse()
*--------------------------------------------------------------------------------
* Note                  :
*--------------------------------------------------------------------------------
* History               : 2005.07.30    Ver 1.00
*""FUNC COMMENT END""*************************************************************/
void check_code(void)
{
```

```
unsigned char rtn;          /* Return value */
unsigned char rtn_0;        /* Return value for data code */
unsigned char rtn_1;        /* Return value for data code */


switch(rcv_mode){
    case IDLE:                      /* When mode is IDLE */
        rcv_mode = LEADER_CODE_H;       /* Mode transition */
        break;

    case LEADER_CODE_H:             /* When mode is LEADER_CODE_H */
        rtn = cmp_pulse(pulse[pulse_cnt],
                        cmp_tbl[LEADER_CODE_H_POS][0] + cmp_tbl[LEADER_CODE_H_POS][1],
                         cmp_tbl[LEADER_CODE_H_POS][0] - cmp_tbl[LEADER_CODE_H_POS][1]);
        if(rtn == OK){
            /* When the range of LEADER_CODE_H */
            rcv_mode = LEADER_CODE_L;       /* Mode transition */
            break;
        }
        rcv_mode = IDLE;                    /* When the receive data error, mode clear */
        break;

    case LEADER_CODE_L:             /* When mode is LEADER_CODE_L */
        rtn = cmp_pulse(pulse[pulse_cnt],
                        cmp_tbl[LEADER_CODE_L_POS][0] + cmp_tbl[LEADER_CODE_L_POS][1],
                         cmp_tbl[LEADER_CODE_L_POS][0] - cmp_tbl[LEADER_CODE_L_POS][1]);
        if(rtn == OK){
          /* When the range of LEADER_CODE_L */
          rcv_bit_cnt = 0;                  /* Number of reception bits clear */
          rcv_mode = CUSTM_CODE_H;      /* Mode transition */
          break;
        }
        rcv_mode = IDLE;                    /* When the receive data error, mode clear */
        break;

    case CUSTM_CODE_H:          /* When mode is CUSTM_CODE_H */
        rtn = cmp_pulse(pulse[pulse_cnt],
                          cmp_tbl[CUSTM_H_POS][0] + cmp_tbl[CUSTM_H_POS][1],
                              cmp_tbl[CUSTM_H_POS][0] - cmp_tbl[CUSTM_H_POS][1]);
        if(rtn == OK){
            /* When the range of CUSTM_CODE_H */
            rcv_bit_cnt++;                  /* Number of reception bits add */
            rcv_mode = CUSTM_CODE_L;    /* Mode transition */
            break;
        }
        rcv_mode = IDLE;                    /* When the receive data error, mode clear */
```

```
        break;

    case CUSTM_CODE_L:          /* When mode is CUSTM_CODE_L */
        rtn = cmp_pulse(pulse[pulse_cnt],
                            cmp_tbl[CUSTM_0_L_POS][0] + cmp_tbl[CUSTM_0_L_POS][1],
                                cmp_tbl[CUSTM_0_L_POS][0] - cmp_tbl[CUSTM_0_L_POS][1]);
        if(rtn != OK){
            /* When data is not 0 */
            rtn = cmp_pulse(pulse[pulse_cnt],
                                cmp_tbl[CUSTM_1_L_POS][0] + cmp_tbl[CUSTM_1_L_POS][1],
                                    cmp_tbl[CUSTM_1_L_POS][0] - cmp_tbl[CUSTM_1_L_POS][1]);
        }

        if(rtn == OK){
            /* When the range of CUSTM_CODE_L( "0" data or "1" data ) */
            if(rcv_bit_cnt <= CUSTM_MAX_BIT_CNT){
                /* When The number of reception bits is within 16 times */
                if(rcv_bit_cnt == CUSTM_MAX_BIT_CNT){
                    /* When The number of reception bits is 16 times */
                    rcv_mode = DATA_CODE_H;     /* Mode transition */
                    break;
                }
                rcv_mode = CUSTM_CODE_H;        /* Mode transition */
                break;
            }
        }
        rcv_mode = IDLE;                    /* When the receive data error, mode clear */
        break;

    case DATA_CODE_H:           /* When mode is DATA_CODE_H */
        rtn = cmp_pulse(pulse[pulse_cnt],
                            cmp_tbl[DATA_H_POS][0] + cmp_tbl[DATA_H_POS][1],
                                cmp_tbl[DATA_H_POS][0] - cmp_tbl[DATA_H_POS][1]);
        if(rtn == OK){
            /* When the range of DATA_CODE_H */
            rcv_bit_cnt++;                      /* Number of reception bits add */
            rcv_mode = DATA_CODE_L;             /* Mode transition */
            break;
        }
        rcv_mode = IDLE;                        /* When the receive data error, mode clear */
        break;

    case DATA_CODE_L:       /* When mode is DATA_CODE_L */

        rtn_0 = NG;             /* Initialization */
        rtn_1 = NG;             /* Initialization */
```

```
        rtn_0 = cmp_pulse(pulse[pulse_cnt],
                          cmp_tbl[DATA_0_L_POS][0] + cmp_tbl[DATA_0_L_POS][1],
                              cmp_tbl[DATA_0_L_POS][0] - cmp_tbl[DATA_0_L_POS][1]);
        if(rtn_0 != OK){
            /* When data is not 0 */
            rtn_1 = cmp_pulse(pulse[pulse_cnt],
                              cmp_tbl[DATA_1_L_POS][0] + cmp_tbl[DATA_1_L_POS][1],
                                  cmp_tbl[DATA_1_L_POS][0] - cmp_tbl[DATA_1_L_POS][1]);
        }

        if( (rtn_0 == OK) || (rtn_1 == OK) ){
            /* When the range of DATA_CODE_L( "0" data or "1" data ) */
            if(rcv_bit_cnt <= RCV_COMP_BIT_CNT){

                /* When The number of reception bits is within 32 times */
                rtn = judge_reversing_code(rtn_0, rtn_1);     /* judge_reversing_code function */
                if(rtn == OK){
                    /* When reversing data judgment OK */
                    if(rcv_bit_cnt == RCV_COMP_BIT_CNT){
                        rcv_mode = STOP_BIT;            /* Mode transition */
                        break;
                    }
                    rcv_mode = DATA_CODE_H;             /* Mode transition */
                    break;
                }
            }
        }
        rcv_mode = IDLE;                        /* When the receive data error, mode clear */
        break;

    case STOP_BIT:              /* When mode is STOP_BIT */
        rtn = cmp_pulse(pulse[pulse_cnt],
                          cmp_tbl[STOP_BIT_POS][0] + cmp_tbl[STOP_BIT_POS][1],
                              cmp_tbl[STOP_BIT_POS][0] - cmp_tbl[STOP_BIT_POS][1]);
        if(rtn == OK){
            /* When the range of STOP_BIT */
            rcv_bit_cnt++;                      /* Number of reception bits add */
            rcv_mode = FRAMESPACE;  /* Mode transition */
            break;
        }
        rcv_mode = IDLE;                        /* When the receive data error, mode clear */
        break;

    case FRAMESPACE:                            /* When mode is FRAMESPACE */
        rtn = cmp_pulse(pulse[pulse_cnt],
                          cmp_tbl[FRAMESPACE_POS][0] + cmp_tbl[FRAMESPACE_POS][1],
                              cmp_tbl[FRAMESPACE_POS][0] - cmp_tbl[FRAMESPACE_POS][1]);
```

```
        if(rtn == OK){
            /* When the range of FRAMESPACE */
            /* Reception completion(The first data) */
            rcv_data_cnt++;                 /* Number of reception completion add */
            pulse[pulse_cnt] = 0;           /* Stopper setting */
            pulse_cnt++;                    /* Pulse storage position add */
            rcv_mode = RE_LEADER_CODE_H;    /* Mode transition */
            break;
        }
        rcv_mode = LEADER_CODE_H;           /* When the receive data error, mode clear */
        break;

    case RE_LEADER_CODE_H:      /* When mode is RE_LEADER_CODE_H */
        rtn = cmp_pulse(pulse[pulse_cnt],
                cmp_tbl[RE_LEADER_CODE_H_POS][0] + cmp_tbl[RE_LEADER_CODE_H_POS][1],
                 cmp_tbl[RE_LEADER_CODE_H_POS][0] - cmp_tbl[RE_LEADER_CODE_H_POS][1]);
        if(rtn == OK){
            /* When the range of RE_LEADER_CODE_H */
            rcv_mode = RE_LEADER_CODE_L;    /* Mode transition */
            break;
        }
        rcv_mode = IDLE;                    /* When the receive data error, mode clear */
        break;

    case RE_LEADER_CODE_L:      /* When mode is RE_LEADER_CODE_L */
        rtn = cmp_pulse(pulse[pulse_cnt],
                cmp_tbl[RE_LEADER_CODE_L_POS][0] + cmp_tbl[RE_LEADER_CODE_L_POS][1],
                 cmp_tbl[RE_LEADER_CODE_L_POS][0] - cmp_tbl[RE_LEADER_CODE_L_POS][1]);
        if(rtn == OK){
            /* When the range of RE_LEADER_CODE_L */
            rcv_mode = RE_STOP_BIT;         /* Mode transition */
            break;
        }
        rcv_mode = IDLE;                    /* When the receive data error, mode clear */
        break;

    case RE_STOP_BIT:           /* When mode is RE_STOP_BIT */
        rtn = cmp_pulse(pulse[pulse_cnt],
                        cmp_tbl[RE_STOP_BIT_POS][0] + cmp_tbl[RE_STOP_BIT_POS][1],
                          cmp_tbl[RE_STOP_BIT_POS][0] - cmp_tbl[RE_STOP_BIT_POS][1]);
        if(rtn == OK){
            /* When the range of RE_STOP_BIT */
            rcv_mode = RE_FRAMESPACE;   /* Mode transition */
            break;
        }
        rcv_mode = IDLE;                    /* When the receive data error, mode clear */
        break;
```

```
        case RE_FRAMESPACE:          /* When mode is RE_FRAMESPACE */
            rtn = cmp_pulse(pulse[pulse_cnt],
                            cmp_tbl[RE_FRAMESPACE_POS][0] + cmp_tbl[RE_FRAMESPACE_POS][1],
                             cmp_tbl[RE_FRAMESPACE_POS][0] - cmp_tbl[RE_FRAMESPACE_POS][1]);
            if(rtn == OK){
                /* When the range of RE_FRAMESPACE */
                /* Reception completion(Data since the second times)*/
                rcv_data_cnt++;                 /* Number of reception completion add */
                pulse[pulse_cnt] = 0;           /* Stopper setting */
                pulse_cnt++;                    /* Pulse storage position add */
                rcv_mode = RE_LEADER_CODE_H;  /* Mode transition */
                break;
            }
            rcv_mode = LEADER_CODE_H;           /* When the receive data error, mode clear */
            break;

        default:
            rcv_mode = IDLE;                    /* When the receive data error, mode clear */
            break;

    }

}


/*""FUNC COMMENT""*****************************************************************
* ID                    : 1.5
*---------------------------------------------------------------------------------
* Include               : Nothing
*---------------------------------------------------------------------------------
* Declaration           : unsigned char cmp_pulse(
*                       :       unsigned short,
*                       :       unsigned short,
*                       :       unsigned short      )
*---------------------------------------------------------------------------------
* Function              : Receive data range judgment
*                       :
*---------------------------------------------------------------------------------
* Arguments             : d_pulse : Pulse value
*                       : hi    : Pulse maximum range value
*                       : low   : Pulse minimum range value
*---------------------------------------------------------------------------------
* Returns               : 0: OK
*                       : 1: NG
*---------------------------------------------------------------------------------
* Input             : unsigned short d_pulse: Pulse storage buffer
```

```
* Output                : Nothing
*-------------------------------------------------------------------------------
* Call functions        : Nothing
*-------------------------------------------------------------------------------
* Note                  :
*-------------------------------------------------------------------------------
* History               : 2005.07.30    Ver 1.00
*""FUNC COMMENT END""*************************************************************/
unsigned char cmp_pulse(unsigned short d_pulse,unsigned short hi,unsigned short low)
{

    if( (d_pulse > low) && (d_pulse < hi) ) {
        /* When the pulse value is an error range or less */
        return OK;
    }
    return NG;

}


/*""FUNC COMMENT""*************************************************************************
* ID                    : 1.6
*-------------------------------------------------------------------------------
* Include               : Nothing
*-------------------------------------------------------------------------------
* Declaration           : unsigned char judge_reversing_code(
*                       :       unsigned char,
*                       :       unsigned char                   )
*-------------------------------------------------------------------------------
* Function              : Reversing data code judgment
*                       :
*-------------------------------------------------------------------------------
* Arguments             : rtn0 : judgment of data code "L" (0data)return value
*                       :               0: OK
*                       :               1: NG
*                       : rtn1 : judgment of data code "L" (1data)return value
*                       :               0: OK
*                       :               1: NG
*-------------------------------------------------------------------------------
* Returns               : 0:OK
*                       : 1:NG
*-------------------------------------------------------------------------------
* Input                 : unsigned char code_low_cnt    :   Reception code "L" counter
*                       : unsigned char rev_cnt         :   Buffer position for
*                       :                                   reversing data confirmation
* Output                : unsigned char code_low_cnt    :   Reception code "L" counter
*                       : unsigned char rev_cnt         :   Buffer position for
```

```
*                       :                                   reversing data confirmation
*-------------------------------------------------------------------------------
* Call functions        : set_reversing_code()
*                       : cmp_reversing_code()
*-------------------------------------------------------------------------------
* Note                  :
*-------------------------------------------------------------------------------
* History               : 2005.07.30    Ver 1.00
*""FUNC COMMENT END""*****************************************************************/
unsigned char judge_reversing_code(unsigned char rtn0, unsigned char rtn1)
{

    unsigned char rtn;          /* Return value */


    code_low_cnt++;             /* Reception code "L" counter add */

    if(code_low_cnt <= DATA_MAX_8_BIT_CNT){
        /* When it is not reversing data */
        set_reversing_code(rtn0, rtn1);     /* set_reversing_code function */

    }
    else if (code_low_cnt <= DATA_MAX_LOW_BIT_CNT){

        /* When the reversing data code */
        if(code_low_cnt == DATA_MAX_8_BIT_CNT+1){
            /* When the reversing data of the first data */
            rev_cnt = 0;    /* Buffer position for reversing data confirmation clear */
        }

        rtn = cmp_reversing_code(rtn0, rtn1);       /* cmp_reversing_code function */
        if(rtn != OK){
            /* When reversing data is NG */
            return NG;
        }
    }
    else{
            return NG;
    }

    if(rev_cnt > REV_PULSE_MAX){
        rev_cnt = 0;            /* Buffer position for reversing data confirmation clear */
    }
    return OK;

}
```

```
/*""FUNC COMMENT""*****************************************************************
* ID                    : 1.7
*---------------------------------------------------------------------------
* Include               : Nothing
*---------------------------------------------------------------------------
* Declaration           : void set_reversing_code(
*                       :       unsigned char,
*                       :       unsigned char        )
*---------------------------------------------------------------------------
* Function              : Reversing data code buffer setting
*                       :
*---------------------------------------------------------------------------
* Arguments             : rtn0 : judgment of data code "L" (0data)return value
*                       :               0: OK
*                       :               1: NG
*                       : rtn1 : judgment of data code "L" (1data)return value
*                       :               0: OK
*                       :               1: NG
*---------------------------------------------------------------------------
* Returns               : Nothing
*---------------------------------------------------------------------------
* Input                 : Nothing
* Output                : unsigned char rev_pulse[]   : Buffer for reversing data confirmation
*                       : unsigned char rev_cnt       : Buffer position for
*                       :                               reversing data confirmation
*---------------------------------------------------------------------------
* Call functions        : Nothing
*---------------------------------------------------------------------------
* Note                  :
*---------------------------------------------------------------------------
* History               : 2005.07.30    Ver 1.00
*""FUNC COMMENT END""*************************************************************/
void set_reversing_code(unsigned char rtn0, unsigned char rtn1)
{

    if(rtn0 == OK){
        /* When data is 0 */
        rev_pulse[rev_cnt] = 0xF1;      /* Reversing code(0xF1) storage */
    }
    else if(rtn1 == OK){
        /* When data is 1 */
        rev_pulse[rev_cnt] = 0xF0;      /* Reversing code(0xF0) storage */
    }
    rev_cnt++;                          /* Buffer position for reversing data confirmation add */
}
```

```
/*""FUNC COMMENT""*****************************************************************
* ID                    : 1.8
*---------------------------------------------------------------------------------
* Include               : Nothing
*---------------------------------------------------------------------------------
* Declaration           : unsigned char cmp_reversing_code(
*                       :       unsigned char,
*                       :       unsigned char                   )
*---------------------------------------------------------------------------------
* Function              : Reversing data code buffer setting
*                       :
*---------------------------------------------------------------------------------
* Arguments             : rtn0 : judgment of data code "L" (0data)return value
*                       :               0: OK
*                       :               1: NG
*                       : rtn1: judgment of data code "L" (1data)return value
*                       :               0: OK
*                       :               1: NG
*---------------------------------------------------------------------------------
* Returns               : 0:OK
*                       : 1:NG
*---------------------------------------------------------------------------------
* Input                 : unsigned char rev_pulse[] : Buffer for reversing data confirmation
*                       : unsigned char rev_cnt     : Buffer position for
*                       :                            reversing data confirmation
* Output                : unsigned char rev_cnt     : Buffer position for
*                       :                            reversing data confirmation
*---------------------------------------------------------------------------------
* Call functions : Nothing
*---------------------------------------------------------------------------------
* Note                  :
*---------------------------------------------------------------------------------
* History               : 2005.07.30    Ver 1.00
*""FUNC COMMENT END""*************************************************************/
unsigned char cmp_reversing_code(unsigned char rtn0, unsigned char rtn1)
{

    /* When data is reversing data */
    if( rtn0 == OK ){
        /* When data is 0 */
        if(rev_pulse[rev_cnt] == 0xF0){
            /* When data reverses */
            rev_cnt++;
            return OK;
        }
    }
```

```
        else if(rtn1 == OK){
            /* When data is 1 */
            if(rev_pulse[rev_cnt] == 0xF1 ){
                /* When data reverses */
                rev_cnt++;
                return OK;
            }
        }
        return NG;

}
```

## 10. Reference Documents

Hardware Manual
M16C/85 Group Hardware Manual Rev. 1.03
The latest version can be downloaded from the Renesas Technology website.

Technical Update/Technical News
The latest information can be downloaded from the Renesas Technology website.

## Website and Support

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

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

| REVISION HISTORY | M32C/85 Group<br>Remote Control Reception |
|---|---|

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.00 | Dec 10, 2006 | – | First Edition issued |
| | | | |

## Notes regarding these materials

1. This document is provided for reference purposes only so that Renesas customers may select the appropriate Renesas products for their use. Renesas neither makes warranties or representations with respect to the accuracy or completeness of the information contained in this document nor grants any license to any intellectual property rights or any other rights of Renesas or any third party with respect to the information in this document.
2. Renesas shall have no liability for damages or infringement of any intellectual property or other rights arising out of the use of any information in this document, including, but not limited to, product data, diagrams, charts, programs, algorithms, and application circuit examples.
3. You should not use the products or the technology described in this document for the purpose of military applications such as the development of weapons of mass destruction or for the purpose of any other military use. When exporting the products or technology described herein, you should follow the applicable export control laws and regulations, and procedures required by such laws and regulations.
4. All information included in this document such as product data, diagrams, charts, programs, algorithms, and application circuit examples, 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 products listed in this document, please confirm the latest product information with a Renesas sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas such as that disclosed through our website. (http://www.renesas.com )
5. Renesas has used reasonable care in compiling the information included in this document, but Renesas assumes no liability whatsoever for any damages incurred as a result of errors or omissions in the information included in this document.
6. When using or otherwise relying on the information in this document, you should evaluate the information in light of the total system before deciding about the applicability of such information to the intended application. Renesas makes no representations, warranties or guaranties regarding the suitability of its products for any particular application and specifically disclaims any liability arising out of the application and use of the information in this document or Renesas products.
7. With the exception of products specified by Renesas as suitable for automobile applications, Renesas products are not designed, manufactured or tested for applications or otherwise in systems the failure or malfunction of which may cause a direct threat to human life or create a risk of human injury or which require especially high quality and reliability such as safety systems, or equipment or systems for transportation and traffic, healthcare, combustion control, aerospace and aeronautics, nuclear power, or undersea communication transmission. If you are considering the use of our products for such purposes, please contact a Renesas sales office beforehand. Renesas shall have no liability for damages arising out of the uses set forth above.
8. Notwithstanding the preceding paragraph, you should not use Renesas products for the purposes listed below:
   (1) artificial life support devices or systems
   (2) surgical implantations
   (3) healthcare intervention (e.g., excision, administration of medication, etc.)
   (4) any other purposes that pose a direct threat to human life
   Renesas shall have no liability for damages arising out of the uses set forth in the above and purchasers who elect to use Renesas products in any of the foregoing applications shall indemnify and hold harmless Renesas Technology Corp., its affiliated companies and their officers, directors, and employees against any and all damages arising out of such applications.
9. You should use the products described herein within the range specified by Renesas, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas shall have no liability for malfunctions or damages arising out of the use of Renesas products beyond such specified ranges.
10. Although Renesas endeavors to improve the quality and reliability of its products, IC products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Please be sure to implement safety measures to guard against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas 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 applicable measures. Among others, since the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
11. In case Renesas products listed in this document are detached from the products to which the Renesas products are attached or affixed, the risk of accident such as swallowing by infants and small children is very high. You should implement safety measures so that Renesas products may not be easily detached from your products. Renesas shall have no liability for damages arising out of such detachment.
12. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written approval from Renesas.
13. Please contact a Renesas sales office if you have any questions regarding the information contained in this document, Renesas semiconductor products, or if you have any other inquiries.