# RL78/G13

## DALI-2 Communication driver

## Introduction

This application note describes a sample program that performs DALI (Digital Addressable Lighting Interface) communication using the RL78/G13 microcontroller.

The sample program operates as a slave (Control Gear). Processes the waveform of Manchester encoded DALI signal using peripheral functions (timer function) mounted on RL78/G13. For command analysis and command execution, use the DALI library for RL78. Low level signal processing is realized by software so that DALI communication can be performed even with a microcontroller that does not have dedicated hardware for DALI communication. When higher performance is required, we recommend a product (RL78/I1A) with dedicated hardware.

This application note assumes that you already have knowledge about DALI. For details about the DALI standard, please refer to the IEC 62386-101 and IEC 62386-102, Edition 2.0 standards and the "Lighting Communications Using RL78/I1A (Reception)" application note (R01AN1115EJ0300).

## Target Device

RL78/G13 (R5F100LE)

When applying the sample program covered in this application note to another RL78 microcontroller, conduct an extensive evaluation Before use.

# Contents

# 1. Specifications

Using the RL78/G13 Target Board (QB-R5F100LE-TB) with a sample program and a DALI conversion circuit a DALI Control Gear function can be realised.

Figure 1-1 shows the DALI system configuration, and Figure 1-2 shows the software configuration of the sample program.
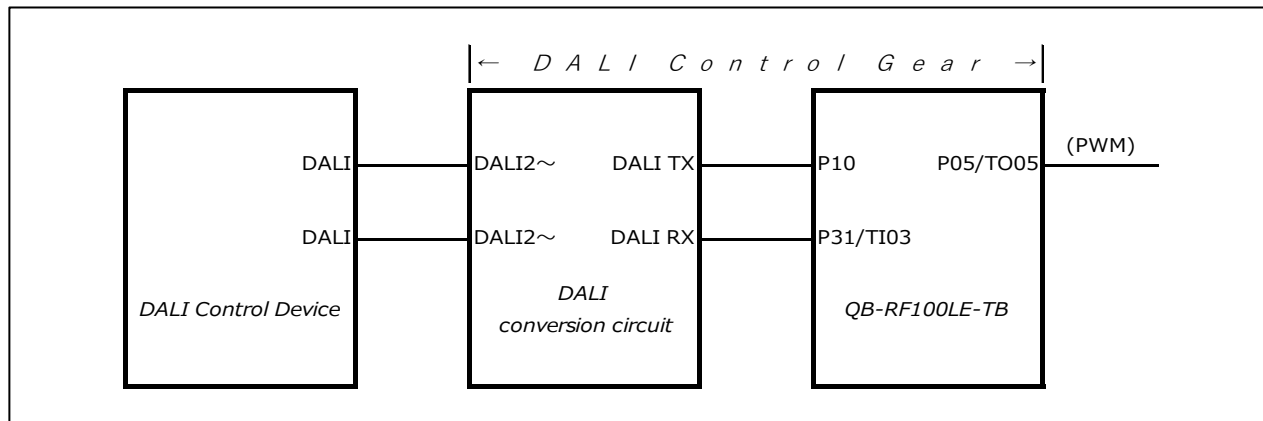


**Figure 1-1 System Configuration**
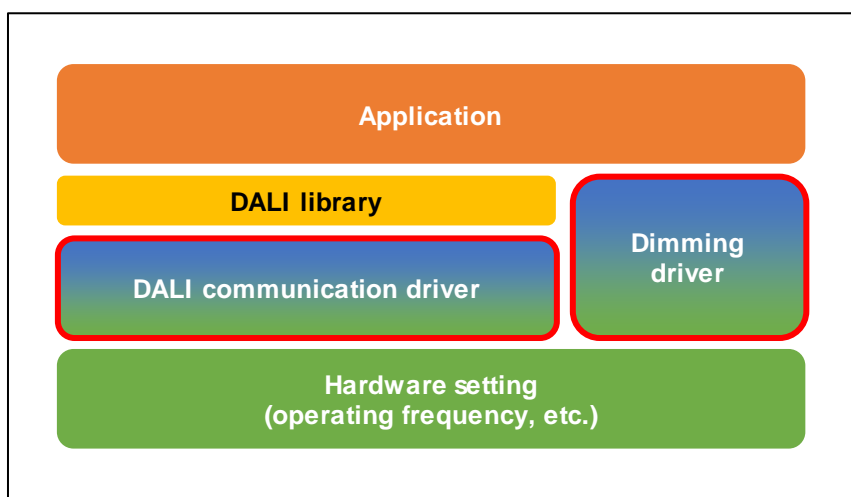


**Figure 1-2 Software configuration of sample program**

The specifications of the software configuration of the sample program are shown below.

- Application :
  Calls the DALI Communication driver, DALI Library and the Dimming driver.

- DALI Library :
  Analyses and implements DALI commands and timing management.
  For details, please refer to " RL78/I1A Family User's Manual DALI Library (IEC62386-102ed2.0)".

- DALI Communication driver (Receive driver and Transmit driver) :
  Receives and transmits the DALI Command.

- Dimming driver :
  Outputs the pulse width modulated dimming (PWM) signal to achieve the required dimming level..

- Hardware initialization processing :
  Initialization of peripheral functions and pins used in RL78/G13 using code generation.

This application note describes "DALI communication driver" and "Dimming driver" shown in Figure 1-2.

For DALI library, please refer to "RL78/I1A Family User's Manual DALI Library (IEC62386-102ed2.0)".

## 1.1    **Folder structure**

Folder structure of the sample program is shown below.
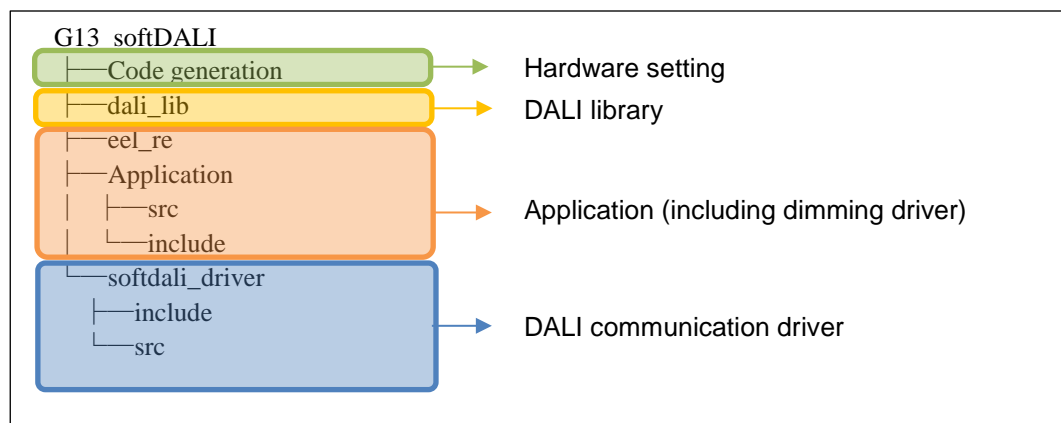


**Figure 1-3 Folder structure of sample program**

## 1.2    **File structure**

The file list of "DALI communication driver" and "Dimmer driver" described in this application note is shown.

**Table 1-1 File list**

| Driver name | File name | Function name |
|---|---|---|
| DALI communication driver (Receive) | r_softdali_rx_api.c | R_SD_RX_ApiInit |
| | | R_SD_RX_ApiGetCommand |
| | r_softdali_rx.c | r_sd_rx_dali_get_command |
| | | r_sd_rx_stop_condition_complete |
| | | r_sd_rx_check_recv_frame_bit |
| | | r_sd_rx_forward_bit_set |
| | | r_sd_rx_receive_status_reset |
| | r_softdali_rx_hw.c | r_sd_rx_hw_timer_init |
| | | r_sd_rx_hw_port_init |
| | | stop_condition_timer_interrupt |
| | | pulse_width_check_timer_interrupt |
| DALI communication driver (Transmit) | r_softdali_tx_api.c | R_SD_TX_ApiInit |
| | | R_SD_TX_ApiSendAnswer |
| | r_softdali_tx.c | r_sd_tx_set_output_data |
| | | r_sd_tx_put_halfbit_data |
| | r_softdali_tx_hw.c | r_sd_tx_hw_timer_init |
| | | r_sd_tx_hw_port_init |
| | | r_sd_tx_width_timer_start |
| | | r_sd_tx_width_timer_stop |
| | | r_sd_tx_put_port |
| | | tx_width_timer_interrupt |
| Dimming driver | r_lamp.c | Lamp_SetLevel |
| | | Lamp_IsOn |

## 1.3     DALI Communication driver

This driver consists of receive driver and transmit driver.

The Receive Driver receives and processes the DALI Forward Frame signal waveform required by the Control Gear. The Transmit Driver transmits the Backward Frame. Both the Forward and Backward frames are processed using software and the timers of the RL78/G13.

The specifications of each driver are shown below.

### 1.3.1     Receive driver

The Forward Frame received by DALI Control Gear consists of Start bit, 16 bits data and Stop condition, and each bit is Manchester encoded.

The specifications of the driver that receives Forward Frame are shown below.

1. Start bit and 16 bits data are received using input pulse interval measurement of the timer function, and judge Half bit, Double Half bit, or timing violation.

2. Decode the received Manchester encoded data as logical bits every 2 Half bits.

3. To detect the Stop condition, use an interval timer of the timer function, and reception is completed with after confirming the lapse of 1900μs from the last rising edge.

### 1.3.2     Transmit driver

The Backward Frame transmitted by DALI Control Gear consists of Start bit, 8 bits data and Stop condition, and each bit is Manchester encoded.

The specifications of the driver that transmits the Backward Frame are shown below.

1. Encode Start bit and 8 bits data to Manchester encode data.

2. According to the Manchester encoded data, change the output terminal setting at each Halfbit timing of DALI.

3. The Half bit timing is measured using the timer function interval timer.

4. Set the output terminal to high level at the timing of the stop condition.

## 1.4     Dimming driver

The specifications of the driver that outputs the dimming signal are shown below.

Depending on the Actual Level obtained from the DALI library, the PWM is output as a dimming signal using the timer function of RL78/G13.

## 2. Development Environment

The DALI Control Gear software solution is confirmed to operate in the following environment.

**Table 2-1 Development Environment**

| Item | Detail |
|---|---|
| DALI Control Device | Renesas Electronics Corporation<br>DALI Master Controller GUI<br>Tessera Technology Inc.<br>TCM-RL78I1A |
| DALI Control Gear | Renesas Electronics Corporation<br>RL78/G13 Target Board (QB-R5F100LE-TB)<br>DALI conversion circuit |
| IDE | Renesas Electronics Corporation<br>CS+ for CC V7.00.00 |
| Compiler | Renesas Electronics Corporation<br>CC-RL V1.70.00 |
| Library | Renesas Electronics Corporation<br>RL78 DALI Library V2.00 [Note1]<br>EEPROM Emulation Library Pack02 V2.00 [Note2] |

Note1 : About RL78 DALI library

This version is a sample.

Note2 : About the data flash library

Sample program is attached to this application note.

In order to operate the sample program, download the EEPROM Emulation Library Pack02 Package Ver.2.00(for CA78K0R/CC-RL Compiler) for RL78 Family separately and copy the library files to the following folder below "Workspace".

Workspace\eel_re\include : eel.h, eel_types.h, fdl.h, fdl_types.h

Workspace\eel_re\lib : eel.lib, fdl.lib

For the method of registering / linking to the project, refer to "Chapter 8 Sample Program" of R20UT2645.

The flash memory self-programming library is available on the Renesas Electronics Website. Please contact your local Renesas Electronics sales office or distributor for more information.

## 3.  Hardware Description

Describes the hardware of DALI Control Gear.
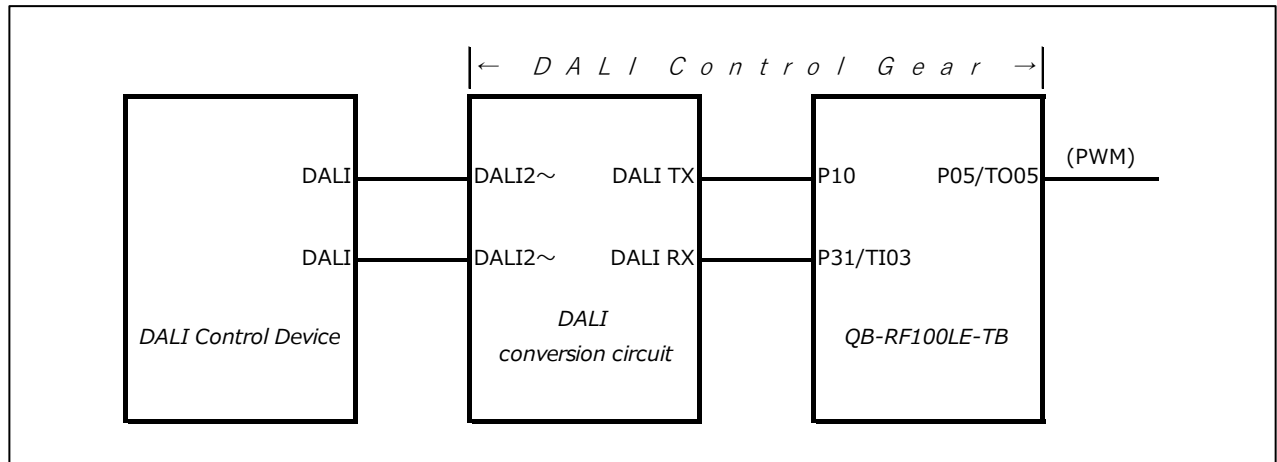


**Figure 3-1 How to connect DALI Control Device and DALI Control Gear**

Note

This circuit image is simplified to show the outline of the connection.

The connection between the DALI bus and the RL78 microcontroller is as follows.

- DALI transmission line (TX) is connected to output pin P10.

- DALI reception line (RX) is connected to timer input pin P31 / TI03.

- The dimming signal output is set to the timer output pin P05 / TO05.

The DALI conversion circuit diagram is shown below.
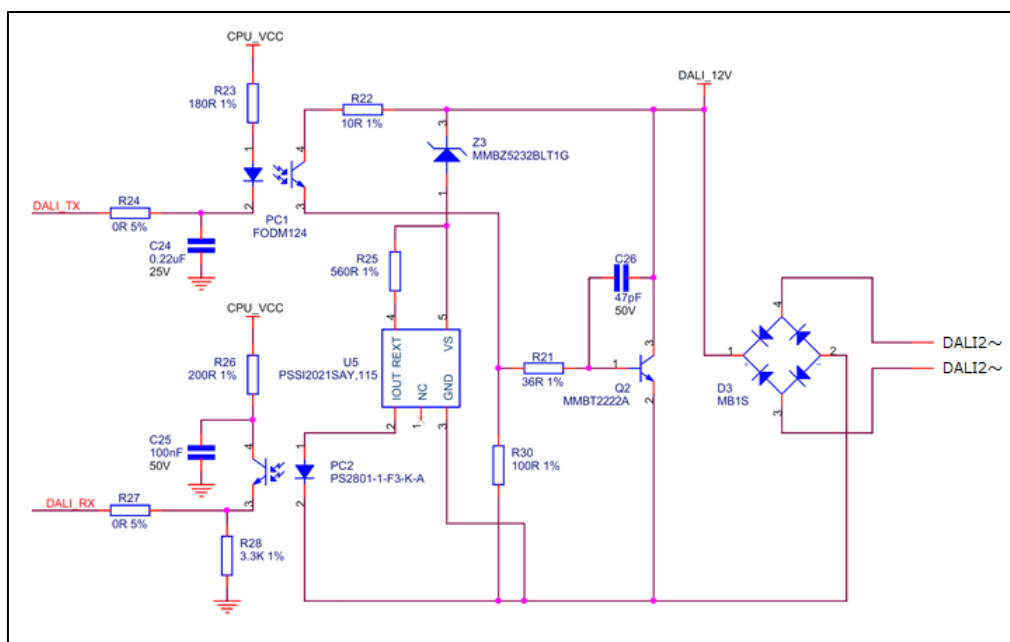


**Figure 3-2 DALI conversion circuit**

## 3.1 Microcontroller Resources

The DALI control gear software solution uses the following resources.

### 3.1.1 List of pins

Table 3-1 shows the pins used and their functions.

**Table 3-1 Pins and functions used**

| Pin Name | I/O | Description |
|---|---|---|
| P31 / TI03 | Input | Capture of reception waveform |
| P10 | Output | Output of transmission waveform |
| P05 / TO05 | Output | PWM output of the duty value corresponding to the dimming level |

### 3.1.2 List of timers

Table 3-2 shows the timers used and their functions.

**Table 3-2 Timers and functions used**

| Timer Name | Mode | Description |
|---|---|---|
| TAU0 channel0 | Interval timer | Monitor Half bit output time of transmitted waveform |
| TAU0 channel1 | Interval timer | 1ms cycle management for DALI library |
| TAU0 channel 2 | Interval timer | Monitor transmission start wait time |
| TAU0 channel 3 | Input pulse interval measurement | Pulse width measurement of received waveform |
| TAU0 channel 4 | PWM | PWM output (master) with 1KHz cycle |
| TAU0 channel 5 | PWM | PWM output (slave) with 1KHz cycle |
| TAU0 channel7 | Interval timer | Stop condition detection of received waveform |

## 4.   **Software Description**

### 4.1   **Operation Outline**

Processing of the sample program is shown below.

1. Initialize the functions to be used, the DALI library, and each driver.
   Note: Initialization of peripheral functions (hdwinit function) is called before the main function.

2. The whole cycle management is performed with 1ms of the interval timer.

3. When a Forward Frame is received. it is decoded by the Receive driver and passed to the DALI library.

4. The DALI library analyzes the received data and makes a command decision.

5. If there is a need to respond with a command, wait for the transmission start wait time, then encode the response data with the transmission driver and transmit the Backward Frame.

6. Output the dimming signal (PWM) according to the Actual Level set by the DALI library.

The above contents are shown in the flowchart.
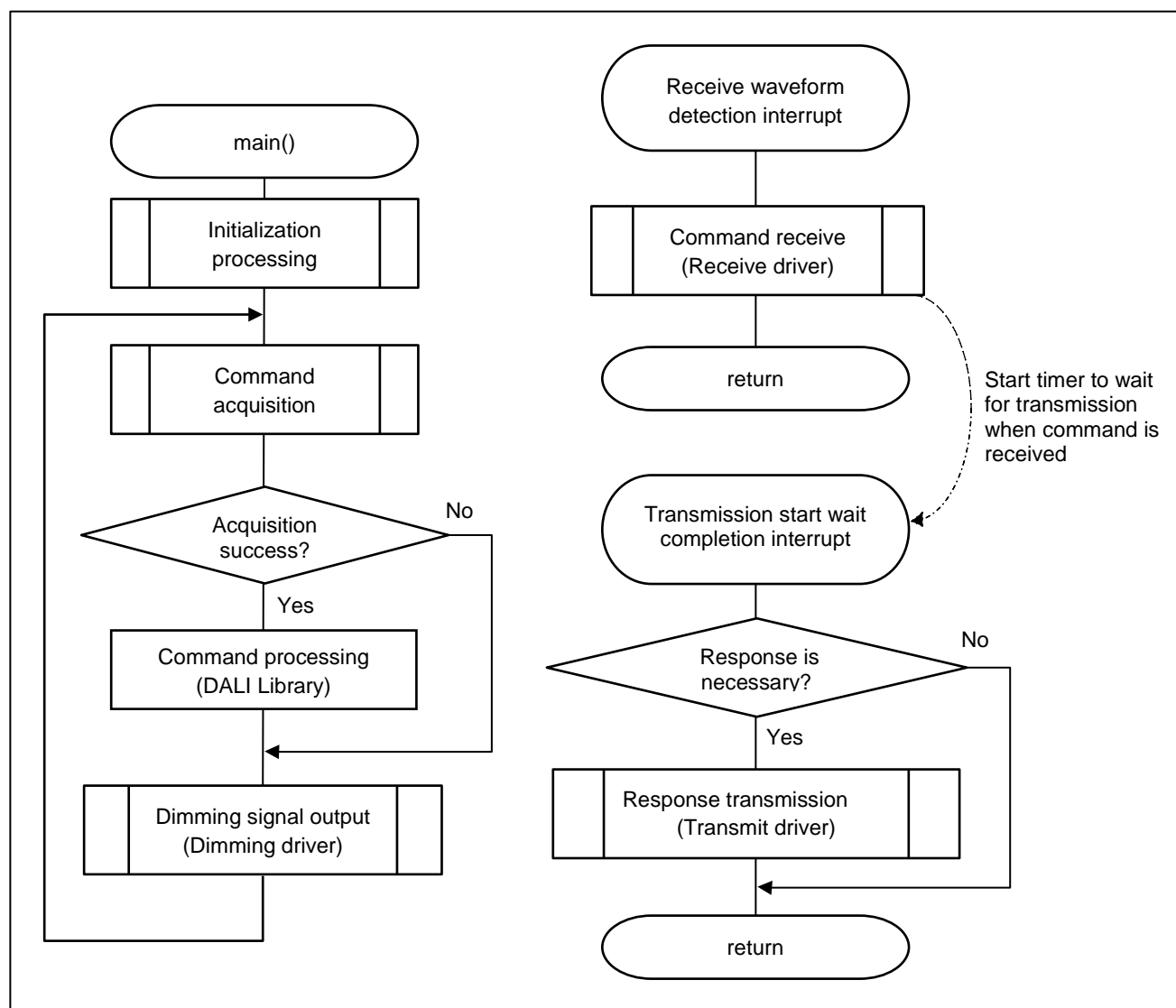


Figure 4-1 Software operation flowchart

## 4.2 Setting the function to be used

The setting of the microcomputer function is done using CS + code generation function. The setting is shown below. The setting made by code generation is executed by the startup routine (hdwinit function).

### 4.2.1 Setting of pins to be used

Table 4-1 shows the pin settings.

**Table 4-1 List of used pin settings**

| Pin Name | Description | I/O |
|----------|-------------|-----|
| P05 / TO05 | Timer output of TAU 0 | Output |

### 4.2.2 Setting of timers to be used

Table 4-2 shows the setting of each channel of Timer Array Unit 0.

**Table 4-2 List of used TAU0 channels settings**

| Timer Name | Mode | Setting time / Cycle | Interrupt |
|------------|------|----------------------|-----------|
| TAU0 channel0 | Unused | - | - |
| TAU0 channel1 | Interval timer | 1ms | Not use |
| TAU0 channel2 | Interval timer | 5ms | End of count |
| TAU0 channel3 | Unused | - | - |
| TAU0 channel4 | PWM | 1ms | Not use |
| TAU0 channel5 | PWM | 0% | Not use |
| TAU0 channel6 | Unused | - | - |
| TAU0 channel7 | Unused | - | - |

Note: TAU0 channel 0 is used as the transmit driver and TAU 0 channels 3 and 7 are used as the receive driver. Since the timer used in the transmit / receive driver is set by R_SD_RX_ApiInit (), R_SD_TX_ApiInit (), set "Unused".

### 4.2.3 List of Option Byte settings

Table 4-3 shows the Option Byte setting.

**Table 4-3 List of Option Byte settings**

| Address | Value | Description |
|---------|-------|-------------|
| 000C0H / 010C0H | 11111111B | Enables the watchdog timer. (Starts counting after the release from the reset state.) |
| 000C1H / 010C1H | 00011111B | LVD reset mode, 2.81V (2.75V～2.87V) |
| 000C2H / 010C2H | 11101000B | HS mode, HOCO: 32 MHz |
| 000C3H / 010C3H | 10000100B | Enables the on-chip debugger. |

### 4.2.4 **C language standard (C99)**

This application note supports C99 standard. To create a project using the sample program, specify the CC-RL compile option "-lang = c99".

### 4.2.5 **Section setting**

Be sure to refer to "(2) Defining sections" in "8.3.2 When the CC-RL Compiler is Used" in "RL78 Family EEPROM Emulation Library Pack02 Package Ver.2.00 Release Note" and make sure to set the section.

### 4.2.6 **Edit hwdinit() function**

In code generation, there are some codes which are deleted by regeneration.

When regeneration is done refer to the following code, please copy the red frame line and add it.

```
r_system.c

void hdwinit(void)
{
    /* Create seed value used for CRCD */
    make_ram_seed();
    DI();
    R_Systeminit();
    EI();
}
```

Note: About the make_ram_seed() function

In the make_ram_seed() function a random number used by the DALI Control Gear is generated.

Please use this function according to your requirements.

### 4.3    Receive driver

The method of decoding the received waveform which conforms to the Receiver timing standard of "IEC 62386-101 ed.2.0" is described below.

#### 4.3.1    Receive driver outline

The outline of the receive driver is shown below.

1.  A waveform passed through the DALI conversion circuit is input to P31/TI03.

2.  Processing of the received waveform uses two channels (3, 7) of Timer Array Unit 0.

3.  Channel 3 measures the pulse width of the received waveform using the input pulse interval measurement mode (both edge detection).

4.  Channel 7 is set to interval timer mode and used for detection of stop condition.

5.  After detecting the stop condition, the reception of the received waveform is completed.

Figure 4-2 shows the state transitions of the receive driver.
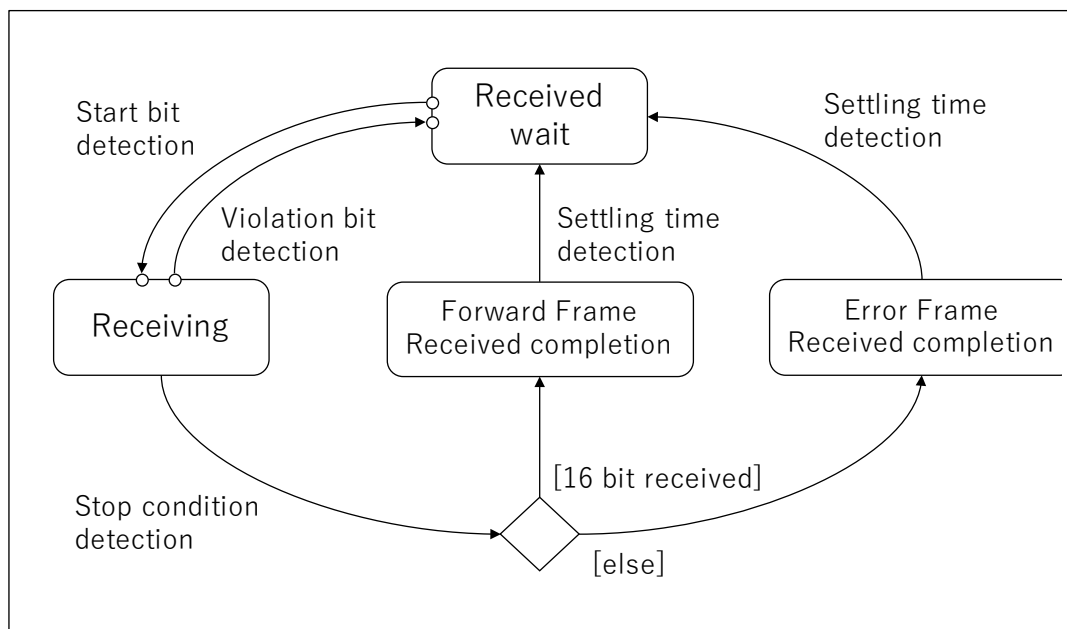


**Figure 4-2 Receive driver state transition**

Table 4-4 and Table 4-5 list the pins and functions used by the receive driver.

The following settings are implemented in R_SD_RX_ApiInit ().

**Table 4-4 Used pins and functions**

| Pin Name | Description | I/O |
|---|---|---|
| P31 / TI03 | Input function of TAU0 | Input |

**Table 4-5 Used timer and function**

| Timer Name | Mode | Setting time / Cycle | Interrupt |
|---|---|---|---|
| TAU0 channel3 | Input pulse interval measurement | Rising / falling edge detection | Edge is detected |
| TAU0 channel7 | Interval timer | 1900µs | End of count |

### 4.3.2 **Function list**

Table 4-6 and Table 4-7 list the function list of the receive driver.

Table 4-6 Receive driver external function list

| Function name | Description |
|---|---|
| R_SD_RX_ApiInit | Receive driver initialization function |
| R_SD_RX_ApiGetCommand | Receive command acquisition function |

Table 4-7 Receive driver internal function list

| Function name | Description |
|---|---|
| r_sd_rx_dali_get_command | Receive command acquisition function |
| r_sd_rx_stop_condition_complete | Stop condition completion processing function |
| r_sd_rx_check_recv_frame_bit | Frame bit timing confirmation processing function |
| r_sd_rx_forward_bit_set | Forward Frame decoding processing |
| r_sd_rx_receive_status_reset | Reception state clear processing |
| r_sd_rx_hw_timer_init | Timer initialization function |
| r_sd_rx_hw_port_init | Port initialization function |
| stop_condition_timer_interrupt | Stop condition timer interrupt handling function |
| pulse_width_check_timer_interrupt | Pulse width measurement interrupt processing function |

### 4.3.3 **Function specification**

The specifications of the major functions of the receive driver are shown below.

(1) Hardware independent function

The specifications of hardware independent functions are described below.

To use these functions you need to include r_softdali_rx_api.h.

| Format | void R_SD_RX_ApiInit(void) | | |
|---|---|---|---|
| Parameters | I/O | Data type | Summary |
| - | - | void | - |
| Return value | | Data type | Summary |
| | | void | - |
| Summary | Initialization processing of receive driver | | |
| Details of processing | Perform initialization processing to use the receive driver.<br>• Call the hardware dependent part initialization function of the receiving driver<br>• Save initialization processing completion status | | |
| Notice | Be sure to call it before using receive driver.<br>Do not call more than once. | | |

| Format | uint8_t R_SD_RX_ApiGetCommand(uint16_t * received_frame) | | |
|---|---|---|---|
| Parameters | I/O | Data type | Summary |
| received_frame | I | uint16_t * | Received frame data |
| Return value | | Data type | Summary |
| | | uint8_t | SD_RET_NOT_INITIALIZE<br>:Not initialize<br>SD_RET_FF_RECEIVED<br>:Forward frame normal received<br>SD_RET_FF_ERROR_RECEPTION<br>:Forward frame error reception<br>SD_RET_FF_NOT_RECEIVED<br>:Forward frame not received |
| Summary | Receive data acquisition processing | | |
| Details of processing | Acquire received data of Forward Frame with receiving driver.<br>  To confirm the reception status, judge the status from the return code of the reception command acquisition function.<br>• If Forward Frame is received<br>  ➢ Set the received data to received_frame and return SD_RET_FF_RECEIVED.<br>• If Error Frame is received<br>  ➢ Return SD_RET_FF_ERROR_RECEPTION<br>• If not received<br>  ➢ Return SD_RET_FF_NOT_RECEIVED | | |
| Notice | Call R_SD_RX_ApiInit () before using it. | | |

(2)   Hardware dependent function
   The specifications of hardware dependent functions are described below.

| Format | void r_sd_rx_hw_timer_init(void) | | |
|---|---|---|---|
| Parameters | I/O | Data type | Summary |
| - | - | void | - |
| Return value | | Data type | Summary |
| | | void | - |
| Summary | Timer function initialization processing of receive driver | | |
| Details of processing | Set the necessary timer for the receive driver. <ul><li>Stop condition measurement timer setting</li><li>Setting of DALI communication input pulse measurement timer</li><li>Noise filter setting of DALI reception port</li><li>Input pulse of DALI communication reception port Noise filter setting of measurement timer</li><li>Input pulse measurement timer interrupt setting</li><li>Start input pulse measurement timer</li></ul> | | |
| Notice | Be sure to call it before starting DALI communication. | | |

| Format | void r_sd_rx_hw_port_init(void) | | |
|---|---|---|---|
| Parameters | I/O | Data type | Summary |
| - | - | void | - |
| Return value | | Data type | Summary |
| | | void | - |
| Summary | Port initialization processing of receive driver | | |
| Details of processing | Configure the necessary ports for the receive driver. <ul><li>Input port setting</li></ul> | | |
| Notice | Be sure to call it before starting DALI communication. | | |

| Format | static void stop_condition_timer_interrupt(void) | | |
|---|---|---|---|
| Parameters | I/O | Data type | Summary |
| - | - | void | - |
| Return value | | Data type | Summary |
| | | void | - |
| Summary | Interval timer interrupt processing for stop condition measurement | | |
| Details of processing | Perform interval timer interrupt processing for stop condition measurement. <ul><li>Stop condition completion processing execution</li><li>DALI communication reception complete interrupt processing (using DALI library)</li><li>Stop interval timer</li></ul> | | |
| Notice | - | | |

| Format | static void pulse_width_check_timer_interrupt(void) | | |
|---|---|---|---|
| Parameters | I/O | Data type | Summary |
| - | - | void | |
| Return value | | Data type | Summary |
| | | void | |
| Summary | Receive pulse width measurement interrupt processing | | |
| Details of processing | Perform processing at interruption of the reception pulse width measurement timer of DALI communication.<br>• Obtain the port status (High/Low) of the receiving port.<br>• Get the pulse counter value.<br>   ➢ When Overflow occurs<br>     Set 0xFFFF<br>   ➢ Normal time<br>     Set acquisition value<br>• Port status<br>   ➢ High<br>     Start Stop condition timer<br>   ➢ Low<br>     Stop Stop condition timer<br>• Call of frame bit confirmation processing function | | |
| Notice | - | | |

### 4.3.4     **Receiver timing definition**

IEC 62386-101 ed.2.0 Receiver timing standard and Receiver timing with this driver.

**(1)    Definition in the driver**

In this driver, each logical bit shown in Figure 4-3 is divided into the following two sections.

Ⓐ First section           : From the beginning edge of the logical bit to the next edge

Ⓑ Second section        : From the internal edge of the logical bit to the next edge

Measure the pulse width in each section and judge each logical bit shown in Figure 4-3 from the measured time.



Logical bit

   Start bit : Communication protocol start bit

   Half bit : Half bit of Manchester code

   Double half bit : Two half bits

   Stop condition : Communication protocol termination

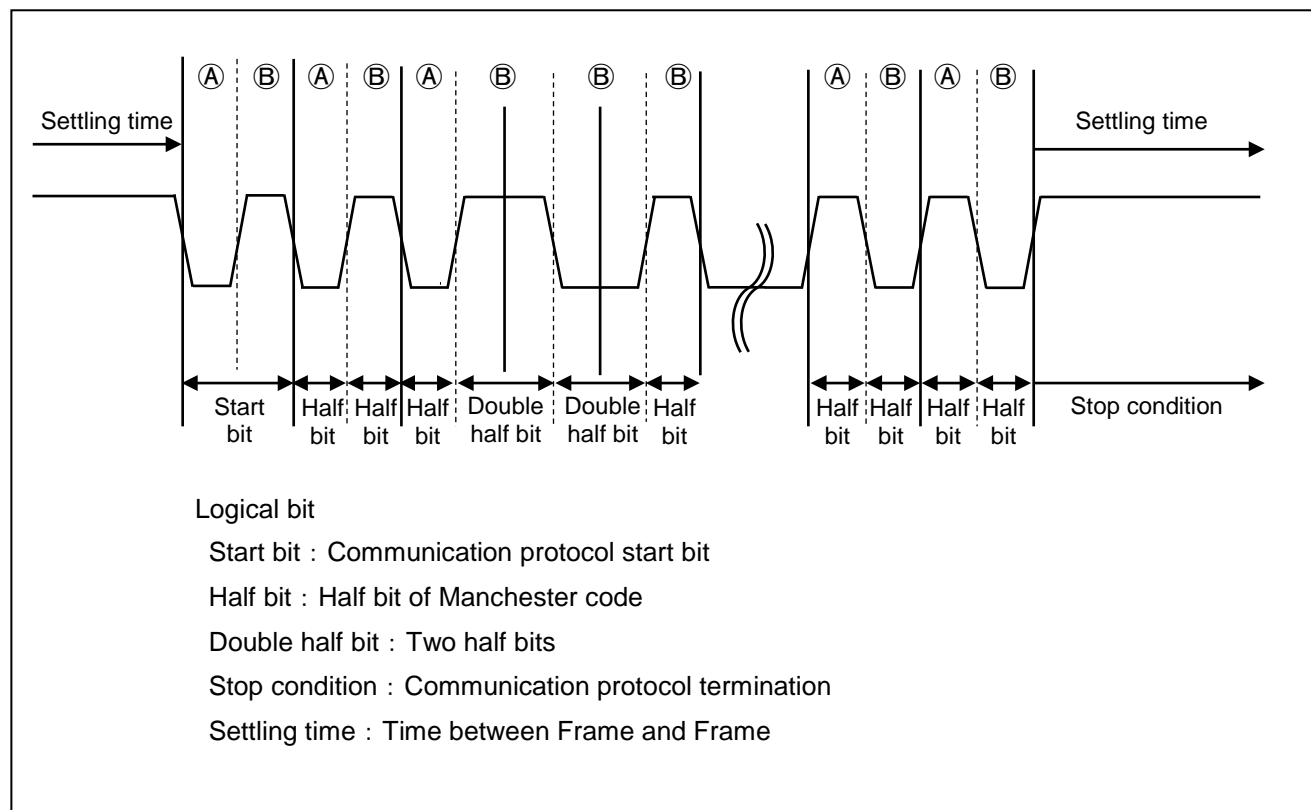   Settling time : Time between Frame and Frame

Figure 4-3 Name in driver

Receiver bit timing in each section is defined in the next chapter.

(2) Receiver bit timing definition of First section

Receiver bit timing used in First section is shown below.

Table 4-8 shows the Receiver bit timing defined in IEC 62386-101 ed.2.0 Specification Table 18.

**Table 4-8 Receiver bit timing of First section (Definition of the standard)**

| Minimum | Typical | Maximum | Description |
|---------|---------|---------|-------------|
|         |         | < 333.3µs | Gray area |
| 333.3µs | 416.7µs | 500µs | Half bit |
| > 500µs |         | < 750µs | Gray area |
| 750µs   |         | 1400µs | Bit timing violation |
| > 1400µs |        | < 2400µs | Gray area |
| 2400µs  |         |         | Stop condition |

In this driver, Receiver bit timing is redefined as shown in **Table 4-9** and Figure 4-4, and each bit is judged.

**Table 4-9 Receiver bit timing of First section (Definition of receive driver)**

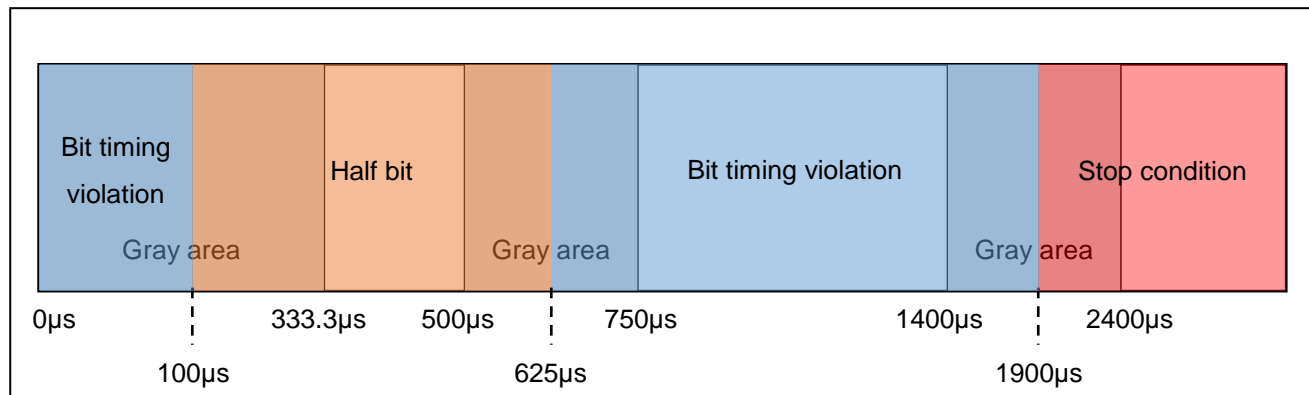| Minimum | Typical | Maximum | Description |
|---------|---------|---------|-------------|
|         |         | < 100µs | Bit timing violation |
| 100µs   | 416.7µs | 625µs | Half bit |
| > 625µs |         | < 1900µs | Bit timing violation |
| 1900µs  |         |         | Stop condition |



Figure 4-4 Receiver bit timing of First section

(3)   Receiver bit timing definition of Second section
Receiver bit timing used in Second section is shown below.

Table 4-10 shows the Receiver bit timing defined in IEC 62386-101 ed.2.0 Specification Table 19.

**Table 4-10 Receiver bit timing of Second section (Definition of the standard)**

| Minimum | Typical | Maximum | Description |
|---------|---------|---------|-------------|
|         |         | < 333.3µs | Gray area |
| 333.3µs | 416.7µs | 500µs | Half bit |
| > 500µs |         | < 666.7µs | Gray area |
| 666.7µs | 833.3µs | 1000µs | Double half bit |
| > 1000µs |        | < 1200µs | Gray area |
| 1200µs  |         | 1400µs | Bit timing violation |
| > 1400µs |        | < 2400µs | Gray area |
| 2400µs  |         |        | Stop condition |

In this driver, Receiver bit timing is redefined as shown in Table 4-11 and Figure 4-5, and each bit is judged.

**Table 4-11 Receiver bit timing of Second section (Definition of receive driver)**

| Minimum | Typical | Maximum | Description |
|---------|---------|---------|-------------|
|         |         | < 100µs | Bit timing violation |
| 100µs   | 416.7µs | 580µs | Half bit |
| > 580µs | 833.3µs | 1100µs | Double half bit |
| > 1100µs |        | < 1900µs | Bit timing violation |
| 1900µs  |         |        | Stop condition |
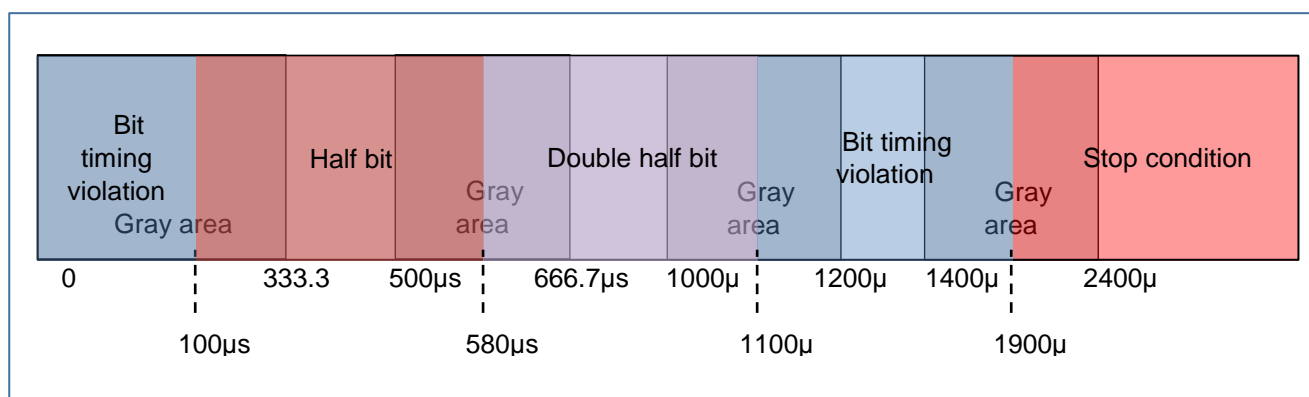


**Figure 4-5 Receiver bit timing of Second section**

### 4.3.5 Receiver bit timing judgement

The method of Receiver bit timing judgement is described below.

**(1) Receiver bit timing judgement method**

Measure the pulse width using both edge detection of input pulse interval measurement (channel 3 of Timer Array Unit 0). For the measured pulse width, as shown in Figure 4-6, judgment of Receiver bit timing is performed for each section. Stop condition does not change level and interrupt for measuring the input pulse interval does not enter, so it is detected by an interrupt by the channel 7 interval timer.

Note   Since the edge does not change in Stop condition, it is detected by the interval timer. Also, when the last logical bit is 1, the Half bit of the Second section and Stop condition are integrated, so when the Stop condition is detected, the Manchester code is decoded.
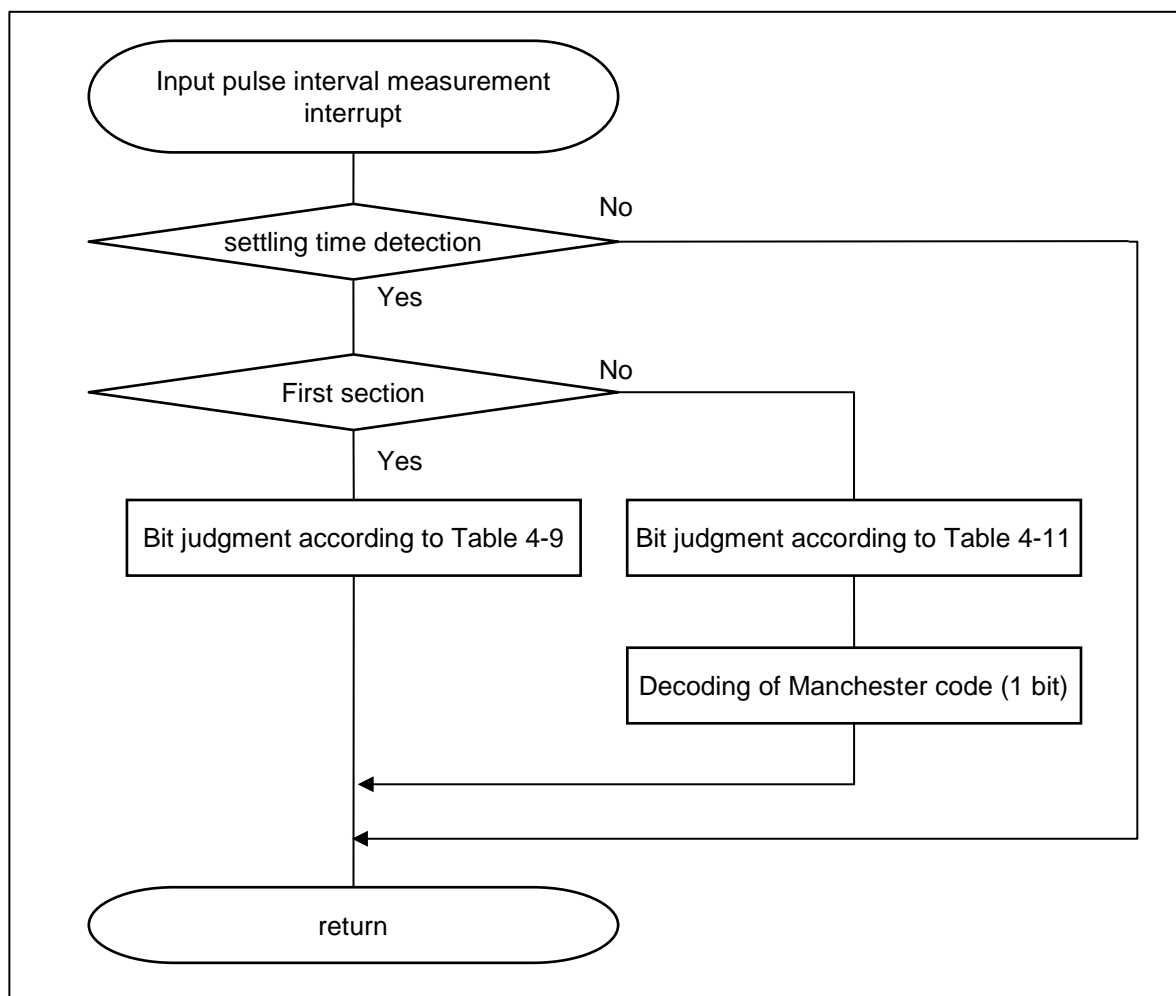


Figure 4-6 Receiver bit timing measurement flowchart

**(2)  Stop condition confirmation method**

How to check the stop condition by the interval timer is described below.

In the receive driver, it is judged as a stop condition when the state of the receive pin continues at high level for a certain period (1900 us in this sample program).

Specifically, as shown in Figure 4-7, the interval timer starts at the rising edge and stops at the falling edge. And, it is judged as a stop condition, when the interval timer interrupt occurs from the last rising edge.
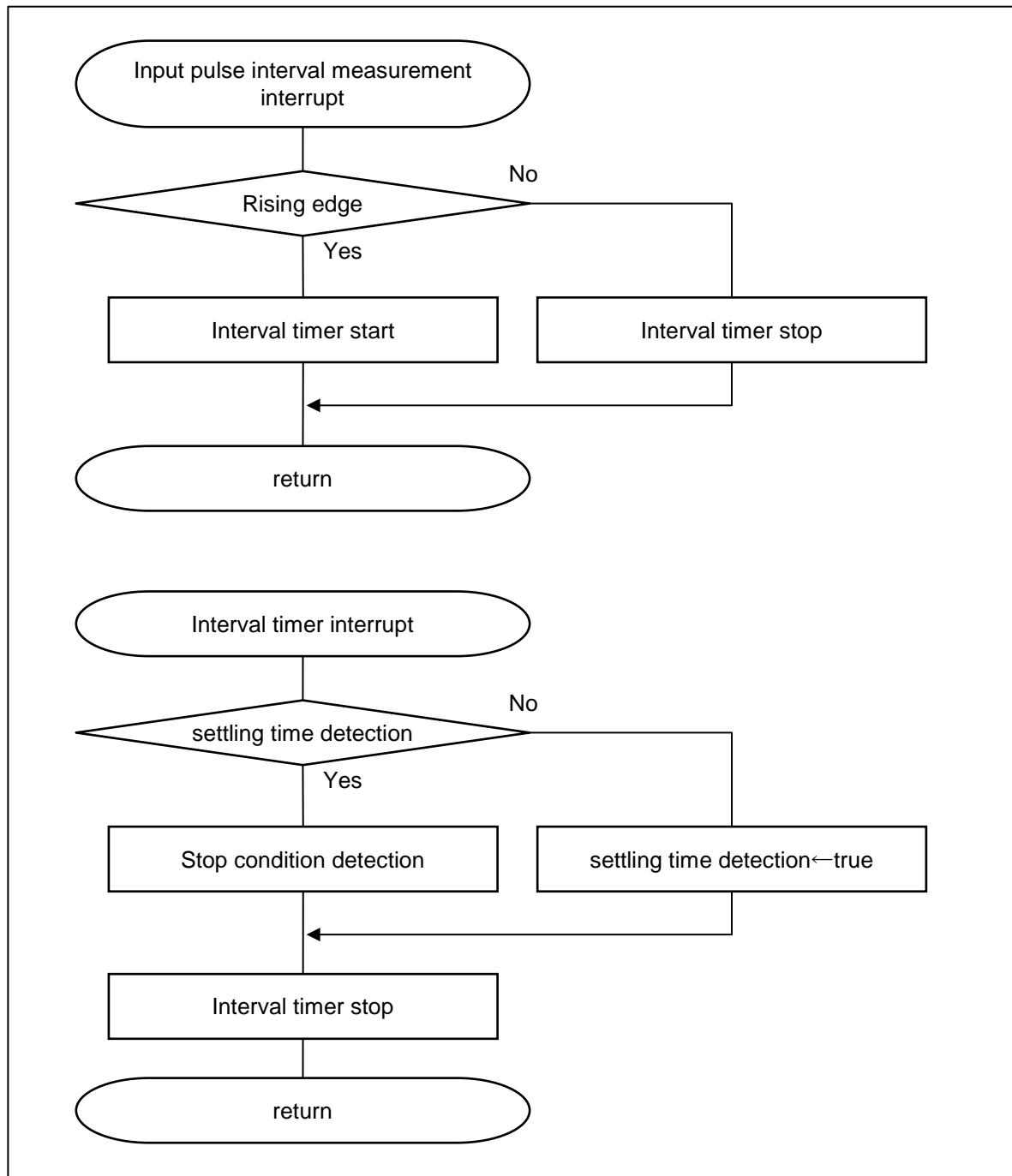


**Figure 4-7 Stop condition confirmation flowchart**

Figure 4-8 shows an example of pulse width measurement interrupts for the received waveform and start and stop of the interval timer.
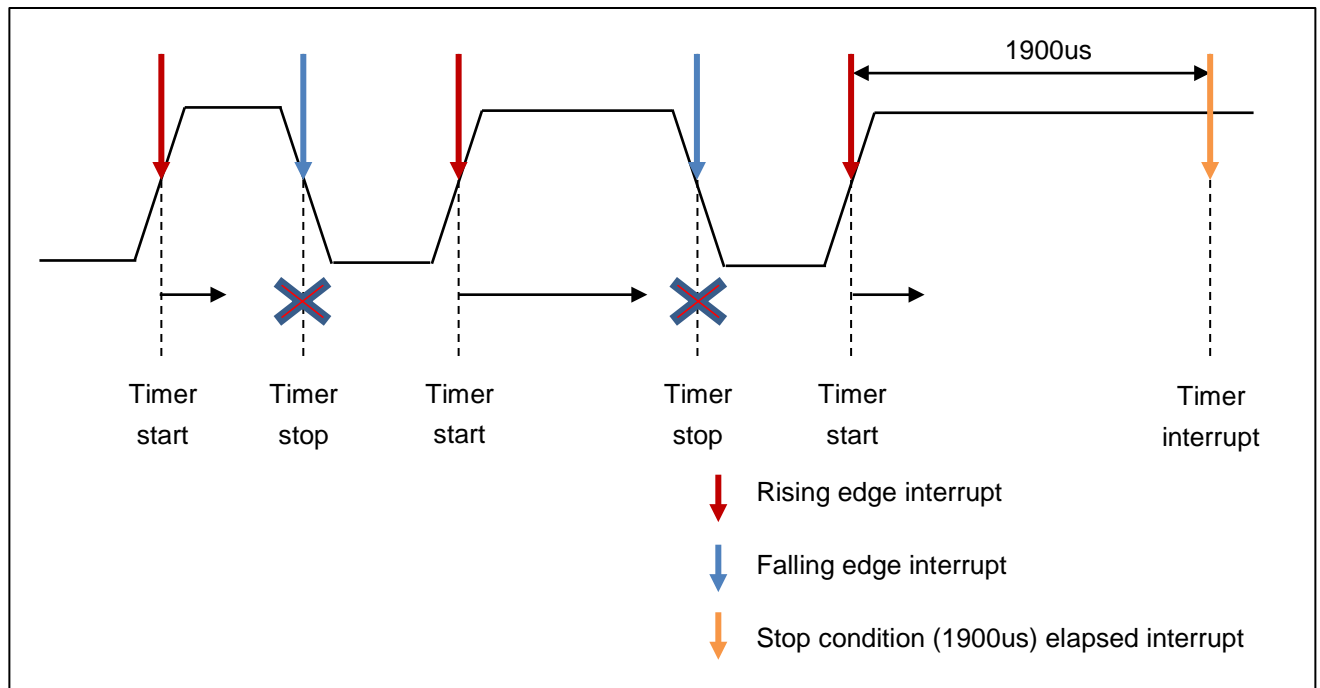


**Figure 4-8 Stop condition confirmation method**

### 4.3.6    Forward Frame decoding method

The decoding method of Forward Frame is described below.

**(1)  Forward Frame decode timing**

Forward Frame decoding timing is shown below.
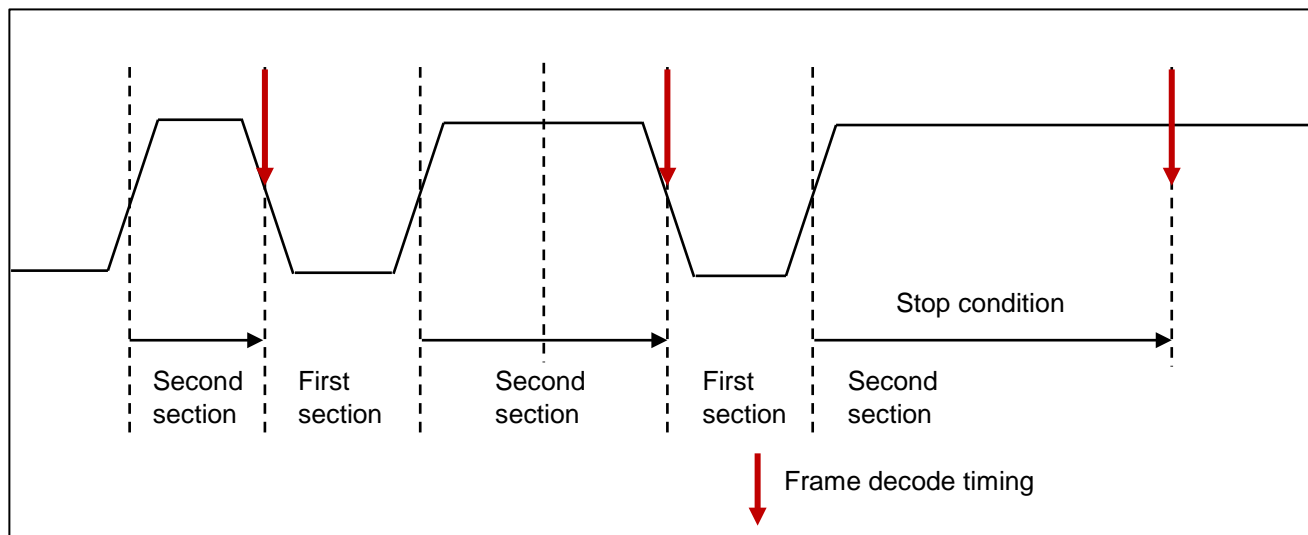


**Figure 4-9 Forward Frame decode timing**

Forward Frame decode timing is set when an interrupt occurs at the second section.

When a stop condition is detected, if it is the Second section, decode Forward Frame.

**(2)  Forward Frame setting data**

Since the received data is Manchester encoded, it decodes 1 at the rising edge and 0 at the falling edge.
See Figure 4-10.
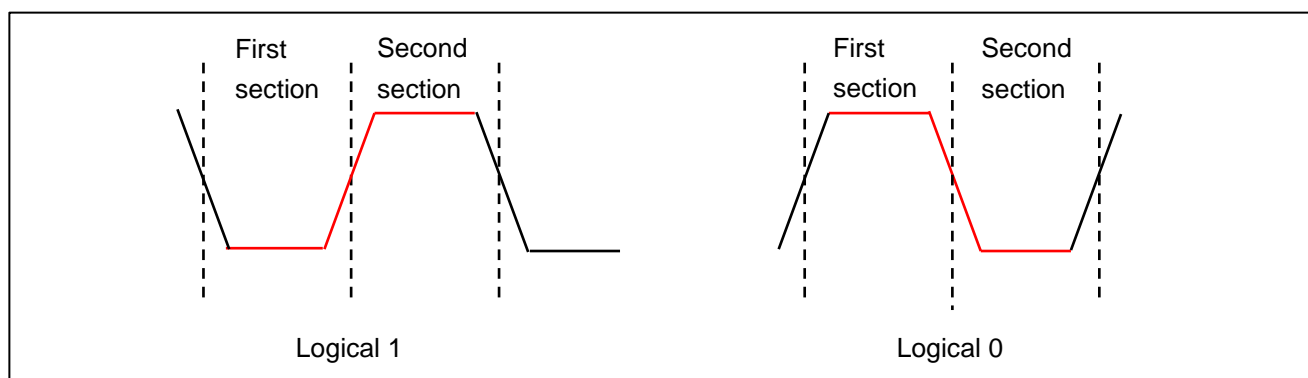


**Figure 4-10 Forward Frame data**

Note    If Second section is Double half bit, since no interrupt occurs at 2 Half bit, so it is decoded at 3 Half bit.

**Forward Frame data judgment**

The method of acquiring settings data for decoding Forward Frame is described below.
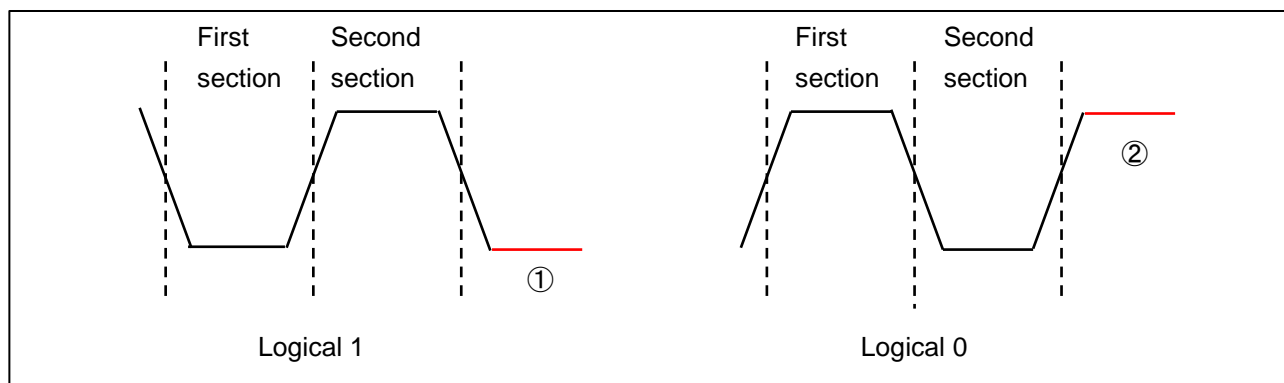


**Figure 4-11 How to judge Forward Frame data**

It checks the status of the output pins when an interrupt occurs of Second section.

Set "1" in the low-level state (1) and set "0" in the high-level state (2) to creates Frame data.

**Table 4-12 Decode from port status to received data**

| Status at frame decoding | Output pin status | Setting data |
|---|---|---|
| Half bit | low-level | "1" |
|  | high-level | "0" |
| Double half bit | low-level | "1" |
|  | high-level | "0" |
| Stop condition | No condition | "1" |

\* In Second section state at Stop condition, always First section is in the low-level state.

Other Frame decoding conditions

- First bit is not saved as data.
- Forward Frame is fixed to 16 bits, and if more data is received, it is recognized as a reception size error.
- Reception size error is recognized if Stop condition is received with less than 16 bits of received data.

## 4.4 Transmit driver

The method for realizing the encoding and waveform output of response data according to the Transmitter timing standard required for DALI Control Gear of IEC 62386-101 ed.2.0 with software is described below.

### 4.4.1 Transmit driver outline

Transmission waveform is output from the I/O pin P10.
The following is a summary of receiving the response data from the DALI Library and starting transmission.

1. The response data is Manchester encoded and stored in the Transmission Data Buffer.
2. Output the Half bit (low level) of the Start bit to the I/O pin, set the channel 0 of Timer Array Unit 0 to interval timer mode, and start the timer.

Figure 4-12 shows the operation flow of the transmit driver.
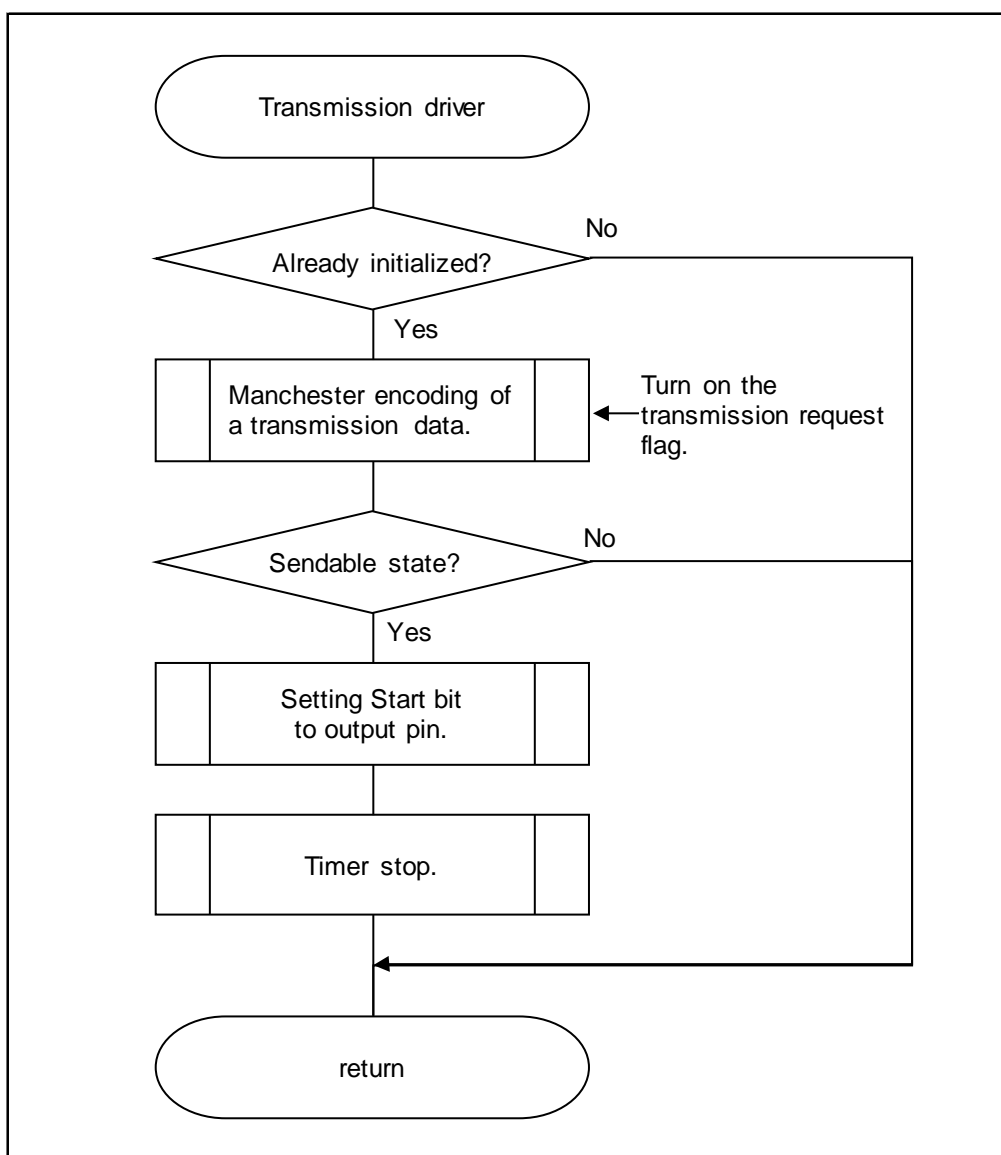


**Figure 4-12 Transmit Driver Flowchart**

The operation overview of interrupt processing by the timer started in the above flow is shown below.

1.  The interval timer generates an interrupt at every Half bit width.
2.  In interrupt handling, set the data in the transmit data buffer bit by bit to the I/O pin.
3.  Stop the interval timer when all the data in the transmit data buffer has been transmitted.
4.  Set the I/O pin to high-level and output Stop condition.

Figure 4-13 shows the interrupt handling flow of the transmit driver.
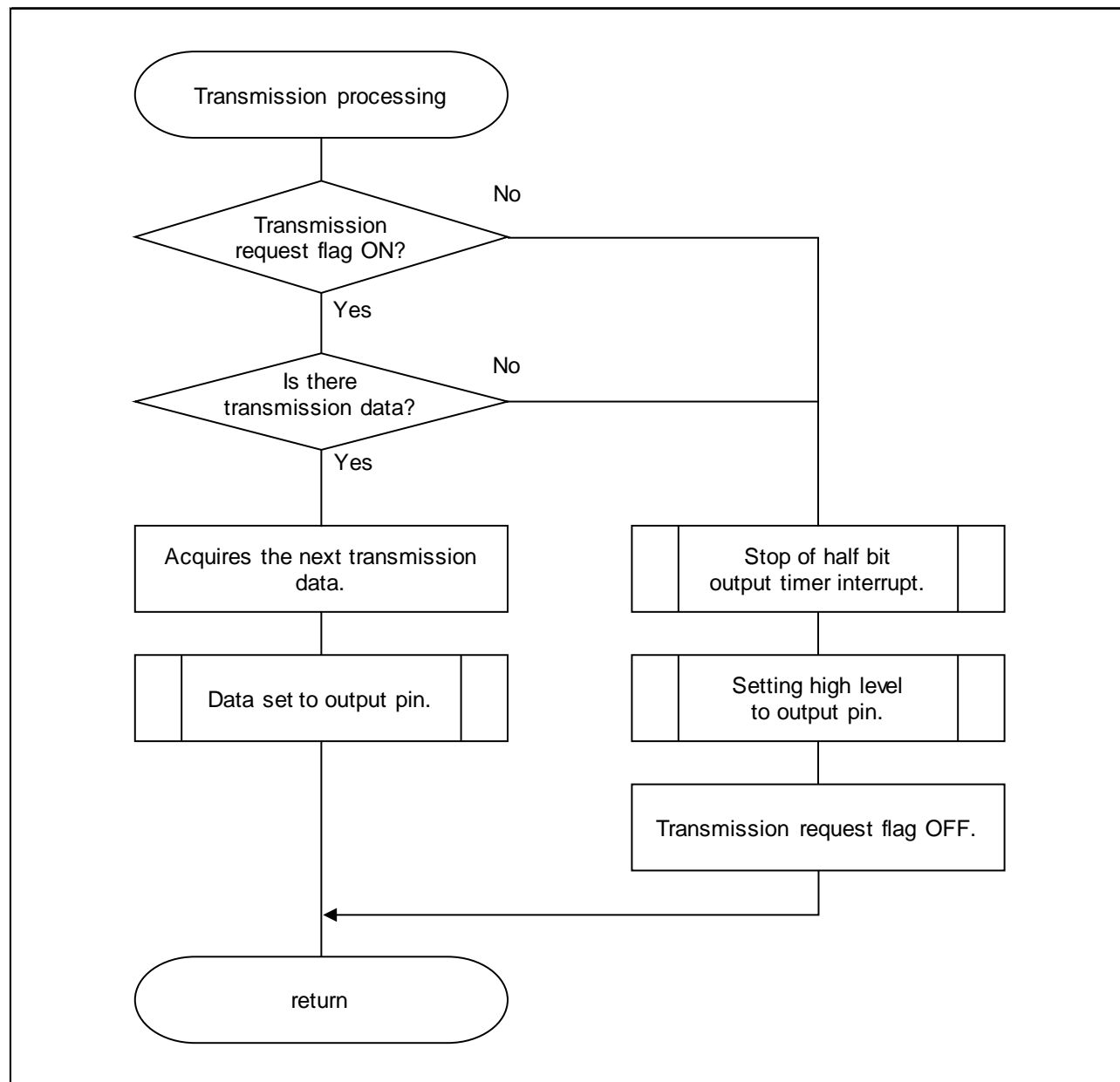


**Figure 4-13 Transmit driver interrupt processing flowchart**

Table 4-13 and Table 4-14 list the pins and functions used by the transmit driver.

**Table 4-13 Used pins and functions**

| Pin Name | Description | I/O |
|----------|-------------|--------|
| P10 | Output pin | Output |

**Table 4-14 Used timer and functions**

| Timer Name | Mode | Setting time / Cycle | Interrupt |
|------------|------|----------------------|-----------|
| TAU0 channel0 | Interval timer | 417µs | End of count |

#### 4.4.2 Function list

Table 4-15 and Table 4-16 list the functions of the transmit driver.

**Table 4-15 Transmit driver external function list**

| Function name | Description |
|---|---|
| R_SD_TX_ApiInit | Transmit driver initialization function |
| R_SD_TX_ApiSendAnswer | Command response transmission function |

**Table 4-16 Transmit driver internal function list**

| Function name | Description |
|---|---|
| r_sd_tx_set_output_data | Command response data setting function |
| r_sd_tx_put_halfbit_data | Command response data output function |
| r_sd_tx_hw_timer_init | Timer initialization function |
| r_sd_tx_hw_port_init | Port initialization function |
| r_sd_tx_width_timer_start | Transmit bit width timer start processing function |
| r_sd_tx_width_timer_stop | Transmit bit width timer stop processing function |
| r_sd_tx_put_port | Port output processing function |
| tx_width_timer_interrupt | Transmit bit width timer interrupt handling function |

### 4.4.3 Function specification

(1) Hardware independent function

The specifications of hardware independent functions are described below.

To use these functions you need to include r_softdali_tx_api.h.

| Format | void R_SD_TX_ApiInit(void) | | |
|---|---|---|---|
| Parameters | I/O | Data type | Summary |
| - | - | void | - |
| Return value | | Data type | Summary |
| | | void | - |
| Summary | Transmit driver initialization processing | | |
| Details of processing | Perform initialization processing to use the transmission driver.<br>• Call of hardware dependent part initialization function<br>• Save initialization processing completion status | | |
| Notice | Be sure to call it before using the transmission driver.<br>Do not call more than once. | | |

| Format | uint8_t R_SD_TX_ApiSendAnswer (uint8_t answer, uint8_t is_corrupted) | | |
|---|---|---|---|
| Parameters | I/O | Data type | Summary |
| answer | I | uint8_t | Transmission data |
| is_corrupted | I | uint8_t | Corrupted Backward Frame transmission instruction flag |
| Return value | | Data type | Summary |
| | | uint8_t | SD_RET_NOT_INITIALIZE<br>　:Not initialize<br>SD_RET_ERROR<br>　: Transmission data creation error<br>SD_RET_NORMAL<br>　:Backward frame normal send |
| Summary | Response data transmission processing of DALI command | | |
| Details of processing | Send BackwardFrame with the Transmit driver. | | |
| Notice | Call R_SD_TX_ApiInit () before using it. | | |

(2) Hardware dependent function

The specifications of hardware dependent functions are described below.

| Format | void r_sd_tx_hw_timer_init (void) | | |
|---|---|---|---|
| Parameters | I/O | Data type | Summary |
| - | - | void | - |
| Return value | | Data type | Summary |
| | | void | - |
| Summary | Timer function initialization processing of transmission driver | | |
| Details of processing | Configure the timer necessary for the transmission driver.<br>• Transmission pulse width measurement interval timer interrupt setting | | |
| Notice | Be sure to call it before starting DALI communication. | | |

| Format | void r_sd_tx_hw_timer_init (void) | | |
|---|---|---|---|
| Parameters | I/O | Data type | Summary |
| - | - | void | - |
| Return value | | Data type | Summary |
| | | void | - |
| Summary | Port initialization processing of transmission driver | | |
| Details of processing | Configure the necessary ports for the transmit driver.<br>• Output port setting | | |
| Notice | Be sure to call it before starting DALI communication. | | |

| Format | void r_sd_tx_width_timer_start (void) | | |
|---|---|---|---|
| Parameters | I/O | Data type | Summary |
| - | - | void | - |
| Return value | | Data type | Summary |
| | | void | - |
| Summary | Transmission pulse width measurement interval timer start processing | | |
| Details of processing | Perform transmission pulse width measurement interval timer start processing. | | |
| Notice | | | |

| Format | void r_sd_tx_width_timer_stop (void) | | |
|---|---|---|---|
| Parameters | I/O | Data type | Summary |
| - | - | void | - |
| Return value | | Data type | Summary |
| | | void | - |
| Summary | Transmission pulse width measurement interval timer stop processing | | |
| Details of processing | Perform transmission pulse width measurement interval timer stop processing.<br>If Corrupted Backward Frame is instructed, clear the Corrupted Backward Frame flag to the DALI library. | | |
| Notice | | | |

| Format | void r_sd_tx_put_port (uint8_t data) | | |
|---|---|---|---|
| Parameters | I/O | Data type | Summary |
| data | I | uint8_t | Port output data (0 or 1) |
| Return value | | Data type | Summary |
| | | void | - |
| Summary | Transmit bit port output processing | | |
| Details of processing | Perform processing to output the transmission bit to the port. | | |
| Notice | | | |

| Format | static void tx_width_timer_interrupt (void) | | |
|---|---|---|---|
| Parameters | I/O | Data type | Summary |
| - | - | void | - |
| Return value | | Data type | Summary |
| | | void | - |
| Summary | Transmit pulse width measurement interval timer interrupt processing | | |
| Details of processing | Transmit Pulse Width Measurement Performs processing at interrupt timer interrupt timer.<br>•   Call transmission data bit output processing | | |
| Notice | | | |

#### 4.4.4 Backward Frame encoding method

The encoding method of Backward Frame is described below.

Retrieve response data 1 bit at a time, Manchester encode into transmission data buffer and store "Backward Frame".

**Table 4-17 Manchester encoding example**

| Start bit | Response data (0x5A) | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 bit | | 7 bit | | 6 bit | | 5 bit | | 4 bit | | 3 bit | | 2 bit | | 1 bit | | 0 bit | |
| 1 | | 0 | | 1 | | 0 | | 1 | | 1 | | 0 | | 1 | | 0 | |
| set 0, 1 ↓ | | set 1, 0 ↓ | | set 0, 1 ↓ | | set 1, 0 ↓ | | set 0, 1 ↓ | | set 0, 1 ↓ | | set 1, 0 ↓ | | set 0, 1 ↓ | | set 1, 0 ↓ | |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 17 bit | 16 bit | 15 bit | 14 bit | 13 bit | 12 bit | 11 bit | 10 bit | 9 bit | 8 bit | 7 bit | 6 bit | 5 bit | 4 bit | 3 bit | 2 bit | 1 bit | 0 bit |
| Manchester encoded data | | | | | | | | | | | | | | | | | |

The buffer storing to "Backward Frame" by Manchester encoding requires "(number of response data bits $\times$ 2) + Start bit", variable of uint32_t (4 bytes) is secured.

Response data is fetched 1 bit at a time, and if the fetched bits are the following,

- When it is "0", "1" is stored in the upper bit, "0" is stored in the lower bit, and the falling edge can be output.

- When it is "1", "0" is stored in the upper bit, "1" is stored in the lower bit, and the rising edge can be output.

The transmission data performed Manchester encoding of are taken out of higher bit and are set to an output port in a timing of Half bit time defined by Transmitter timing.

### 4.4.5 **Transmitter bit timing definition**

The Transmitter bit timing used within this driver is described below.
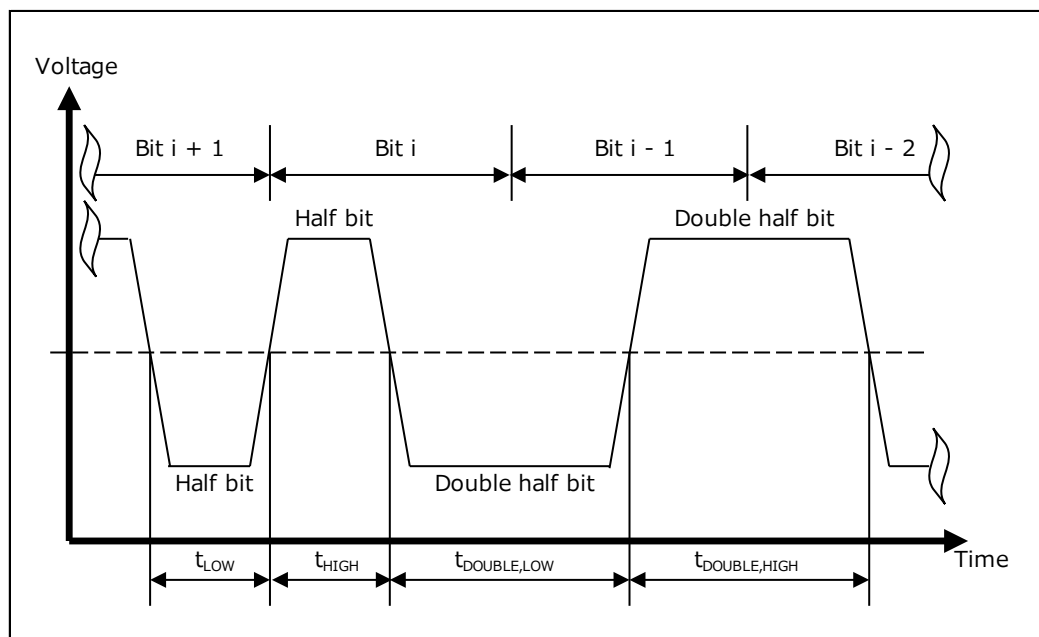


**Figure 4-14 Transmitter bit timing example**

**Table 4-18 Transmitter bit timing**

|  | Minimum | Typical | Maximum |
|---|---|---|---|
| Half bit time $t_{HIGH}$, $t_{LOW}$ | 366.7us | 416.7us | 466.7us |
| Double half bit time $t_{DOUBLE,LOW}$, $t_{DOUBLE,HIGH}$ | 733.3us | 833.3us | 933.3us |
| Stop condition time $T_{STOP}$ | 2,450us |  |  |

Set the interval timer to 417 us as an approximate value of the Half bit time Typical value shown in Table 4-18, and set the output terminal for each Half bit time to transmit the backward frame.

### 4.4.6 **Transmitter bit timing generation**

The output method of Backward Frame is shown below.

Figure 4-15 shows the operation of the timer to be used, the timing to output to the pin, and the waveform that flows on the DALI bus. The blue arrow in the figure indicates the operation of the program (CPU). By setting the transmit data to the P10 pin, the same waveform is output on the DALI Line.
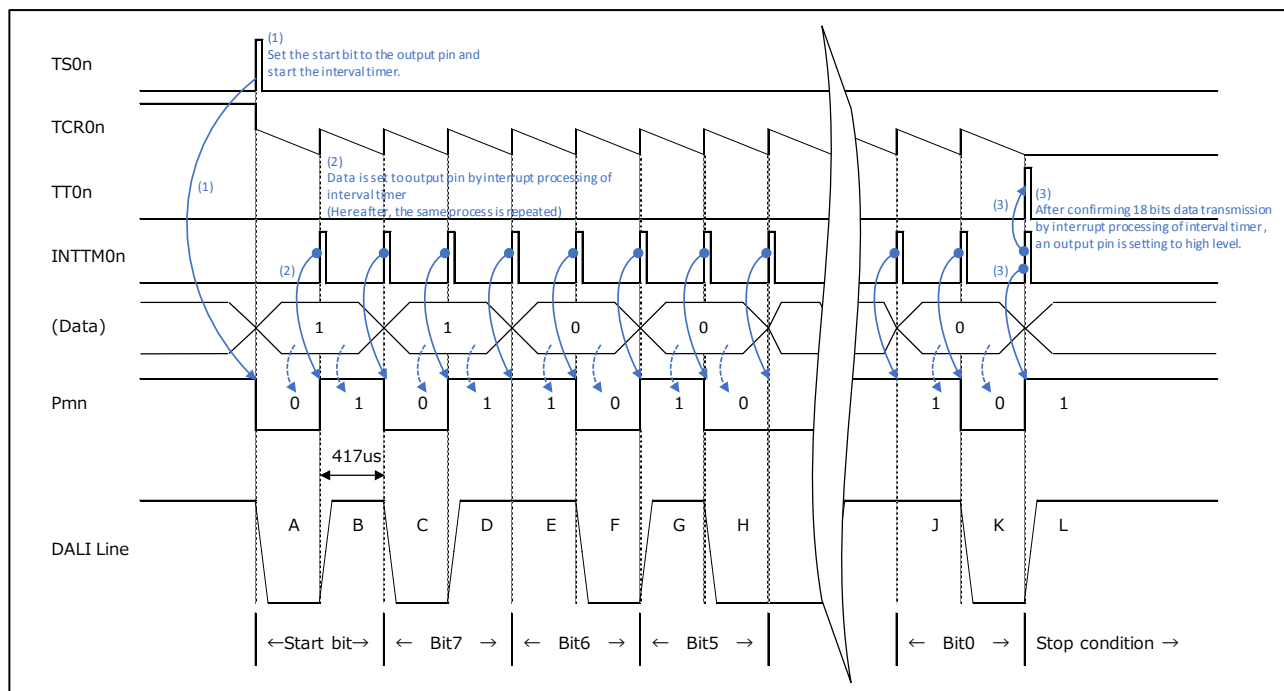


**Figure 4-15 Example of Backward Frame transmission timing**

(1)  When receiving a data transmission request, set the Half bit of the Start bit to the I/O pin and start the interval timer of the Transmitter bit timing. ("A" in "Figure 4-15 Example of Backward Frame transmission timing")

(2)  After that, set the data stored in the transmit data buffer at the interrupt timing of the interval timer to the output terminal in order from the most significant bit. ("B" to "K" in "Figure 4-15 Example of Backward Frame transmission timing")

(3)  At the end of all output, stop the interval timer of Transmitter bit timing, set the high level signal to the output terminal, and output the stop condition. ("L" in "Figure 4-15 Example of Backward Frame transmission timing")

Note

Delay may occur if the processing of another interrupt takes a lot of time.

If the Half bit interrupt is delayed, the Backward Frame is judged to be damaged, so set the interrupt level so that it is recognized as a Backward Frame.

### 4.4.7 **Corrupted Backward Frame generation**

The following describes how to generate a Corrupted Backward Frame of IEC62386-101 ed.2.0.

If it has multiple logical units and even one of the backward frames is different, it needs to send a Corrupted Backward Frame containing Active State (low level signal) of at least 1300 μs and a maximum of 2000 μs. (For details, refer to chapter 9.5.2 of the DALI standard document "IEC 62386-101 Edition 2.0".)

In this driver, 0b010101010101010000 is transmitted as Corrupted Backward Frame as shown inFigure 4-16.
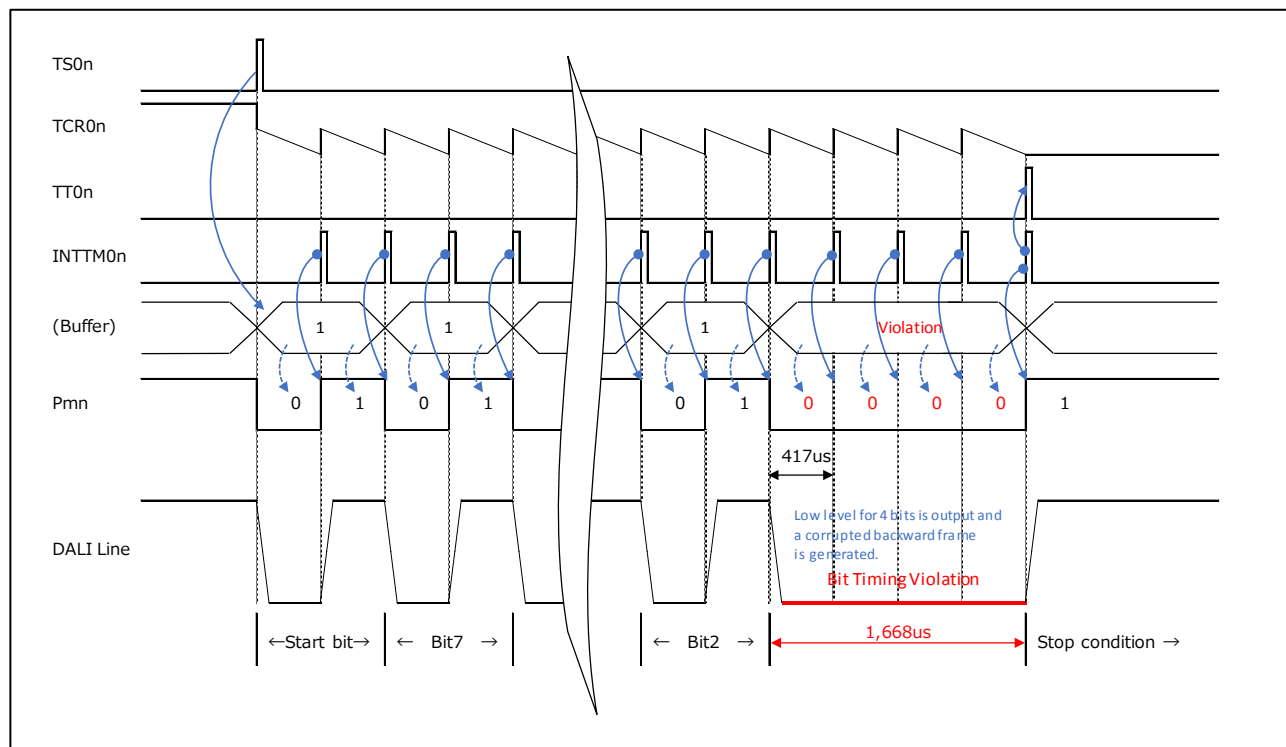


**Figure 4-16 Corrupted Backward Frame**

Note    In this sample program, since it is composed of one logical unit, Corrupted Backward Frame will never occur. It corresponds to multiple logical units and transmits when the response differs.

## 4.5 Dimming driver

The method of outputting the dimming signal (PWM) from the set Actual Level is shown below.

### 4.5.1 Function list

The following is a list of functions used in the dimming driver.

| Function name | Description |
|---|---|
| Lamp_SetLevel | Dimming Level setting function |
| Lamp_IsOn | Lighting confirmation function |

### 4.5.2 Function specification

The specifications of each function are shown below.

| Format | void Lamp_SetLevel (uint8_t ch, uint8_t level) | | |
|---|---|---|---|
| Parameters | I/O | Data type | Summary |
| ch | I | uint8_t | Channel number |
| level | I | uint8_t | Dimming Level |
| Return value | | Data type | Summary |
| | | void | - |
| Summary | Setting of dimming level | | |
| Details of processing | Acquires the count value of the PWM defined in the DUTY MAP from the dimming level and sets it in the timer register. | | |
| Notice | - | | |

| Format | bool Lamp_IsOn (uint8_t ch) | | |
|---|---|---|---|
| Parameters | I/O | Data type | Summary |
| ch | I | uint8_t | Channel number |
| Return value | | Data type | Summary |
| | | bool | Current lighting state（true:ON / false:OFF） |
| Summary | Return lighting status | | |
| Details of processing | Returns the lighting status of the specified channel. | | |
| Notice | startup is not taken into account. | | |

### 4.5.3    Dimming signal driver

With this driver, dimming signal (PWM) is output without directly adjusting the lighting of the lamp.

As for the dimming signal (PWM), control the duty ratio directly as the dimming ratio as follows.

**Table 4-19 Correspondence between duty ratio and dimming ratio**

| Duty ratio | Dimming ratio |
|---|---|
| 0% | 0% (turn off) |
| 0.1〜100% | Dimming at 0.1 to 100% (according to the specifications) |

### 4.5.4    Duty value calculation method

The relationship between Actual Level (Level), Dimming ratio (Light Output), and PWM control register setting (Count Value) is shown in DUTY MAP.

DUTY MAP is based on the following.

— PWM cycle: 1ms(1kHz)
— A maximum count level: 32000 (High-speed on-chip oscillator: 32MHz)
— DALI standard book (IEC 62386-102 Edition2.0 Chapter 9.3 Table-3 Dimming curve)

The calculation method of Light Output when Actual Level is 1 to 254 is shown below.

$$\text{Light Output} = 10^{\frac{actualLevel-1}{253/3}-1}(\%)$$

Calculation method of Count Value from Light Output is shown below.

$$\text{Count Value} = 32000 \times \frac{\text{Light Output}}{100}$$

**Table 4-20 DUTY MAP**

| Level (bit) | Light output (%) | Count value | Level (bit) | Light output (%) | Count value | Level (bit) | Light output (%) | Count value | Level (bit) | Light output (%) | Count value | Level (bit) | Light output (%) | Count value |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.100 | 32 | 52 | 0.402 | 129 | 103 | 1.620 | 518 | 154 | 6.520 | 2086 | 205 | 26.241 | 8397 |
| 2 | 0.103 | 33 | 53 | 0.414 | 132 | 104 | 1.665 | 533 | 155 | 6.700 | 2144 | 206 | 26.967 | 8629 |
| 3 | 0.106 | 34 | 54 | 0.425 | 136 | 105 | 1.711 | 548 | 156 | 6.886 | 2204 | 207 | 27.713 | 8868 |
| 4 | 0.109 | 35 | 55 | 0.437 | 140 | 106 | 1.758 | 563 | 157 | 7.076 | 2264 | 208 | 28.480 | 9114 |
| 5 | 0.112 | 36 | 56 | 0.449 | 144 | 107 | 1.807 | 578 | 158 | 7.272 | 2327 | 209 | 29.269 | 9366 |
| 6 | 0.115 | 37 | 57 | 0.461 | 148 | 108 | 1.857 | 594 | 159 | 7.473 | 2391 | 210 | 30.079 | 9625 |
| 7 | 0.118 | 38 | 58 | 0.474 | 152 | 109 | 1.908 | 611 | 160 | 7.680 | 2458 | 211 | 30.911 | 9892 |
| 8 | 0.121 | 39 | 59 | 0.487 | 156 | 110 | 1.961 | 628 | 161 | 7.893 | 2526 | 212 | 31.767 | 10165 |
| 9 | 0.124 | 40 | 60 | 0.501 | 160 | 111 | 2.015 | 645 | 162 | 8.111 | 2596 | 213 | 32.646 | 10447 |
| 10 | 0.128 | 41 | 61 | 0.515 | 165 | 112 | 2.071 | 663 | 163 | 8.336 | 2668 | 214 | 33.550 | 10736 |
| 11 | 0.131 | 42 | 62 | 0.529 | 169 | 113 | 2.128 | 681 | 164 | 8.567 | 2741 | 215 | 34.479 | 11033 |
| 12 | 0.135 | 43 | 63 | 0.543 | 174 | 114 | 2.187 | 700 | 165 | 8.804 | 2817 | 216 | 35.433 | 11339 |
| 13 | 0.139 | 44 | 64 | 0.559 | 179 | 115 | 2.248 | 719 | 166 | 9.047 | 2895 | 217 | 36.414 | 11652 |
| 14 | 0.143 | 46 | 65 | 0.574 | 184 | 116 | 2.310 | 739 | 167 | 9.298 | 2975 | 218 | 37.422 | 11975 |
| 15 | 0.147 | 47 | 66 | 0.590 | 189 | 117 | 2.374 | 760 | 168 | 9.555 | 3058 | 219 | 38.457 | 12306 |
| 16 | 0.151 | 48 | 67 | 0.606 | 194 | 118 | 2.440 | 781 | 169 | 9.820 | 3142 | 220 | 39.522 | 12647 |
| 17 | 0.155 | 50 | 68 | 0.623 | 199 | 119 | 2.507 | 802 | 170 | 10.091 | 3229 | 221 | 40.616 | 12997 |
| 18 | 0.159 | 51 | 69 | 0.640 | 205 | 120 | 2.577 | 825 | 171 | 10.371 | 3319 | 222 | 41.740 | 13357 |
| 19 | 0.163 | 52 | 70 | 0.658 | 211 | 121 | 2.648 | 847 | 172 | 10.658 | 3411 | 223 | 42.895 | 13726 |
| 20 | 0.168 | 54 | 71 | 0.676 | 216 | 122 | 2.721 | 871 | 173 | 10.953 | 3505 | 224 | 44.083 | 14107 |
| 21 | 0.173 | 55 | 72 | 0.695 | 222 | 123 | 2.797 | 895 | 174 | 11.256 | 3602 | 225 | 45.303 | 14497 |
| 22 | 0.177 | 57 | 73 | 0.714 | 228 | 124 | 2.874 | 920 | 175 | 11.568 | 3702 | 226 | 46.557 | 14898 |
| 23 | 0.182 | 58 | 74 | 0.734 | 235 | 125 | 2.954 | 945 | 176 | 11.888 | 3804 | 227 | 47.846 | 15311 |
| 24 | 0.187 | 60 | 75 | 0.754 | 241 | 126 | 3.035 | 971 | 177 | 12.217 | 3909 | 228 | 49.170 | 15734 |
| 25 | 0.193 | 62 | 76 | 0.775 | 248 | 127 | 3.119 | 998 | 178 | 12.555 | 4018 | 229 | 50.531 | 16170 |
| 26 | 0.198 | 63 | 77 | 0.796 | 255 | 128 | 3.206 | 1026 | 179 | 12.902 | 4129 | 230 | 51.930 | 16618 |
| 27 | 0.203 | 65 | 78 | 0.819 | 262 | 129 | 3.294 | 1054 | 180 | 13.260 | 4243 | 231 | 53.367 | 17077 |
| 28 | 0.209 | 67 | 79 | 0.841 | 269 | 130 | 3.386 | 1084 | 181 | 13.627 | 4361 | 232 | 54.844 | 17550 |
| 29 | 0.215 | 69 | 80 | 0.864 | 276 | 131 | 3.479 | 1113 | 182 | 14.004 | 4481 | 233 | 56.362 | 18036 |
| 30 | 0.221 | 71 | 81 | 0.888 | 284 | 132 | 3.576 | 1144 | 183 | 14.391 | 4605 | 234 | 57.922 | 18535 |
| 31 | 0.227 | 73 | 82 | 0.913 | 292 | 133 | 3.675 | 1176 | 184 | 14.790 | 4733 | 235 | 59.526 | 19048 |
| 32 | 0.233 | 75 | 83 | 0.938 | 300 | 134 | 3.776 | 1208 | 185 | 15.199 | 4864 | 236 | 61.173 | 19575 |
| 33 | 0.240 | 77 | 84 | 0.964 | 308 | 135 | 3.881 | 1242 | 186 | 15.620 | 4998 | 237 | 62.866 | 20117 |
| 34 | 0.246 | 79 | 85 | 0.991 | 317 | 136 | 3.988 | 1276 | 187 | 16.052 | 5137 | 238 | 64.607 | 20674 |
| 35 | 0.253 | 81 | 86 | 1.018 | 326 | 137 | 4.099 | 1312 | 188 | 16.496 | 5279 | 239 | 66.395 | 21246 |
| 36 | 0.260 | 83 | 87 | 1.047 | 335 | 138 | 4.212 | 1348 | 189 | 16.953 | 5425 | 240 | 68.233 | 21835 |
| 37 | 0.267 | 85 | 88 | 1.076 | 344 | 139 | 4.329 | 1385 | 190 | 17.422 | 5575 | 241 | 70.121 | 22439 |
| 38 | 0.275 | 88 | 89 | 1.105 | 354 | 140 | 4.449 | 1424 | 191 | 17.905 | 5730 | 242 | 72.062 | 23060 |
| 39 | 0.282 | 90 | 90 | 1.136 | 364 | 141 | 4.572 | 1463 | 192 | 18.400 | 5888 | 243 | 74.057 | 23698 |
| 40 | 0.290 | 93 | 91 | 1.167 | 373 | 142 | 4.698 | 1503 | 193 | 18.909 | 6051 | 244 | 76.107 | 24354 |
| 41 | 0.298 | 95 | 92 | 1.200 | 384 | 143 | 4.828 | 1545 | 194 | 19.433 | 6219 | 245 | 78.213 | 25028 |
| 42 | 0.306 | 98 | 93 | 1.233 | 395 | 144 | 4.962 | 1588 | 195 | 19.971 | 6391 | 246 | 80.378 | 25721 |
| 43 | 0.315 | 101 | 94 | 1.267 | 405 | 145 | 5.099 | 1632 | 196 | 20.524 | 6568 | 247 | 82.603 | 26433 |
| 44 | 0.324 | 104 | 95 | 1.302 | 417 | 146 | 5.240 | 1677 | 197 | 21.092 | 6749 | 248 | 84.889 | 27164 |
| 45 | 0.332 | 106 | 96 | 1.338 | 428 | 147 | 5.385 | 1723 | 198 | 21.675 | 6936 | 249 | 87.239 | 27916 |
| 46 | 0.342 | 109 | 97 | 1.375 | 440 | 148 | 5.535 | 1771 | 199 | 22.275 | 7128 | 250 | 89.654 | 28689 |
| 47 | 0.351 | 112 | 98 | 1.413 | 452 | 149 | 5.688 | 1820 | 200 | 22.892 | 7325 | 251 | 92.135 | 29483 |
| 48 | 0.361 | 116 | 99 | 1.452 | 465 | 150 | 5.845 | 1870 | 201 | 23.526 | 7528 | 252 | 94.686 | 30300 |
| 49 | 0.371 | 119 | 100 | 1.492 | 477 | 151 | 6.007 | 1922 | 202 | 24.177 | 7737 | 253 | 97.307 | 31138 |
| 50 | 0.381 | 122 | 101 | 1.534 | 491 | 152 | 6.173 | 1975 | 203 | 24.846 | 7951 | 254 | 100.000 | 32000 |
| 51 | 0.392 | 125 | 102 | 1.576 | 504 | 153 | 6.344 | 2030 | 204 | 25.534 | 8171 | | | |

## 5. **References**

- RL78/G13 User's Manual: Hardware (R01UH0146)

- RL78/G13 Target board QB-R5F100LE-TB User's Manual (R20UT0624XJ0200)

- RL78/I1A Family User's Manual DALI Library (IEC62386-102ed2.0)

- EEPROM Emulation Library Pack02 Package Ver.2.00(for CA78K0R/CC-RL Compiler) for RL78 Family (R20UT2645)

- Digital addressable lighting interface – Part 101: General requirements – System components (IEC 62386-101 Edition2.0)

- Digital addressable lighting interface – Part 102: General requirements – Control gear (IEC 62386-102 Edition2.0)

## Website and Support

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

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

All trademarks and registered trademarks are the property of their respective owners.

## Revision History

| Rev. | Date | Description | |
| --- | --- | --- | --- |
| | | **Page** | **Summary** |
| 1.01 | 26th Sep., 2019 | All | First edition issued |
| | | | |

# General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Handling of Unused Pins

   Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.

   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment when power is supplied.

   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products

   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

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