# RL78 Family

## MIDI Performance Control Sample Software Using SIS

### Introduction

This application note provides examples of using communication control with MIDI devices by using the MIDI interface SIS (Software Integration System) module.

The LED matrix display is controlled in accordance with NoteOn messages (MIDI messages) generated by hexadecimal keyboard input. MIDI messages can also be transferred to the sound module to play sounds by using the MIDI interface SIS module.

The compliant standard is as follows.

- MIDI 1.0

For details on the MIDI standard, refer to the preceding specification.

### Target Devices

RL78/G16

When applying the sample program covered in this application note to another microcomputer, modify the program according to the specifications for the target microcomputer and conduct an extensive evaluation of the modified program.

Contents

1. Specification ....4
1.1 Overview of Specification ....4
1.1.1 Communication specifications ....5
1.2 Operation Details ....8

2. Hardware Description ....13
2.1 Hardware Configuration ....13
2.2 Pin Connection Diagrams ....15
2.3 List of Used Pins ....17

3. Software Description ....18
3.1 Software Environment ....18
3.2 Peripheral Function Settings ....18
3.3 Setting of Option Byte ....23
3.4 List of Macros ....24
3.5 List of Constants ....26
3.6 List of Variables ....27
3.7 List of Functions ....30
3.8 Function Specifications ....31
3.9 Flowcharts ....36
3.9.1 Main processing ....36
3.9.2 Volume setting update processing ....39
3.9.3 Processing to acquire volume switch input ....40
3.9.4 LED matrix display processing ....41
3.9.5 System timer acquisition processing ....42
3.9.6 System timer count processing ....42
3.9.7 LED matrix data transmission processing ....43
3.9.8 Processing to check the end of LED matrix data transmission ....44
3.9.9 TAU0_3 interrupt processing ....44
3.9.10 AD conversion end interrupt processing ....45
3.9.11 LED matrix data CSI transmission end processing ....45
3.9.12 UART0 transmission end processing ....46
3.9.13 UART0 reception end processing ....46
3.9.14 Hexadecimal keyboard key input processing ....47
3.9.15 Input processing for one column of the hexadecimal keyboard ....48
3.9.16 Key input determination processing ....49
3.9.17 Count processing for the key input confirmation timer ....50
3.9.18 Processing to start the key input confirmation timer ....51
3.9.19 Processing to acquire the key input confirmation timer count ....51

4. Sample Code ....52

5. Notes ....52
5.1 Operation of the Hexadecimal Keyboard ....52

6. Reference Documents ..... 52

Revision History ..... 54

# 1. Specification

## 1.1 Overview of Specification

This sample program provides examples of using the MIDI message output function by using the MIDI interface SIS (Software Integration System) module.

The LED matrix display is controlled in accordance with NoteOn messages (MIDI messages) generated by hexadecimal keyboard input. MIDI messages can be transferred from the MIDIOUT pin to the sound module to play sounds by using the transmission function.

Table 1-1 lists the peripheral functions for use and their application. Figure 1-1 shows an overview of the sample program operation.

Table 1-1 Peripheral Functions for Use and Their Application

| Peripheral Function | Application |
|---|---|
| Serial interface UART0<br>P03/TxD0 | UART communication with MIDI devices |
| A/D converter<br>P05/ANI4 | Volume switch input for volume selection |
| Serial interface CSI20<br>P13/SCK20, P16/GPIO, P15/SO20 | SPI communication with the LED matrix module |
| P10/GPIO, P21/GPIO, P60/GPIO, P61/GPIO,<br>P11/GPIO, P43/GPIO, P137/GPIO, P12/GPIO | Key matrix input to the hexadecimal keyboard |

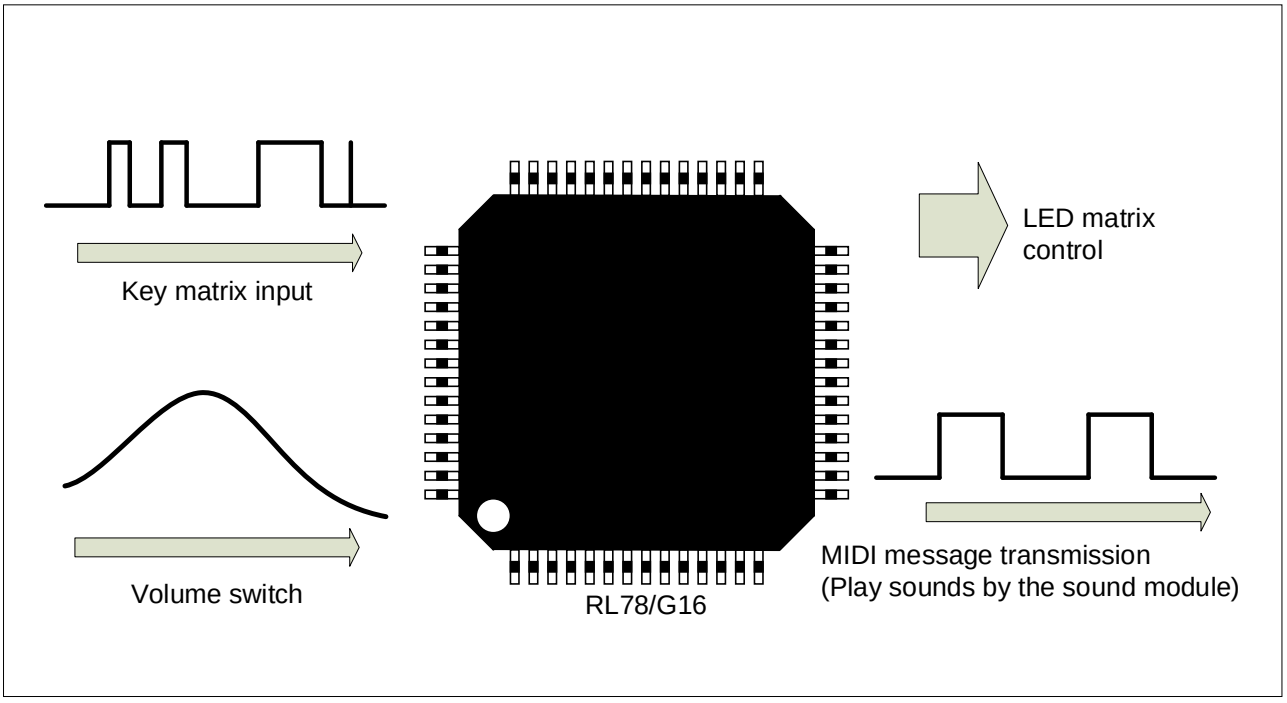



Figure 1-1 Overview of Sample Program Operation

### 1.1.1 Communication specifications

The following describes the MIDI standard data configuration used in this sample program.

As shown in Figure 1-2, the MIDI data column is a bit column for unidirectional asynchronous communication of 31.25 Kb/sec. Each byte to be sent consists of ten bits (one start bit, eight data bits, and one stop bit).

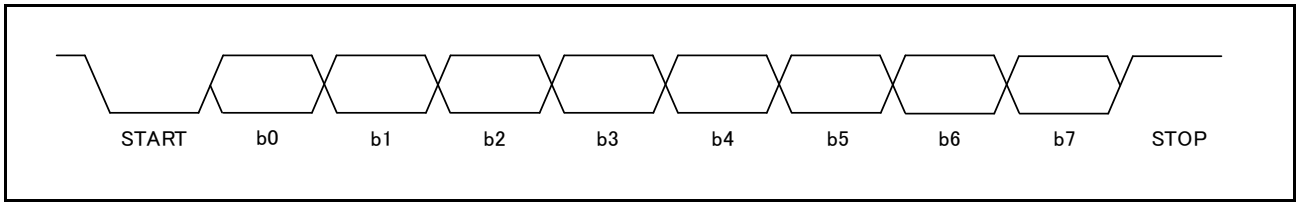



Figure 1-2 MIDI Data Column

As shown in Figure 1-3, a MIDI message consists of a status byte and data bytes, and is roughly categorized as a channel message or a system message according to the status byte.

Status byte

Channel Message Structure

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| 1 | Type of msg | | | Channel# | | | |

System Message Structure

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | Type of msg | | | |

Data byte

| b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
|---|---|---|---|---|---|---|---|
| 0 | Data | | | | | | |

Figure 1-3 MIDI Data Structure

The following describes the NoteOn (keystroke) message, which is a channel message used by this sample program to control the LED matrix.

As shown in Figure 1-4, the NoteOn (keystroke) message consists of the status byte followed by two data bytes.

The status byte contains the channel number (Channel#).

Data byte 1 contains the note number (Note#), indicating the note.

Data byte 2 contains “Velocity”, indicating the velocity of the sound.

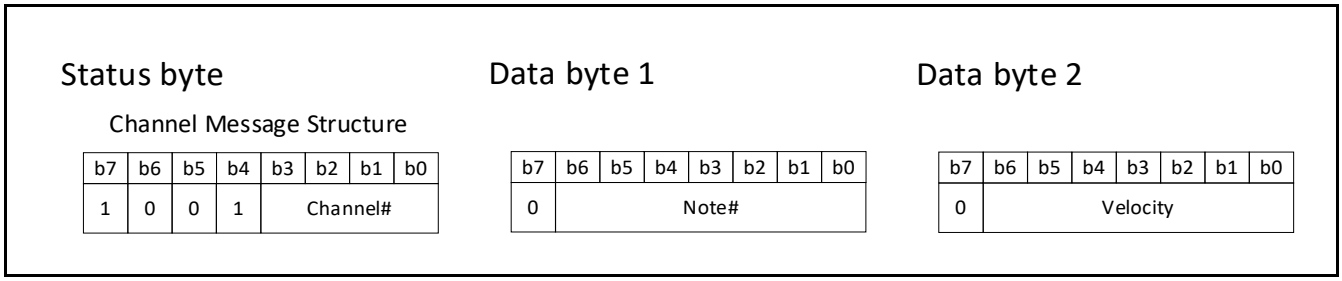



Figure 1-4 NoteOn (Keystoke) Message

The following describes the program change message used to specify the tone for a MIDI channel.

As shown in Figure 1-5, the message consists of a status byte followed by one data byte.

The status byte contains the channel number (Channel#).

Data byte 1 contains the program number (Prog#), indicating the tone.

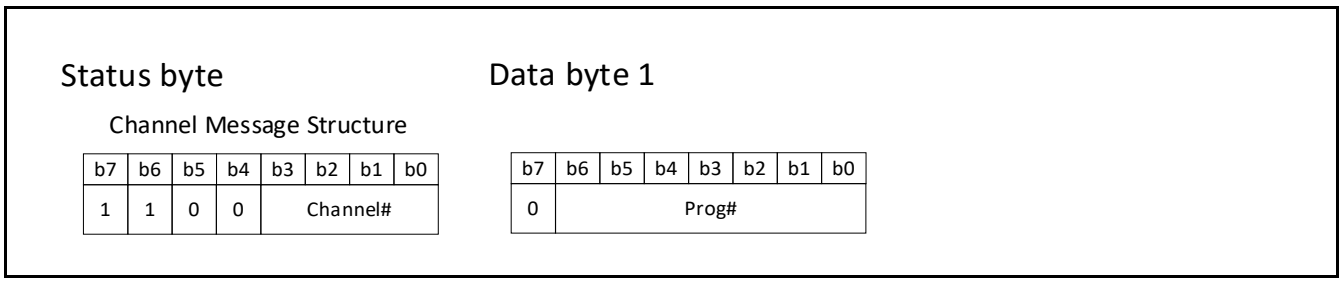



Figure 1-5 Program Change Message

The following describes the control change message used to expand the number of tones for a MIDI channel.

The bank settings for MSB and LSB provide 16,384 choices for a program number specified in the program change message.

As shown in Figure 1-6, the message consists of a status byte followed by two data bytes.

The status byte contains the channel number (Channel#).

Data byte 1 contains the controller number (Controller#), indicating the target bank.

The controller number must be set to 0 for Bank Select MSB and 32 for Bank Select LSB.

Data byte 2 contains the bank number (Bank#), indicating the bank set value.

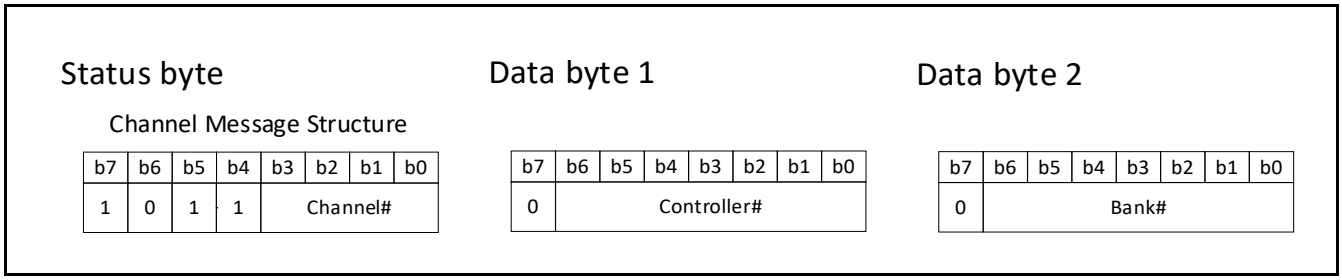



Figure 1-6 Control Change Message (Bank Select)

The following describes the control change message used to specify the volume for a MIDI channel.

As shown in Figure 1-7, the message consists of a status byte followed by two data bytes.

The status byte contains the channel number (Channel#).

Data byte 1 contains the controller number (Controller#), indicating the channel volume.

Specify 7 for the controller number.

Data byte 2 contains the volume (Volume#).

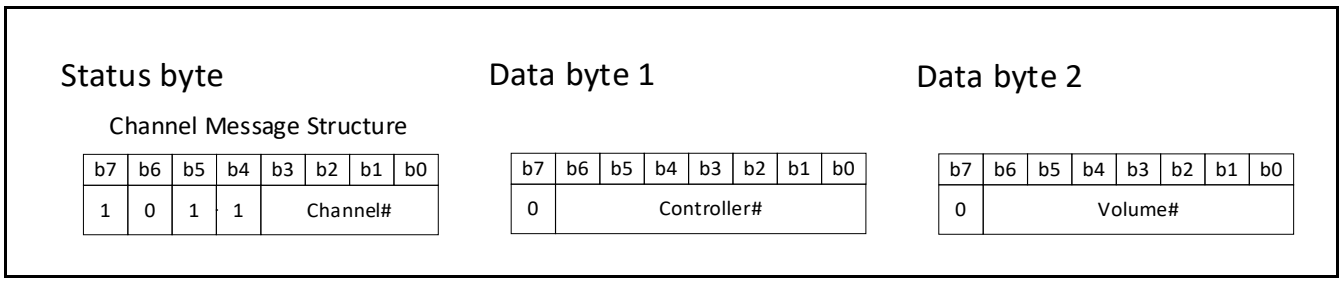



Figure 1-7 Control Change Message (Channel Volume)

## 1.2 Operation Details

This sample program assumes a hexadecimal keyboard as a piano keyboard and turns on the LED matrix according to the key information generated by key input. When a key is pressed, the sample program sends the NoteOn (keystroke) message of the note corresponding to the key to the sound module to play a sound. While the key is held down, the LED stays lit and sounding might continue depending on the instrument assigned to the MIDI channel. When the key is released, the LED is turned off and a NoteOff message is sent to stop the sound.

(1) Input on the hexadecimal keyboard

- The hexadecimal keyboard is a keypad having 16 keys (0 to F). The keys are arranged in a matrix with 4 rows and 4 columns. Key pressing status is read in units of columns.
- This sample program is placed in melody mode immediately after a reset. In this mode, as shown in Figure 1-8, notes are assigned to 0 to 9, F, and E keys as keyboard keys. Pressing a key turns on the LED matrix and plays a sound from the MIDI sound source.
- Pressing the “A” key changes the mode between melody mode and MIDI channel change mode.
- Pressing the “C” or “D” key changes the pitch by one octave.

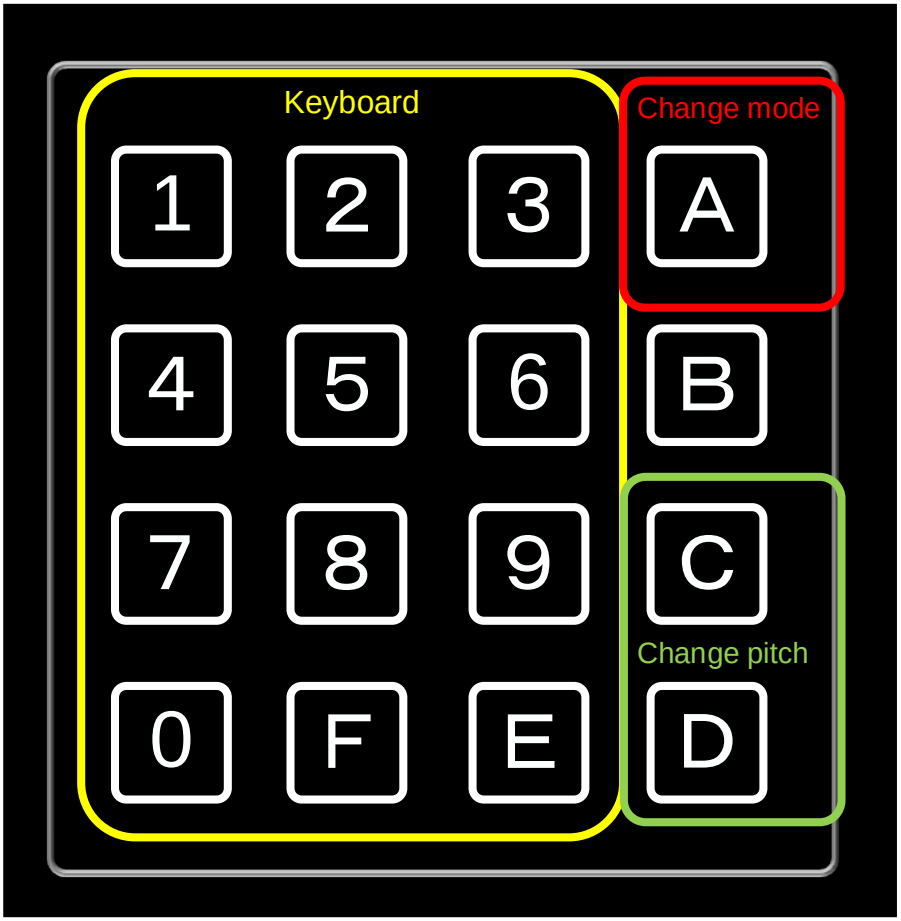



Figure 1-8 Assignment on Hexadecimal Keyboard

(2) Specifying and changing sound target MIDI channels

- The MIDI standard defines 1 to 16 channels (denoting “instruments”), which are referred to as MIDI channels.
- One of the tones that are preset for each MIDI channel can be selected to play sounds.
- MIDI channels are changed by using the hexadecimal keyboard in MIDI channel change mode.
- Pressing the A key in melody mode, as shown in Figure 1-9, causes transition to MIDI channel change mode.
- The current MIDI channel is displayed immediately after transition to MIDI channel change mode. This display also waits for input of a tens place digit of the MIDI channel number, where the valid input value is 0 or 1. When you enter a tens place digit, the display changes and waits for input of a ones place digit. In this state, 0 to 9 are valid input values for the MIDI channel. Enter a ones place digit so that the two digit number is within the range from 01 to 16.
- Pressing the “A” key in channel change mode returns to melody mode. If a two digit number is displayed at this time, that number is set for the MIDI channel. If the display still waits for input of a ones place digit, the current MIDI channel will be used without change.
- Figure 1-10 shows how the display target channels are displayed on the LED matrix. (Black is off. Red and green are lit.)

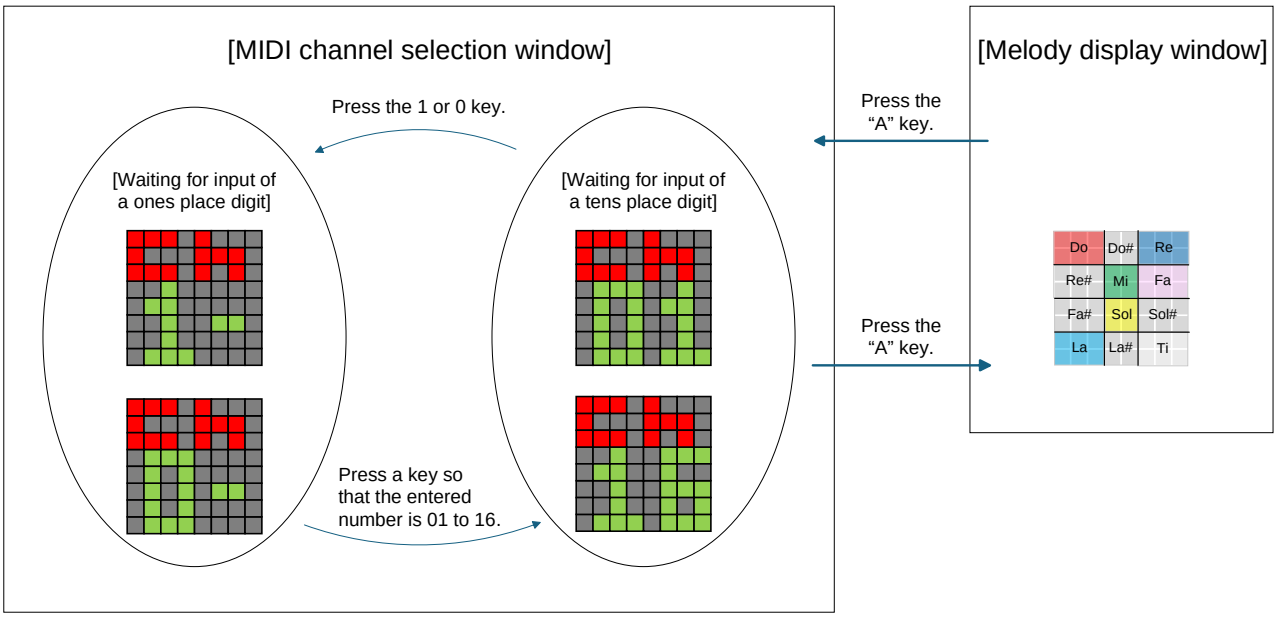



Figure 1-9 State Transition of Setting and Changing the Target MIDI Channel for Sound Output

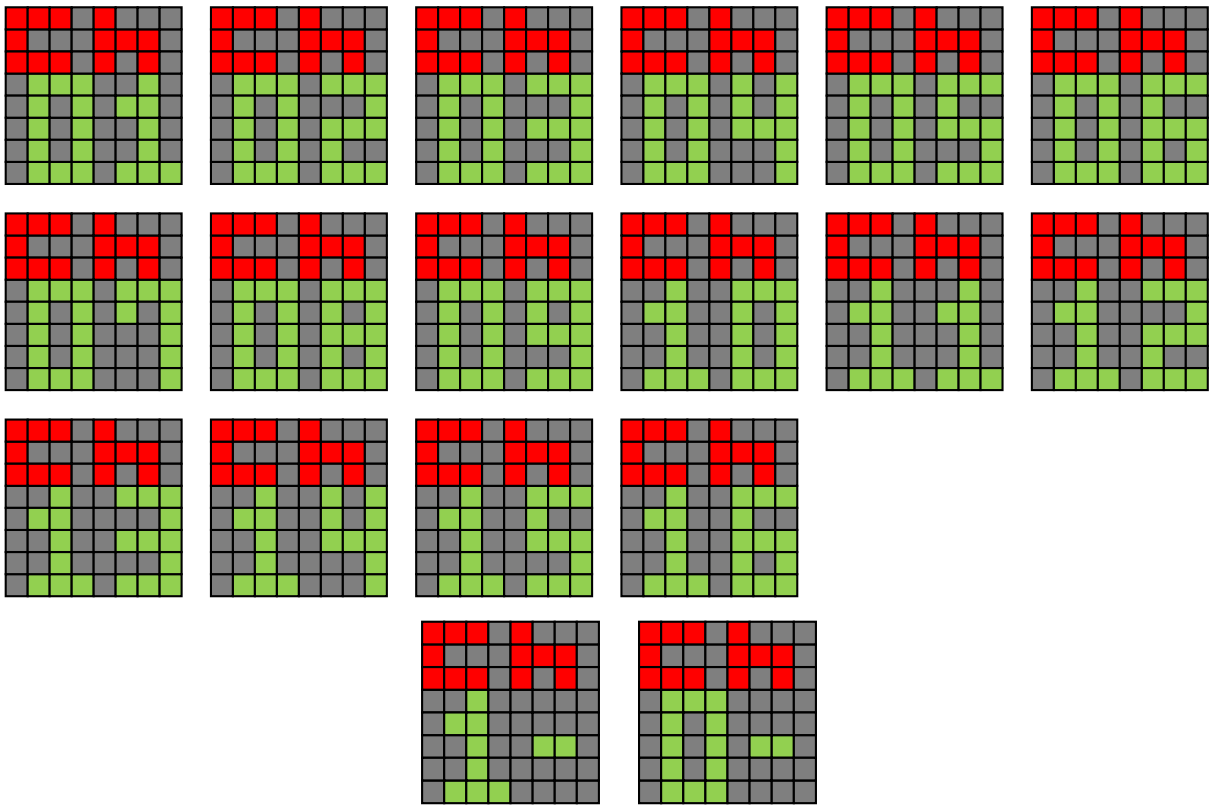


Figure 1-10 MIDI Channel Displays

(3) Specifying and changing the pitch

- The 4 × 4 hexadecimal keyboard can play sounds only in the range of one octave. However, the pitch can be changed in the range from -5 to +5 by one octave, assuming that the pitch is set to ±0 immediately after reset.
- The user can change the pitch by pressing the “C” or “D” key. (Change is possible even while sound is being played.)
- Pressing the “C” key increments the current pitch by 1, and pressing the “D” key decrements the current pitch by 1. In both cases, pitch information is displayed on the LED matrix for one second.
- Figure 1-11 shows how the pitch information is displayed on the LED matrix. (Black is off. Green is lit.)

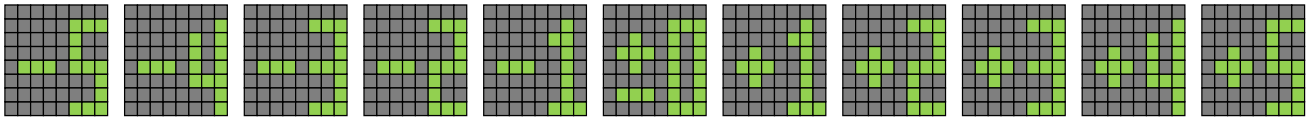


Figure 1-11 Pitch Information Display

(4) LED matrix display during melody display

- Assuming the hexadecimal keyboard as a piano keyboard, NoteOn (keystroke) messages are sent and the LED matrix is turned on according to the scale information based on the positions of pressed keys. That is, the LED matrix is turned on at the beginning of the sound.
- Figure 1-12 shows the correspondence between the hexadecimal keyboard and piano keyboard. Figure 1-13 shows the notes and LED matrix lighting positions.

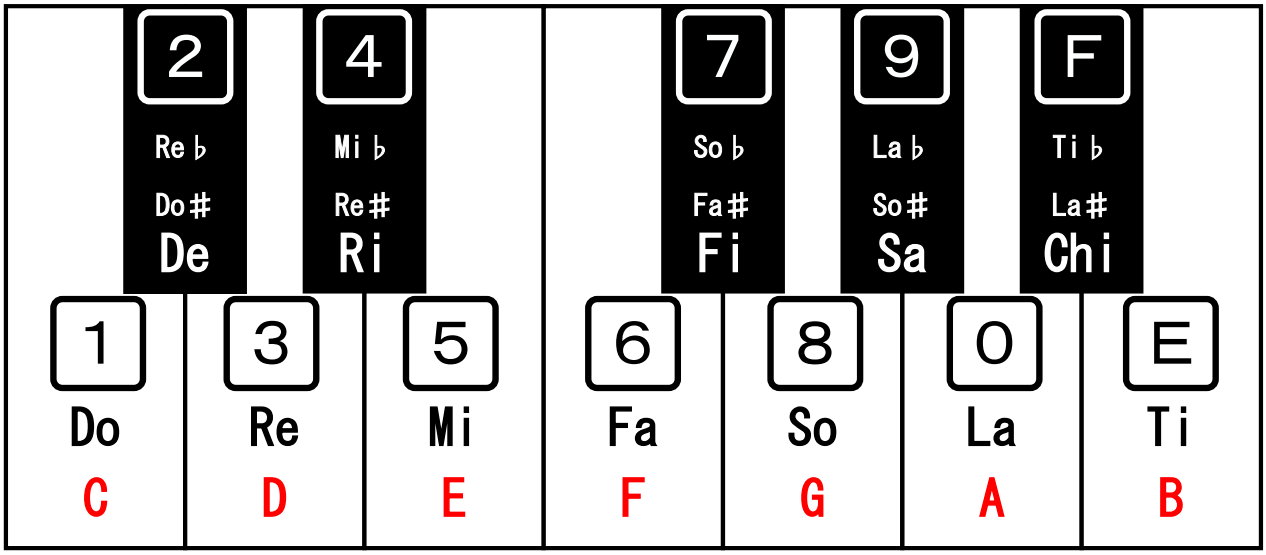



Figure 1-12 Correspondence Between the Hexadecimal Keyboard and Piano Keyboard

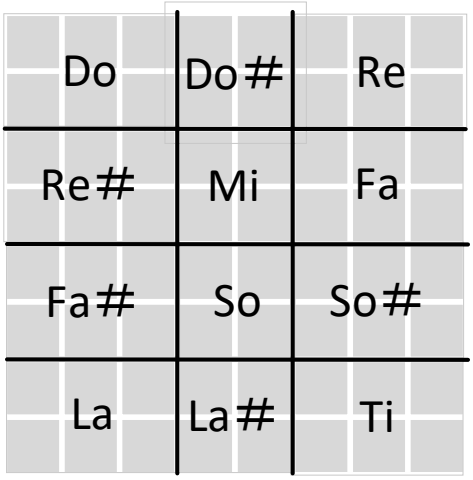



Figure 1-13 Correspondence Between the Notes and LED Matrix Lighting Positions

- Colors are assigned to each note (do, re, mi, fa, so, la, ti) as shown in Figure 1-14.As an example, for piano keyboard, the received keystroke messages are shown in a single color for the white keys and in two colors for the black keys.
- The LED stays lit while the key is held down. Also, sounding might continue depending on the instrument assigned to the MIDI channel.

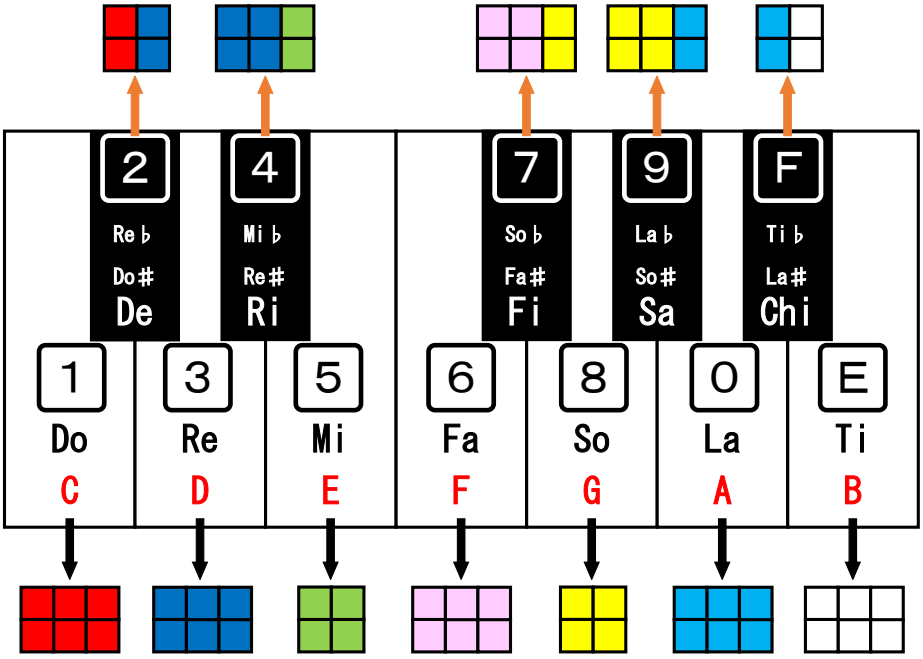



Figure 1-14 Display Color for Each Note

(5) Specifying and changing the volume

- Setting values in the range from 1 to 127, excluding 0 (mute), are converted to eight volume levels.
- The user can change the volume by turning Volume(A0). (Change is possible even while sound is being played.)
- When Volume(A0) is turned, volume information is displayed on the LED matrix for 1 second.
- The volume level is indicated by the width of the yellow band. The leftmost column is lit for minimum (level 1), and all eight columns in a row are lit for maximum (level 8).
- Figure 1-15 shows how the volume information is displayed on the LED matrix. (Black is off. Red, blue, green, and yellow are lit.)

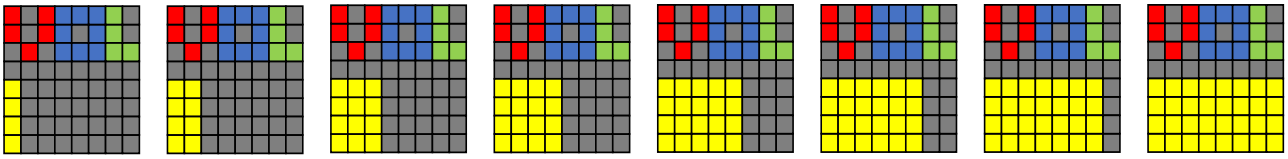


Figure 1-15 Volume Information Display

# 2. Hardware Description

## 2.1 Hardware Configuration

Table 2-1 describes the hardware used in this sample program.

Table 2-1 Hardware List

| Hardware | Application |
|---|---|
| Board used | Manufactured by Renesas Electronics<br>RL78/G16 Fast Prototyping Board (RTK5RLG160C00000BJ) |
| MCU used | RL78/G16 (R5F121BCAFP) |
| Operating frequency | High-speed on-chip oscillator clock (fHOCO): 16 MHz |
| Operating voltage | 5.0V |
| MIDI shield board | SparkFun MIDI Shield |
| MIDI to MIDI (male-to-male) cable | SANWA SUPPLY KB-MID01-18K |
| MIDI sound module | Roland SOUND Canvas SC-88 Pro |
| LED matrix module | 52pi EP-0075 RPI-RGB-LED-Matrix |
| Hexadecimal keyboard | digilent Pmod KYPD 16-button Keypad |

Figure 2-1 and Figure 2-2 show the configurations used in this application note.

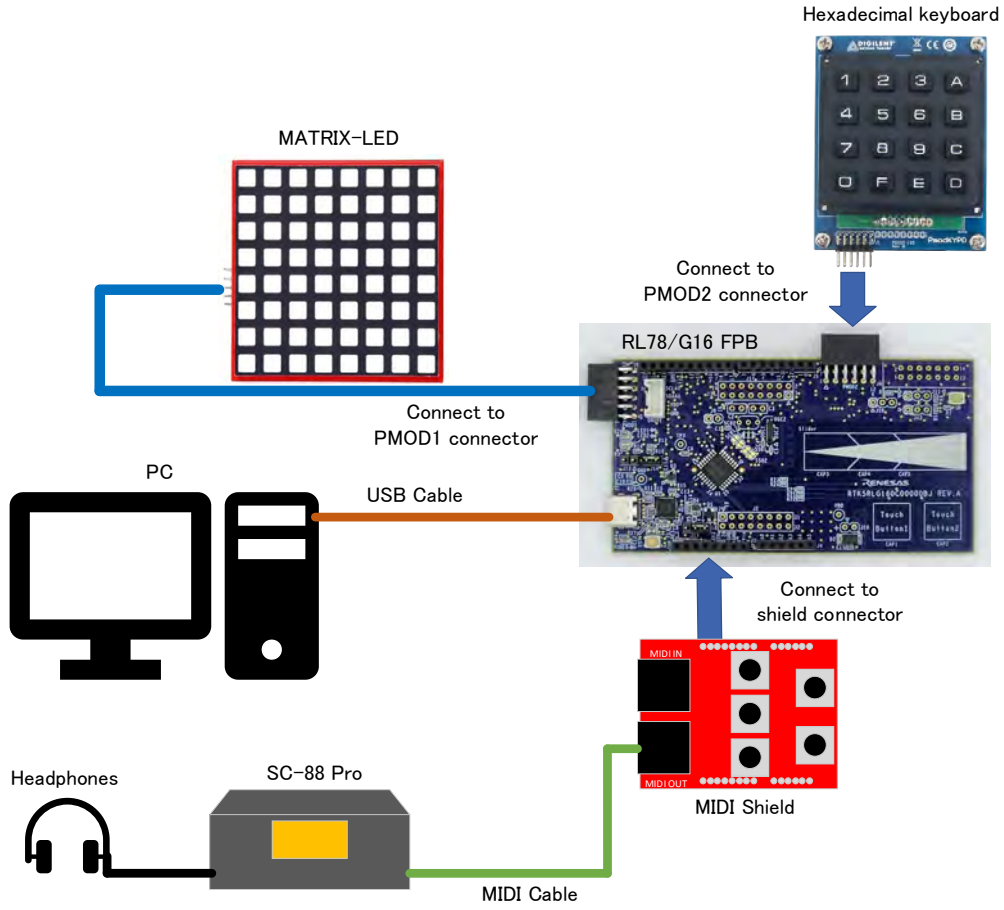



Figure 2-1 Hardware Configuration

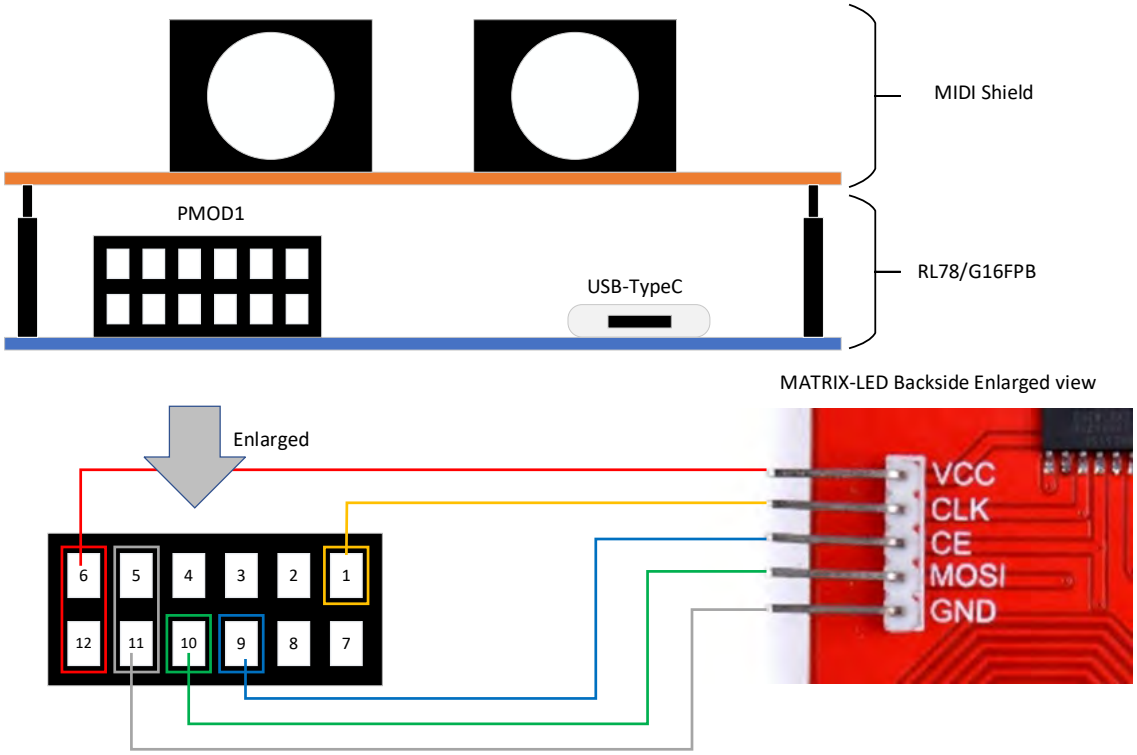



Figure 2-2 Wiring Between PMOD1 and MATRIX-LED

## 2.2 Pin Connection Diagrams

Figure 2-3 shows a pin connection diagram between the RL78/G16 FPB and the MIDI Shield. Figure 2-4 shows a pin connection diagram between the RL78/G16 FPB and the MATRIX-LED. Figure 2-5 shows a pin connection diagram between the RL78/G16 FPB and the hexadecimal keyboard.

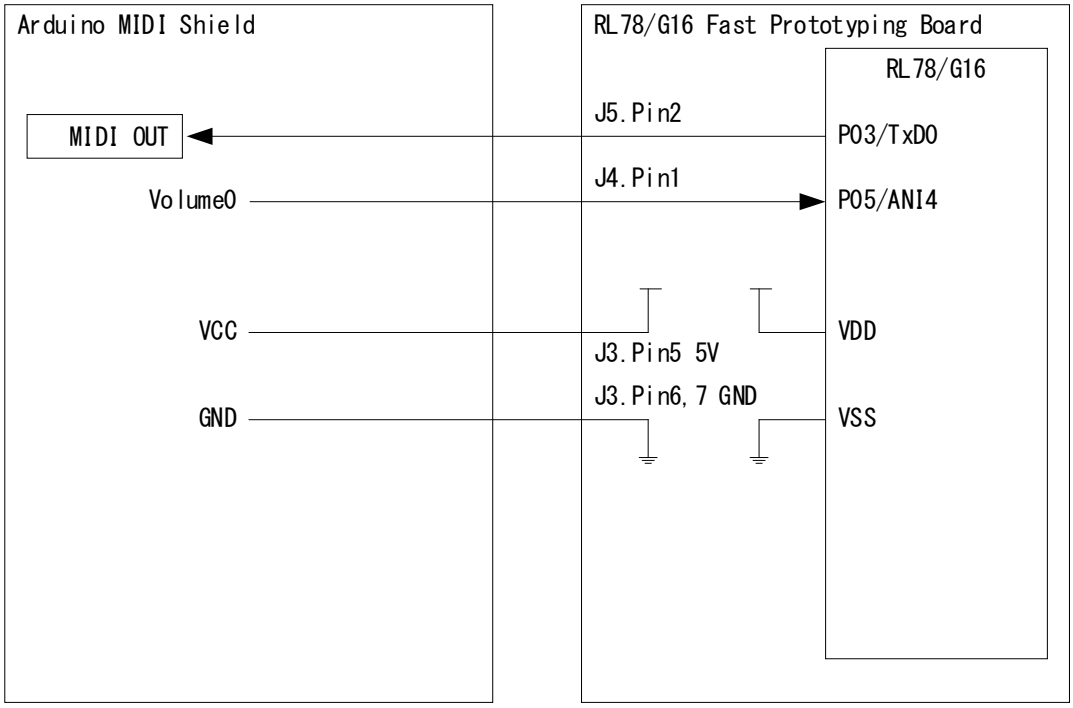



Figure 2-3 Pin Connection Diagram Between RL78/G16 FPB and MIDI Shield

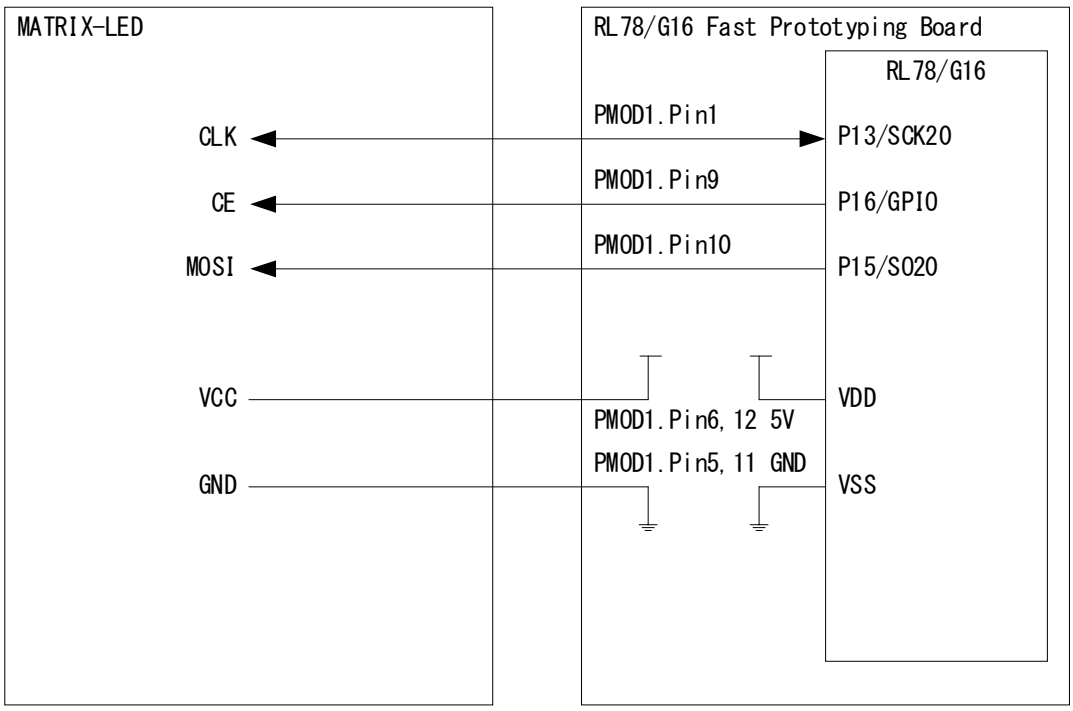



Figure 2-4 Pin Connection Diagram Between RL78/G16 FPB and MATRIX-LED

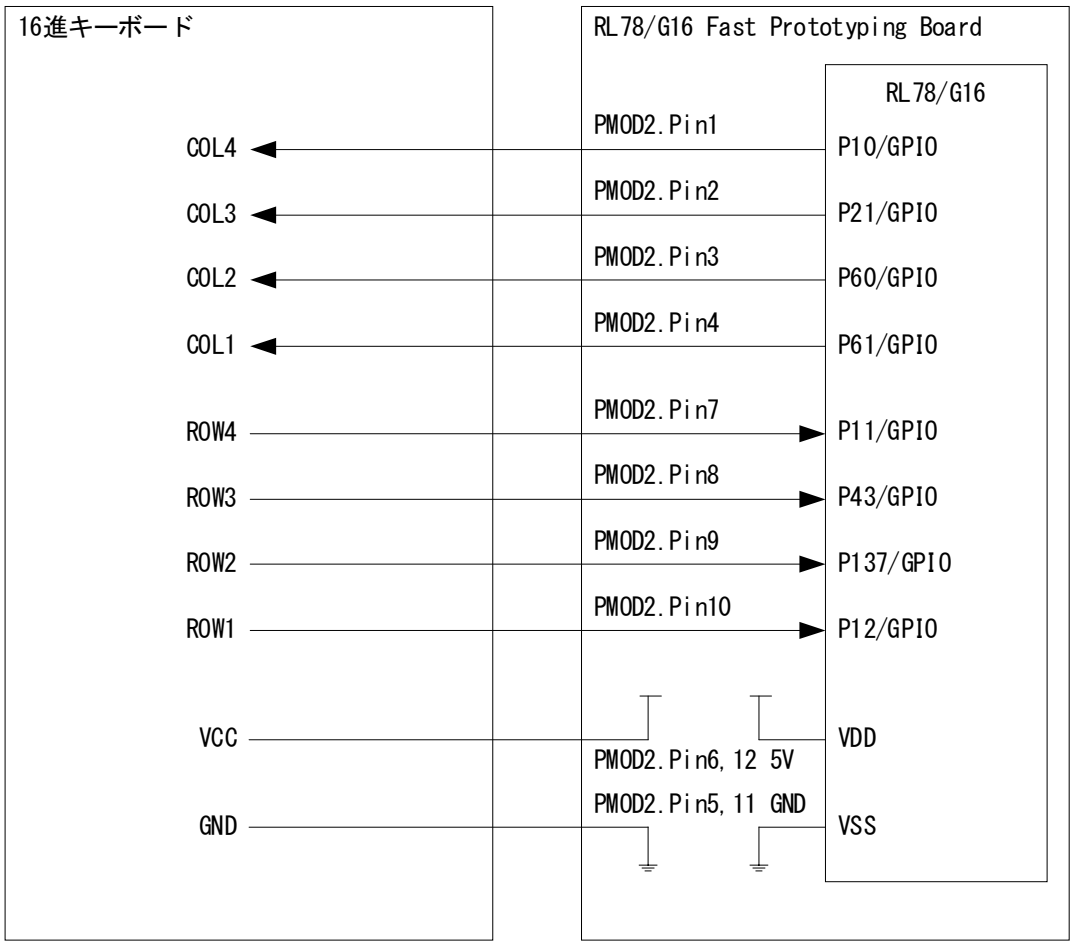



Figure 2-5 Pin Connection Diagram Between RL78/G16 FPB and Hexadecimal Keyboard

## 2.3 List of Used Pins

Table 2-2 lists the used pins and their functions.

Table 2-2 Used Pins and Functions

| Pin Name | I/O | Description |
|---|---|---|
| P03/TxD0 | Output | MIDI message transmission |
| P05/ANI4 | Input | Sound volume setting |
| P13/SCK20 | Output | MATRIX-LED SPI clock |
| P16/GPIO | Output | MATRIX-LED SPI chip selection |
| P15/SO20 | Output | MATRIX-LED SPI MOSI |
| P61/GPIO | Output | Hexadecimal keyboard COL1 |
| P60/GPIO | Output | Hexadecimal keyboard COL2 |
| P21/GPIO | Output | Hexadecimal keyboard COL3 |
| P10/GPIO | Output | Hexadecimal keyboard COL4 |
| P12/GPIO | Input | Hexadecimal keyboard ROW1 |
| P137/GPIO | Input | Hexadecimal keyboard ROW2 |
| P43/GPIO | Input | Hexadecimal keyboard ROW3 |
| P11/GPIO | Input | Hexadecimal keyboard ROW4 |

Note: Only the used pins are connected in this application note. When creating a circuit, refer to section 2.3 Connection of Unused Pins in the RL78/G16 User's Manual: Hardware (R01UH0980) and appropriately handle the pins not used in this application note so that the circuit design satisfies the electrical characteristics.

# 3. Software Description

## 3.1 Software Environment

Table 3-1 shows the software used in this sample program.

Table 3-1 Software

| Software | Application |
|---|---|
| Integrated development environment | Manufactured by Renesas Electronics<br>e$^2$ studio 2024-07 |
| C compiler | Manufactured by Renesas Electronics<br>C Compiler Package for RL78 Family [CC-RL] V1.14.00 |
| Smart Configurator (SC) | Smart Configurator for RL78 V1.10.0 |
| Board Support Package (BSP) | Manufactured by Renesas Electronics<br>V1.70 |

## 3.2 Peripheral Function Settings

Figure 3-1 shows the settings of the 1-ms interval timer.

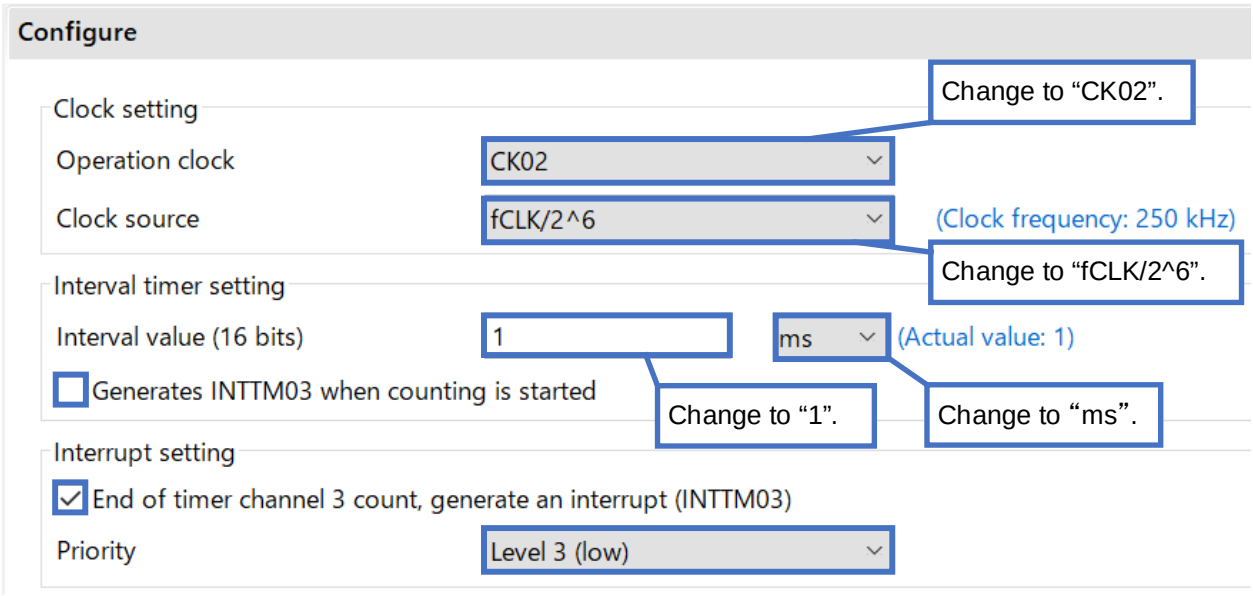



Figure 3-1 1-ms Interval Timer Settings

Figure 3-2 shows the A/D converter settings to select analog input channel 4 and specify 10-bit data for conversion results.

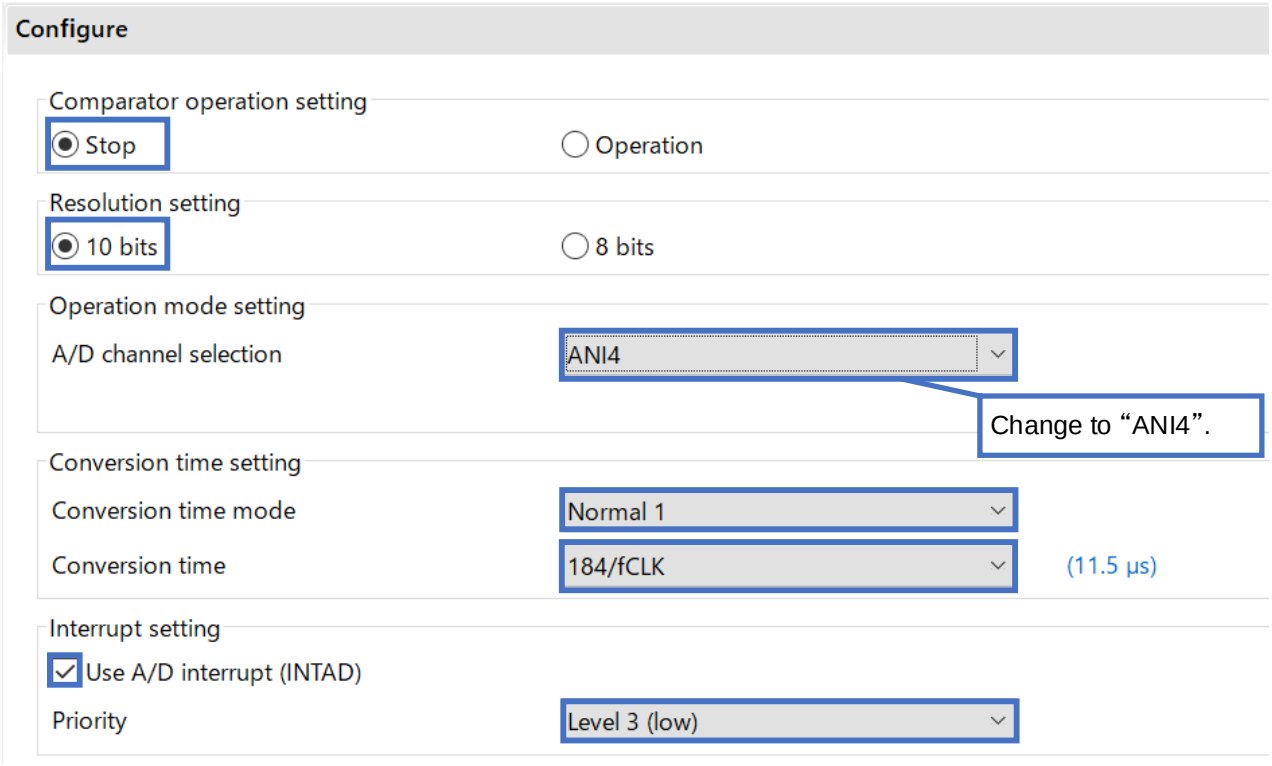



Figure 3-2 Analog Input Settings

Figure 3-3 shows the SPI communication settings to specify MSB first for the data transfer direction and 4 Mbps for the communication speed.

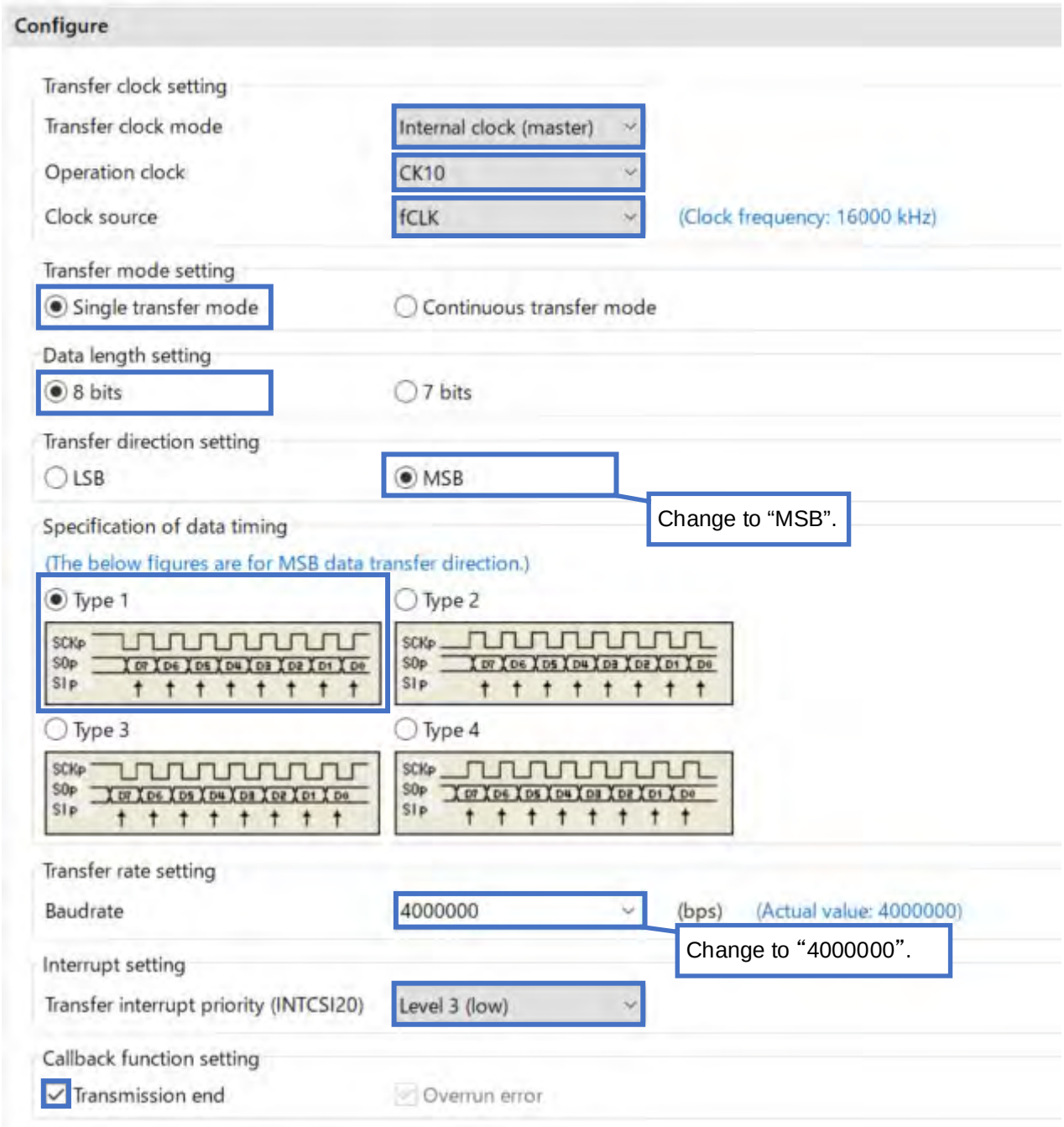



Figure 3-3 SPI Communication Settings

Figure 3-4 shows the communication settings for UART transmission that comply with the MIDI communication standard.

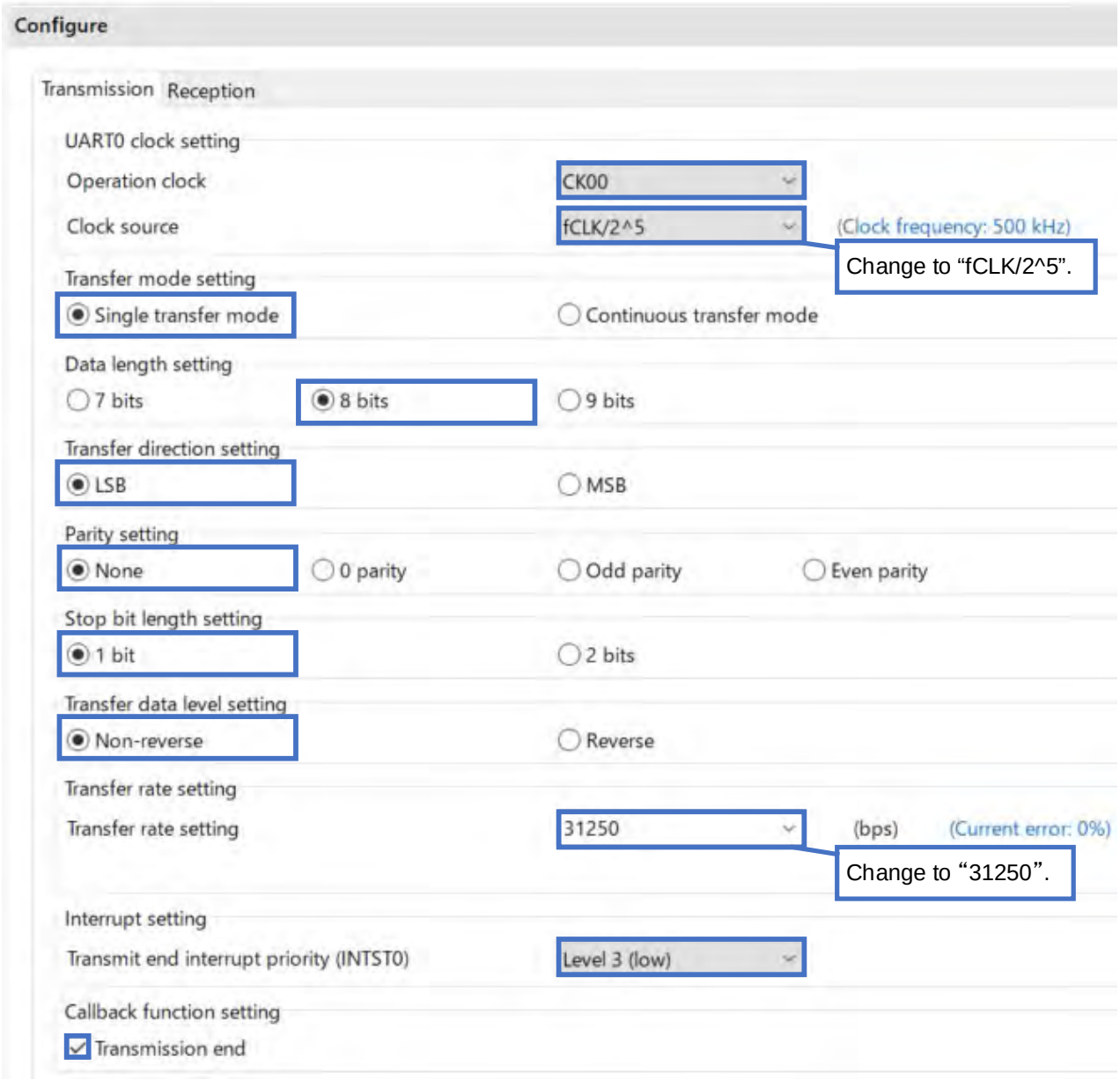



Figure 3-4 Communication Settings for MIDI Transmission

Figure 3-5 shows the communication settings for UART reception that comply with the MIDI communication standard.

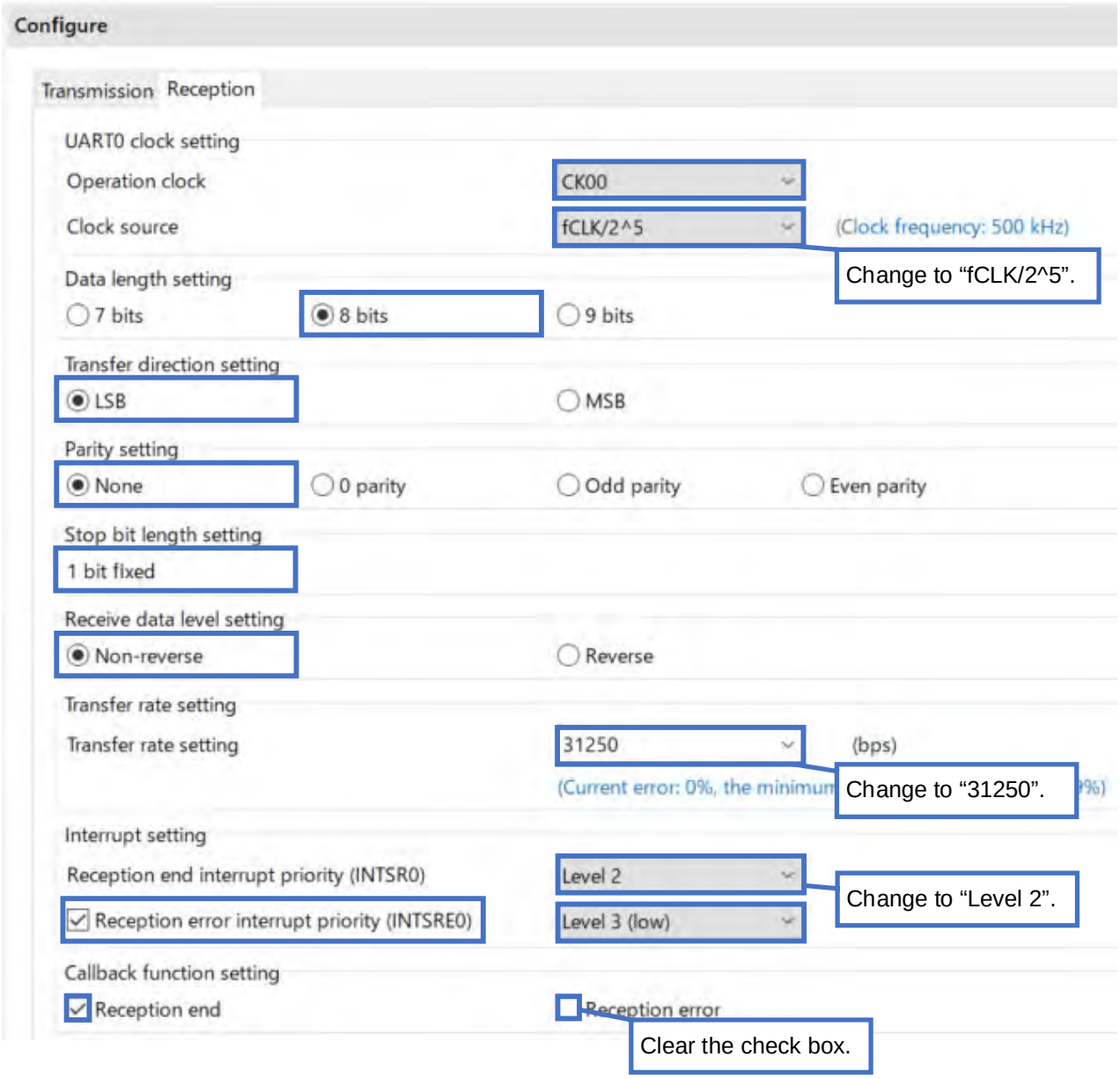



Figure 3-5 Communication Settings for MIDI Reception

## 3.3 Setting of Option Byte

Table 3-2 shows the option byte settings.

Table 3-2 Setting of Option Byte

| Address | Setting Value | Contents |
|---|---|---|
| 000C0H | 11101111B | Disables the watchdog timer.<br>(Counting stopped after reset) |
| 000C1H | 11110111B | SPOR operations (VSPOR)<br>At rising edge TYP. 2.90V (2.76 V ～ 3.02 V)<br>At falling edge TYP. 2.84V (2.70 V ～ 2.96 V) |
| 000C2H | 11111001B | High-speed on-chip oscillator clock: 16MHz |
| 000C3H | 10000101B | Enables on-chip debugging |

## 3.4 List of Macros

Table 3-3 lists the macros used in the sample program.

Table 3-3 Macros Used in Sample Program (1/2)

| Macro Name | Set Value | Description |
|---|---|---|
| DEMO_MIDI_NOTE_MAX | 127 | Maximum MIDI note value |
| DEMO_MIDI_PITCH_NUM | 12 | Number of notes in one octave |
| DEMO_MIDI_DISPlAY_VOL_MAX | 8 | Maximum MIDI volume |
| DEMO_MIDI_CH_MAX | 16 | Maximum MIDI channel number |
| DEMO_MIDI_CH_MIN | 1 | Minimum MIDI channel number |
| DEMO_MIDI_MIDDLE_C | 60 | Medium MIDI note value |
| DEMO_MIDI_VELOCITY | 127 | Velocity in NoteOn messages |
| DEMO_ADC_DATA_DIVISION | 1024 | Volume switch input resolution |
| DEMO_ADC_BUFF_SIZE | 4 | Number of volume switch data buffers |
| DEMO_ADC_INPUT_DIFFERENCE | 8 | Volume switch input threshold |
| DEMO_DISPLAY_MODE_MELODY | 0 | Display information type: Note display |
| DEMO_DISPLAY_MODE_CH_SET | 1 | Display information type: Channel display |
| DEMO_DISPLAY_MODE_COLOR_VOL_SET | 2 | Display information type: Volume display |
| DEMO_DISPLAY_MODE_OCTAVE_SHIFT_SET | 3 | Display information type: Pitch display |
| DEMO_DISPLAY_VOL_SET_TIME | 1000 | Volume display period [ms] |
| DEMO_DISPLAY_OCTAVE_SHIFT_SET_TIME | 1000 | Octave display period [ms] |
| DEMO_MATRIX_CATHODE_COLOR | 3 | Number of LED color elements: Red, Blue, Green |
| DEMO_MATRIX_DIGIT | 8 | Number of LED rows |
| DEMO_SOUND_VOLUME_PATTERN | 8 | Number of volume levels: 8 |
| DEMO_KYPD_OCTAVE_SHIFT_NONE | 5 | Initial pitch adjustment value (±0) |
| DEMO_KYPD_OCTAVE_SHIFT_MIN | 0 | Minimum pitch adjustment value (-5) |
| DEMO_KYPD_OCTAVE_SHIFT_MAX | 10 | Maximum pitch adjustment value (+5) |
| KYPD_SET_MODE_OUTPUT | 0 | Mode register output mode set value |
| KYPD_SET_MODE_INPUT | 1 | Mode register input mode set value |
| DEMO_SYSTEM_TIMER_START_FUNC | - | Alias of R_Config_TAU0_3_Start |
| DEMO_MATRIX_LED_SPI_CSPIN | - | Alias of CSI20 chip selection (P1_bit.no6) |
| DEMO_MATRIX_LED_SPI_START_FUNC | - | Alias of R_Config_CSI20_DEMO_MATRIX_LED_Start |
| DEMO_MATRIX_LED_SPI_SEND_FUNC | - | Alias of R_Config_CSI20_DEMO_MATRIX_LED_Send |
| DEMO_ANALOG_VOLUME_INPUT_START_FUNC | - | Macro that sequentially calls R_Config_ADC_DEMO_VOLUME_Set_OperationOn() and R_Config_ADC_DEMO_VOLUME_Start() to prepare for starting AD conversion |
| DEMO_KYPD_MATRIX_ROW1_PIN | - | Alias of key matrix input ROW1 (P1_bit.no2) |
| DEMO_KYPD_MATRIX_ROW2_PIN | - | Alias of key matrix input ROW2 (P13_bit.no7) |

Table 3-4 Macros Used in Sample Program (2/2)

| マクロ名 | Set Value | Description |
|---|---|---|
| DEMO_KYPD_MATRIX_ROW3_PIN | - | Alias of key matrix input ROW3 (P4_bit.no3) |
| DEMO_KYPD_MATRIX_ROW4_PIN | - | Alias of key matrix input ROW4 (P1_bit.no1) |
| DEMO_KYPD_MATRIX_COL1_PIN | - | Alias of key matrix output COL1 (P6_bit.no1) |
| DEMO_KYPD_MATRIX_COL2_PIN | - | Alias of key matrix output COL2 (P6_bit.no0) |
| DEMO_KYPD_MATRIX_COL3_PIN | - | Alias of key matrix output COL3 (P2_bit.no1) |
| DEMO_KYPD_MATRIX_COL4_PIN | - | Alias of key matrix output COL4 (P1_bit.no0) |
| DEMO_KYPD_MATRIX_COL1_PINMODE | - | Alias of key matrix output COL1 mode register (PM6_bit.no1) |
| DEMO_KYPD_MATRIX_COL2_PINMODE | - | Alias of key matrix output COL2 mode register (PM6_bit.no0) |
| DEMO_KYPD_MATRIX_COL3_PINMODE | - | Alias of key matrix output COL3 mode register (PM2_bit.no1) |
| DEMO_KYPD_MATRIX_COL4_PINMODE | - | Alias of key matrix output COL4 mode register (PM1_bit.no0) |

## 3.5 List of Constants

Table 3-5 list global variables.

Table 3-5 Constants

| Type | Constant Name | Description | Used Functions |
|---|---|---|---|
| const uint8_t | g_anti_blur | Anti-blur LED data | demo_display_main |
| const uint8_t | g_disp_scale | Scale data table by lighting position for the LED matrix | demo_display_main |
| const uint8_t | g_color_table_tone | LED data table by note in melody mode | demo_display_main |
| const uint8_t | g_display_ch_graph | LED data table to display channel settings when changing channels | demo_display_main |
| const uint8_t | g_color_table_vol | LED data table to display the volume level when changing the volume | demo_display_main |
| const uint8_t | g_sound_volume_msk | Mask data table to display the volume level when changing the volume | demo_display_main |
| const uint8_t | g_display_octave_shift_graph | LED data table to display the pitch adjustment value when changing the pitch | demo_display_main |
| const uint8_t | g_kypd_tenkey | Hexadecimal keyboard input value table for MIDI channel change mode | main |
| const uint8_t | g_disp_volume | Volume data table by volume level | main |
| const timbre_tone_t | g_demo_timbre | Tone setting information table by MIDI channel | main |

## 3.6 List of Variables

Table 3-6 through Table 3-8 list global variables.

Table 3-6 Global Variables (1/3)

| Type | Variable Name | Description | Used Functions |
|---|---|---|---|
| uint8_t | g_demo_adc_finish | Flag indicating that AD conversion with the volume switch ends | main、r_Config_ADC_DEMO_VOLUME_interrupt、demo_volume_input |
| uint16_t | g_demo_adc_data | AD value of the volume switch | main、r_Config_ADC_DEMO_VOLUME_interrupt、demo_volume_input |
| uint16_t | g_demo_adc_buff | AD value buffer of the volume switch | main、demo_volume_monitor_main、demo_volume_input |
| uint16_t | g_demo_adc_average | Average AD value of the volume switch | main、demo_volume_monitor_main |
| uint16_t | g_demo_adc_average_bak | Previous average AD value of the volume switch | main、demo_volume_monitor_main |
| uint8_t | g_demo_adc_step | State of AD conversion processing of the volume switch: 0 (conversion stopped), 1 (conversion in progress) | main、demo_volume_monitor_main、demo_volume_input |
| uint8_t | g_demo_adc_input_index | Index of the AD value buffer of the volume switch | demo_volume_input |
| pitch_blink_t | g_demo_matrix_ch_info | Display information by note | main、demo_display_main |
| uint8_t | g_demo_kypd_buff | Hexadecimal keyboard input status buffer | demo_kypd_matrix_main、demo_display_main |
| uint8_t | g_demo_spi_sending_flag | Flag indicating LED data transmission in progress: 0 (transmission end), 1 (transmission in progress) | main、r_Config_CSI20_DEMO_MATRIX_LED_callback_sendend、demo_matrix_led_data_send、demo_matrix_led_send_busy_check |

Table 3-7 Global Variables (2/3)

| Type | Variable Name | Description | Used Functions |
|---|---|---|---|
| uint32_t | g_demo_display_vol_start_time | Volume display start time when changing the volume | main、 |
| uint32_t | g_demo_display_octave_shift_start_time | Pitch information display start time when changing the pitch | main、 demo_display_main |
| uint8_t | g_demo_display_mode | LED display mode: DEMO_DISPLAY_MODE_MELODY (Melody mode), DEMO_DISPLAY_MODE_CH_SET (MIDI channel change mode), DEMO_DISPLAY_MODE_COLOR_VOL_SET (Volume change mode), DEMO_DISPLAY_MODE_OCTAVE_SHIFT_SET (Pitch change mode) | main、 demo_display_main |
| uint8_t | g_demo_last_display_mode | Return destination LED display mode: The LED display mode to be resumed after a given period since the volume change mode or pitch change mode was displayed. Either DEMO_DISPLAY_MODE_MELODY (melody mode) or DEMO_DISPLAY_MODE_CH_SET (MIDI channel change mode) can be specified. | main、 demo_display_main |
| uint8_t | g_demo_display_ch | Display target channel: 0 to 15 = Ch1 to Ch16, 16 = Display “0”- and wait for entry of the first digit, 17 = Display “1-“ and wait for entry of the first digit | main、 demo_display_main |
| uint8_t | g_demo_current_ch | Sound target channel: 0 to 15 = Ch1 to Ch16 | main、 demo_volume_monitor_main |
| uint8_t | g_demo_display_vol | Volume: 0 to 7 = Volume level 1 to 8 | main、 demo_volume_monitor_main 、 demo_display_main |
| uint32_t | g_demo_timer | System timer value of the sample program [ms] | demo_time_now、 demo_timer_cycle |
| uint8_t | g_demo_matrix_led_send_buff | Buffer for sending LED display data | demo_display_main |
| uint16_t | g_digit_now | Update target LED index | demo_display_main |
| uint8_t | g_demo_kypd_matrix_keys | Confirmation information for the input status for each key of the hexadecimal keyboard | main |

Table 3-8 Global Variables (3/3)

| Type | Variable Name | Description | Used Functions |
|---|---|---|---|
| uint8_t | g_demo_kypd_key_chg_timer | Retention time of the status change for each key of the hexadecimal keyboard | demo_kypd_timer_cycle、demo_kypd_start_key_chg_timer、demo_kypd_get_keys_chg_timer_cnt、 |
| uint8_t | g_demo_octave_shift | Pitch adjustment value | main、demo_display_main |

## 3.7 List of Functions

Table 3-9 lists the functions.

Table 3-9 Functions

| Function Name | Overview |
|---|---|
| demo_volume_monitor_main() | Volume setting update processing |
| demo_volume_input() | Processing to acquire the volume switch status |
| demo_display_main() | LED data update processing |
| demo_time_now() | Acquires the time elapsed from the start of the program. |
| demo_timer_cycle() | Called from a TAU0_3 periodic interrupt to update the time elapsed from the start of the program. |
| demo_matrix_led_data_send() | LED data transmission processing |
| demo_matrix_led_send_busy_check() | Processing to monitor the end of LED data transmission |
| R_Config_TAU0_3_Start() | TAU0_3 timer start processing |
| r_Config_TAU0_3_interrupt() | Callback processing for a TAU0_3 periodic interrupt |
| R_Config_ADC_DEMO_VOLUME_Start() | Clears the AD conversion end interrupt flag, enables AD conversion end interrupts, and enables AD conversion operation. |
| R_Config_ADC_DEMO_VOLUME_ Set_OperationOn() | Enables AD voltage comparator operations. |
| R_Config_ADC_DEMO_VOLUME_Get_Result_10bit() | Processing to acquire AD conversion results |
| r_Config_ADC_DEMO_VOLUME_interrupt() | Callback processing when INTAD AD conversion ends |
| R_Config_CSI20_DEMO_MATRIX_LED_Start() | CSI20 start processing |
| R_Config_CSI20_DEMO_MATRIX_LED_Send() | CSI20 data transmission processing |
| r_Config_CSI20_DEMO_MATRIX_LED_ callback_sendend() | Callback processing when CSI20 transmission ends |
| r_Config_CSI20_DEMO_MATRIX_LED_interrupt() | CSI20 transfer end interrupt processing |
| R_Config_UART0_Send() | UART0 transmission processing |
| R_Config_UART0_Receive() | UART0 reception processing |
| r_Config_UART0_callback_sendend() | UART0 transmission end processing |
| r_Config_UART0_callback_receiveend() | UART0 reception end processing |
| r_Config_UART0_interrupt_send() | UART0 transmission interrupt processing |
| r_Config_UART0_interrupt_receive() | UART0 reception interrupt processing |
| r_Config_UART0_interrupt_error() | Communication error interrupt processing in UART0 reception |
| demo_kypd_timer_cycle() | Called from a TAU0_3 periodic interrupt to update the key input confirmation timer count for the hexadecimal keyboard |
| demo_kypd_start_key_chg_timer() | Processing to start the key input change confirmation timer |
| demo_kypd_matrix_sense() | Processing to determine the confirmed key input status |
| demo_kypd_matrix_main() | Main processing of hexadecimal keyboard input |
| demo_kypd_matrix_col_read() | Processing to read the hexadecimal keyboard column status |
| demo_kypd_get_keys_chg_timer_cnt() | Processing to acquire the value of the key input change confirmation timer |

## 3.8 Function Specifications

This section describes the function specifications of the sample program.

| demo_volume_monitor_main() | |
|---|---|
| Overview | Volume setting update processing |
| Header | - |
| Declaration | uint8_t demo_volume_monitor_main(void); |
| Description | This function updates the volume according to the change in the volume switch. |
| Arguments | None |
| Return values | 0: The volume is not changed.<br>1: The volume is changed. |

| demo_volume_input() | |
|---|---|
| Overview | Processing to acquire the status of the volume switch |
| Header | - |
| Declaration | uint8_t demo_volume_input(void); |
| Description | This function acquires the status of the volume switch. |
| Arguments | None |
| Return values | 0: Volume switch input is being acquired.<br>1: Volume switch input is acquired. |

| demo_display_main() | |
|---|---|
| Overview | Processing to update the LED matrix display |
| Header | - |
| Declaration | void demo_display_main(void); |
| Description | This function updates the contents of the LED matrix display. |
| Arguments | None |
| Return values | None |

| demo_time_now() | |
|---|---|
| Overview | Acquisition of the current time |
| Header | - |
| Declaration | uint32_t demo_time_now(void); |
| Description | This function returns the time elapsed from the start of the sample program. |
| Arguments | None |
| Return values | uint32_t: Time elapsed from the start of the sample program |

| demo_timer_cycle() | |
|---|---|
| Overview | Time update |
| Header | - |
| Declaration | void demo_timer_cycle(void); |
| Description | This function is called from within periodic interrupt processing to update the time elapsed from the start of the sample program. |
| Arguments | None |
| Return values | None |

| demo_matrix_led_data_send() | |
|---|---|
| Overview | LED data transmission |
| Header | - |
| Declaration | void demo_matrix_led_data_send(uint8_t * data, uint16_t len); |
| Description | This function sends data to the LED matrix. |
| Arguments | uint8_t * data: Send data address<br>uint16_t len: Send data size |
| Return values | None |

| demo_matrix_led_send_busy_check() | |
|---|---|
| Overview | LED data transmission end monitoring |
| Header | - |
| Declaration | uint8_t demo_matrix_led_send_busy_check(void); |
| Description | This function references the communication-in-progress flag (g_demo_spi_sending_flag) and returns the transmission status. |
| Arguments | None |
| Return values | 0: Transmission end<br>1: Transmission in progress |

| r_Config_TAU0_3_interrupt() | |
|---|---|
| Overview | Interval timer interrupt processing |
| Header | r_cg_macrodriver.h, r_cg_userdefine.h, Config_TAU0_3.h, r_midi_rl78_if.h |
| Declaration | static void __near r_Config_TAU0_3_interrupt(void); |
| Description | This is a TAU0_3 count end interrupt function.<br>This function calls a MIDI 1-ms interval notification function.<br>It also calls a system timer count function.<br>It also calls a timer count function for hexadecimal keyboard input confirmation. |
| Arguments | None |
| Return values | None |

| r_Config_ADC_DEMO_VOLUME_interrupt() | |
|---|---|
| Overview | AD conversion end interrupt processing |
| Header | r_cg_macrodriver.h, r_cg_userdefine.h, Config_ADC_DEMO_VOLUME.h |
| Declaration | static void __near r_Config_ADC_DEMO_VOLUME_interrupt(void); |
| Description | This is an A/D conversion end interrupt function.<br>This function reads the AD conversion results and then saves them in the buffer (g_demo_adc_data).<br>It sets the AD conversion end flag (g_demo_adc_finish). |
| Arguments | None |
| Return values | None |

| r_Config_CSI20_DEMO_MATRIX_LED_callback_sendend() | |
|---|---|
| Overview | Callback processing when CSI20 transmission ends |
| Header | r_cg_macrodriver.h, r_cg_userdefine.h, Config_CSI20_DEMO_MATRIX_LED.h, midi_matrixled_demo.h |
| Declaration | static void r_Config_CSI20_DEMO_MATRIX_LED_callback_sendend(void); |
| Description | This callback function is called when CSI20 transmission ends.<br>This function sets the SPI CS pin to High.<br>Then, this function resets the communication-in-progress flag (g_demo_spi_sending_flag). |
| Arguments | None |
| Return values | None |

| r_Config_UART0_callback_sendend() | |
|---|---|
| Overview | UART0 transmission end callback processing |
| Header | r_cg_macrodriver.h, r_cg_userdefine.h, Config_UART0.h, r_midi_rl78_if.h |
| Declaration | static void r_Config_UART0_callback_ sendend(void); |
| Description | This function is called to perform callback processing when UART0 transmission ends.<br>This function calls (R_MIDI_NotifyEvent) to notify the MIDI interface SIS (Software Integration System) module of the end of transmission. |
| Arguments | None |
| Return values | None |

| r_Config_UART0_callback_receiveend() | |
|---|---|
| Overview | UART0 transmission end callback processing |
| Header | r_cg_macrodriver.h、r_cg_userdefine.h、Config_UART0.h、r_midi_rl78_if.h |
| Declaration | static void r_Config_UART0_callback_ sendend(void); |
| Description | This function is called to perform callback processing when UART0 transmission ends.<br>This function calls (R_MIDI_NotifyEvent) to notify the MIDI interface SIS (Software Integration System) module of the end of transmission. |
| Arguments | None |
| Return values | None |

| demo_kypd_timer_cycle() | |
|---|---|
| Overview | Update of the key input confirmation timer count |
| Header | midi_matrixled_kypd_demo.h |
| Declaration | void demo_kypd_timer_cycle(void); |
| Description | This function is called from within periodic interrupt processing to update the key input confirmation timer count. |
| Arguments | None |
| Return values | None |

| demo_kypd_start_key_chg_timer() | |
|---|---|
| Overview | Starting the input confirmation timer |
| Header | midi_matrixled_kypd_demo.h |
| Declaration | void demo_kypd_start_key_chg_timer(uint8_t index); |
| Description | This function sets the time (10 ms) after which the change of the target key will be confirmed. |
| Arguments | Index of the target key |
| Return values | None |

| demo_kypd_matrix_sense() | |
|---|---|
| Overview | Processing to determine the confirmed key input status |
| Header | midi_matrixled_kypd_demo.h |
| Declaration | uint8_t demo_kypd_matrix_sense(uint8_t *key); |
| Description | This function determines whether pressing or releasing of a specific key is confirmed, and then returns the result. |
| Arguments | Address of the key to be determined |
| Return values | 0: No change in the key status<br>DEMO_KYPD_KEYMAK E: Key-pressing confirmed<br>DEMO_KYPD_KEYBREAK: Key-releasing confirmed<br>DEMO_KYPD_KEYMAKE + DEMO_KYPD_KEYBREAK: Key-pressing and key-releasing confirmed |

| demo_kypd_matrix_main() | |
|---|---|
| Overview | Processing to update confirmation information of hexadecimal keyboard input |
| Header | midi_matrixled_kypd_demo.h |
| Declaration | void demo_kypd_matrix_main(uint8_t keys[][4]); |
| Description | A single call of this function updates input information for one column of the hexadecimal keyboard.<br>If any change is found in the hexadecimal keyboard input, this function sets a timer for how long the status must continue before confirmation.<br>If the key input status continues until the timer expires, the key input is confirmed and a confirmation flag indicating that the key is pressed or released is set in confirmation information. |
| Arguments | Address of confirmation information of hexadecimal keyboard input |
| Return values | None |

| demo_kypd_matrix_col_read() | |
|---|---|
| Overview | Input processing for one column of the hexadecimal keyboard |
| Header | midi_matrixled_kypd_demo.h |
| Declaration | uint8_t demo_kypd_matrix_col_read(uint8_t col); |
| Description | This function returns key pressing information for one column of the hexadecimal keyboard.<br>Each bit In this information is set to 0 (key not pressed) or 1 (key pressed).<br>BIT3: Key pressing status of row 1<br>BIT2: Key pressing status of row 2<br>BIT1: Key pressing status of row 3<br>BIT0: Key pressing status of row 4 |
| Arguments | Column number |
| Return values | Input status of the column of the hexadecimal keyboard specified by the argument |

| demo_kypd_get_keys_chg_timer_cnt() | |
|---|---|
| Overview | Processing to acquire the count value of the key input confirmation timer |
| Header | midi_matrixled_kypd_demo.h |
| Declaration | uint8_t demo_kypd_get_keys_chg_timer_cnt(uint8_t index); |
| Description | This function returns the count value of the input confirmation timer for the target key |
| Arguments | Index of the target key |
| Return values | Count value of the key input confirmation timer |

## 3.9 Flowcharts

### 3.9.1 Main processing

Figure 3-6、Figure 3-7 and Figure 3-8 show the flowchart for the main processing.

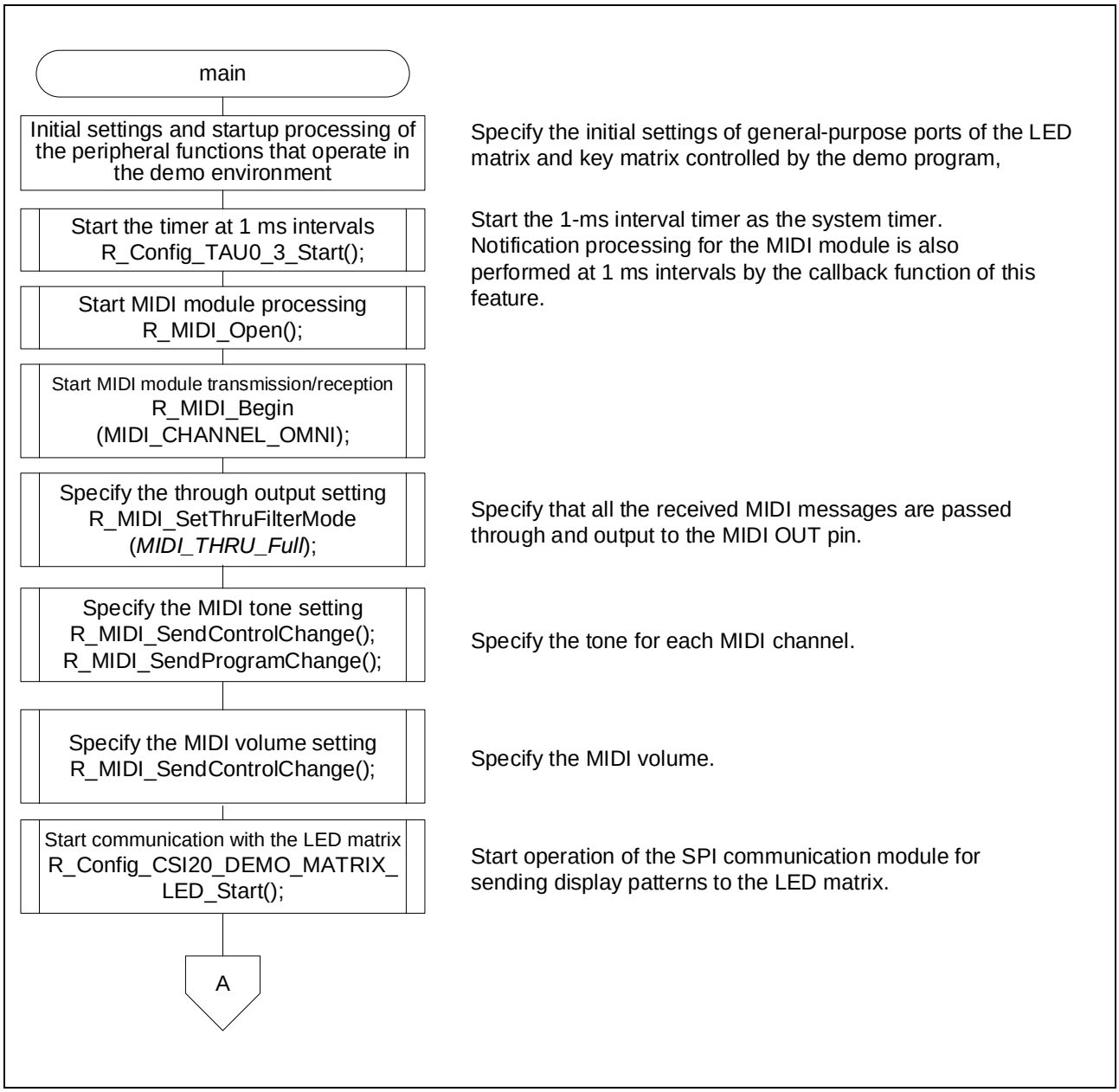



Figure 3-6 Main Processing (1/3)

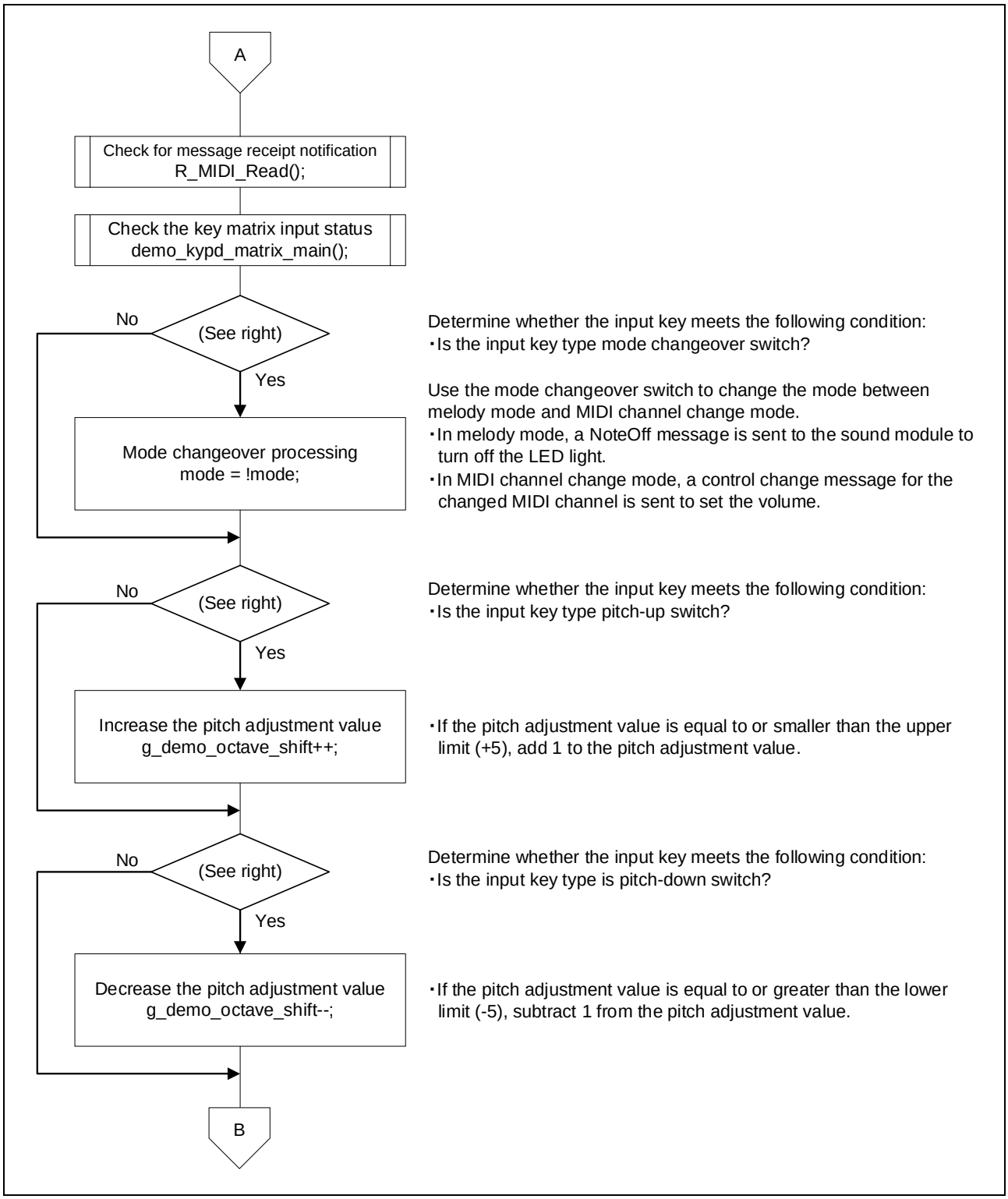
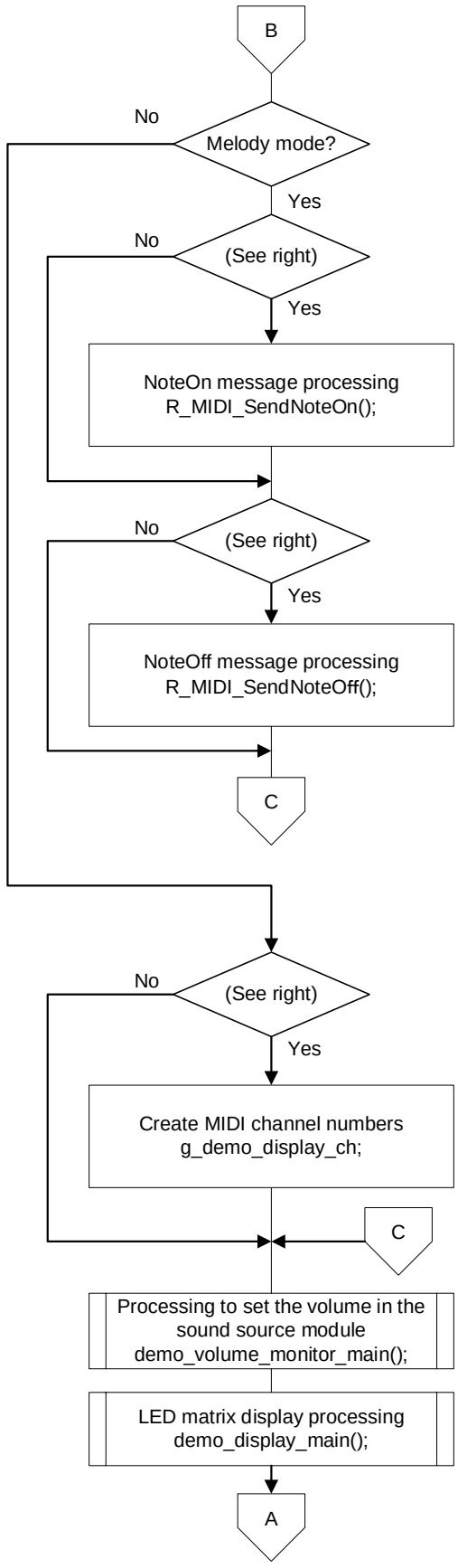



Figure 3-7 Main Processing (2/3)




Determine whether the current mode is melody mode.
In melody mode, you can play music by using key input.

Determine whether the input key meets the following condition:
・Is pressing of the key with a note assigned confirmed?

A sound is played when key-pressing is confirmed.
・Add the pitch adjustment value to the note number of the NoteOn message.
・Pitch adjustment values are given by one octave.
・Specify the scale information used to turn on the LED matrix lights corresponding to keys.

Determine whether the input key meets the following condition:
・Is releasing of the key with a note assigned confirmed?

The sound stops when key-releasing is confirmed.
・The note number of the NoteOff message is the note number specified for the NoteOn message.
・Specify the scale information used to turn off the LED matrix lights corresponding to keys.

If the current mode is not melody mode, it must be MIDI channel change mode.
In MIDI channel change mode, you can change MIDI channels by using key input.

Determine whether the input key meets the following condition:
・Is pressing of a numeric key confirmed?

Create MIDI channel numbers based on the order in which numeric keys are pressed.
・In MIDI channel change mode, the mode changeover switch is also used to confirm MIDI channels and updates channels if the entered two-digit number is within the range from 01 to 16.

Acquire the A/D conversion value from Volume0 and determine the volume to be set in the sound source module.

The volume level is displayed for 1 second immediately after Volume0 is turned. The pitch adjustment value is displayed for 1 second immediately after the pitch-up or pitch-down switch is pressed. In any other case, melody mode or MIDI channel change mode is displayed according to the mode setting.

Figure 3-8 Main Processing (3/3)

### 3.9.2 Volume setting update processing

Figure 3-9 shows the flowchart for volume setting update processing.

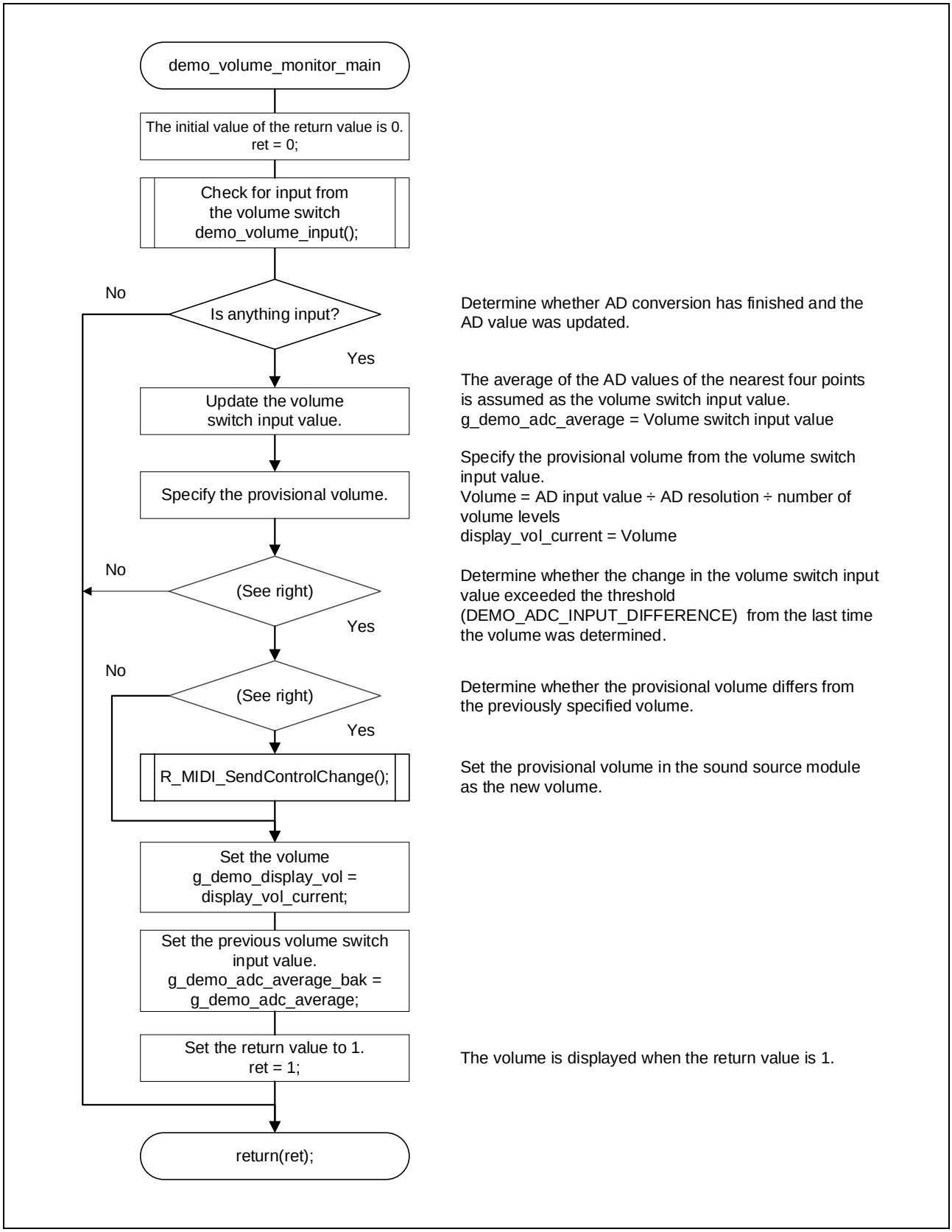



Figure 3-9 Volume Setting Update Processing

### 3.9.3 Processing to acquire volume switch input

Figure 3-10 shows the flowchart for processing to acquire volume switch input.

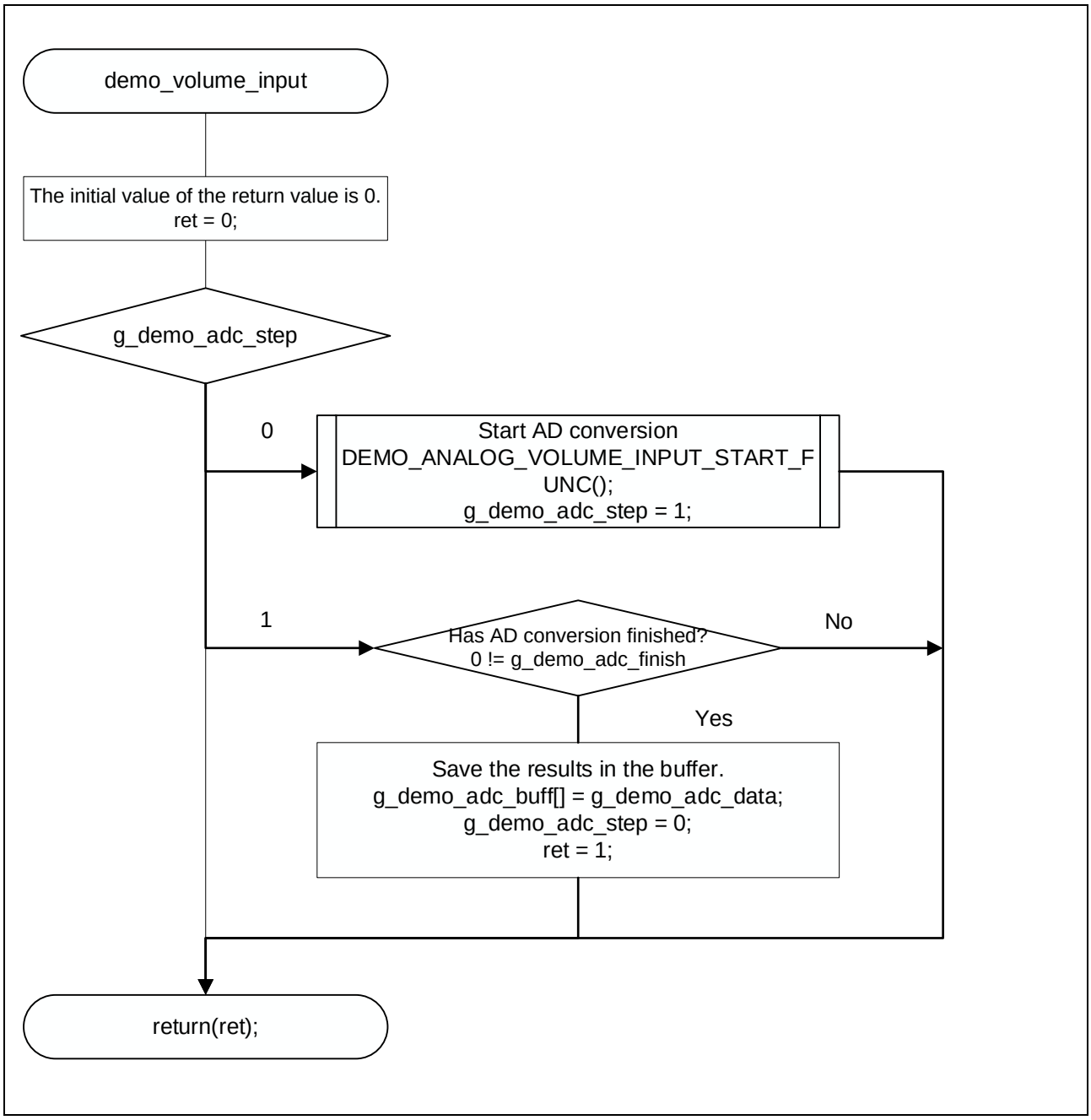



Figure 3-10 Processing to Acquire Volume Switch Input

### 3.9.4 LED matrix display processing

Figure 3-11 shows the flowchart for LED matrix display processing.

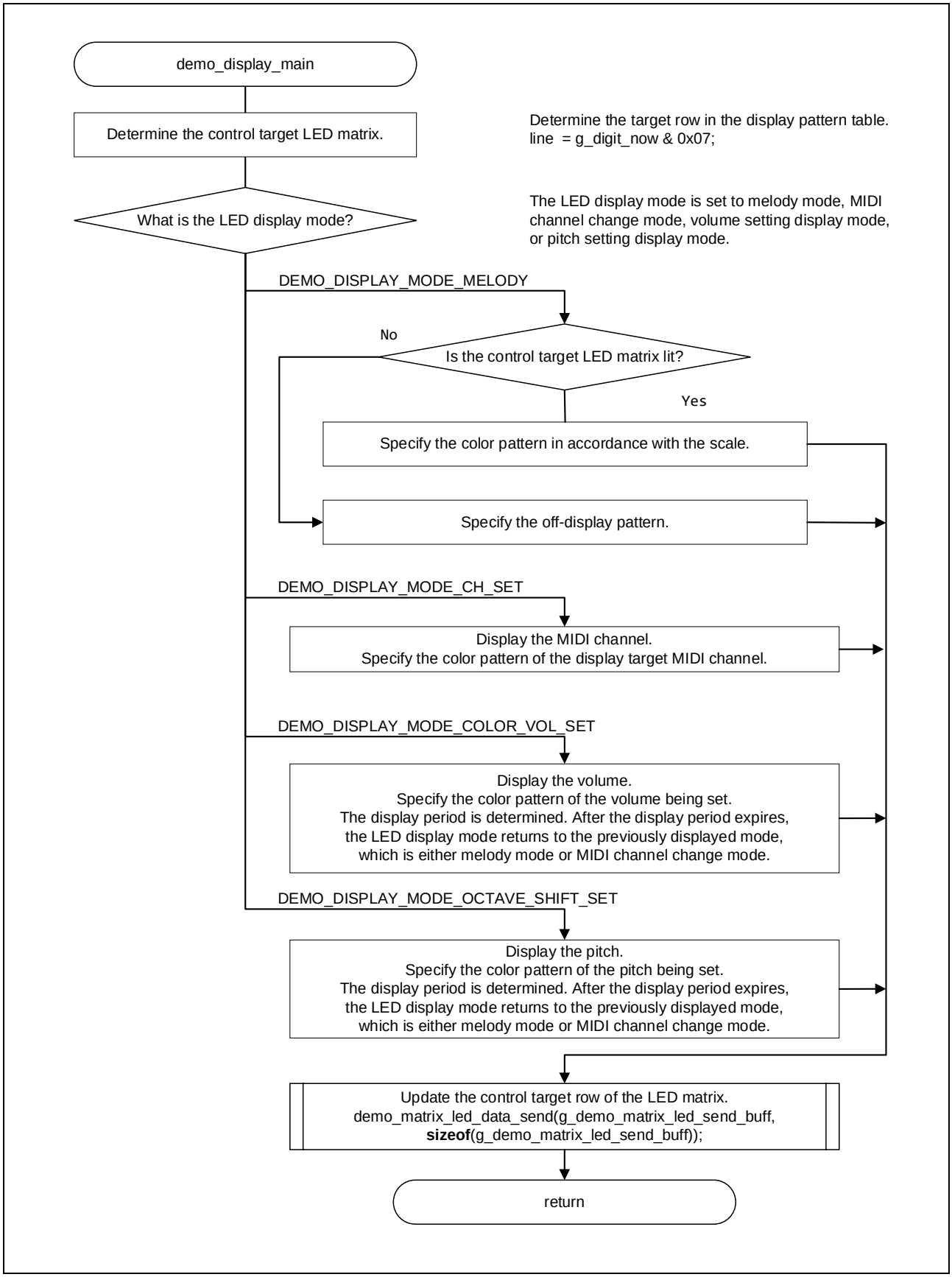



Figure 3-11 LED Matrix Display Processing.

### 3.9.5 System timer acquisition processing

Figure 3-12 shows the flowchart for system timer acquisition processing.

Note: This function simply returns a global variable that can be referenced from multiple locations. This function is provided to clearly indicate that the same variable is being referenced.

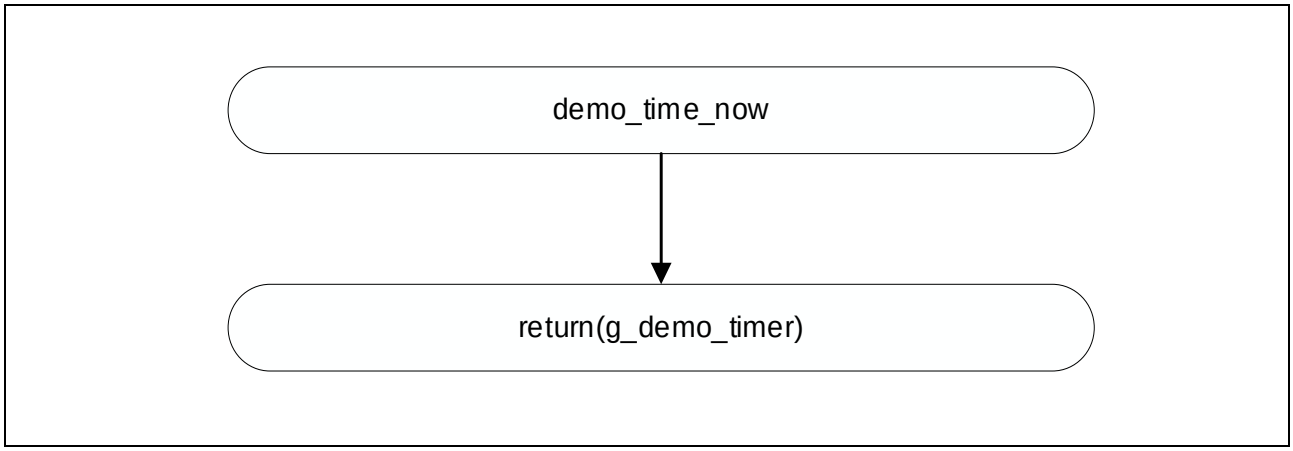



Figure 3-12 System Timer Acquisition Processing

### 3.9.6 System timer count processing

Figure 3-13 shows the flowchart for system timer count processing.

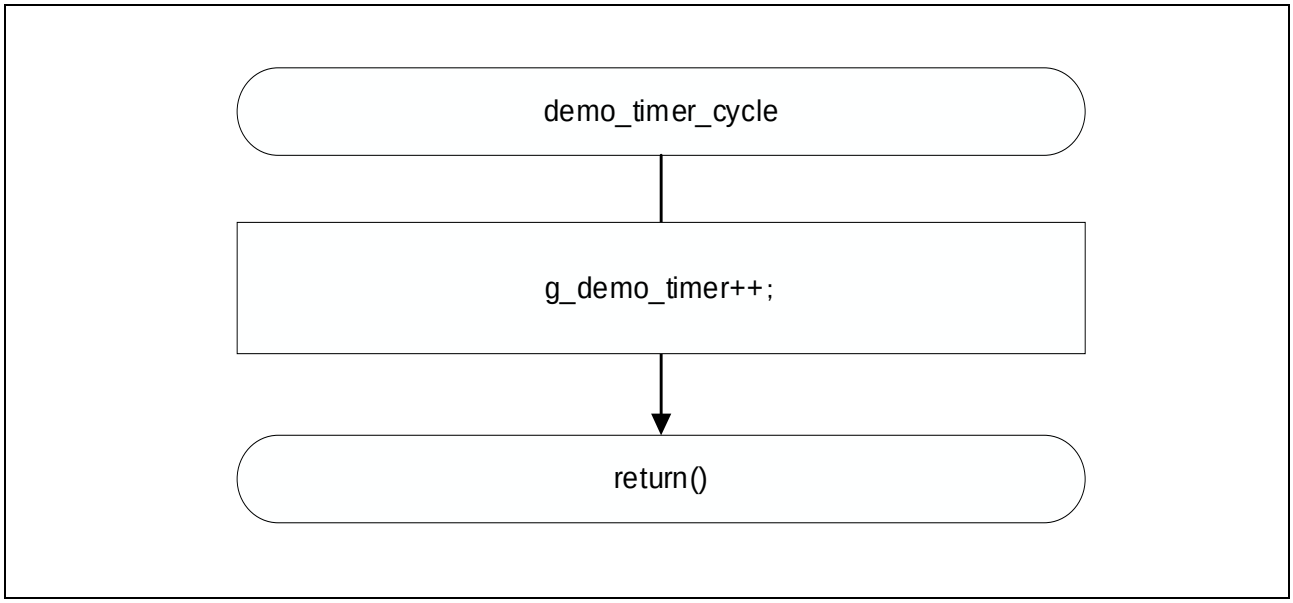



Figure 3-13 System Timer Count Processing

### 3.9.7 LED matrix data transmission processing

Figure 3-14 shows the flowchart for LED matrix data transmission processing.

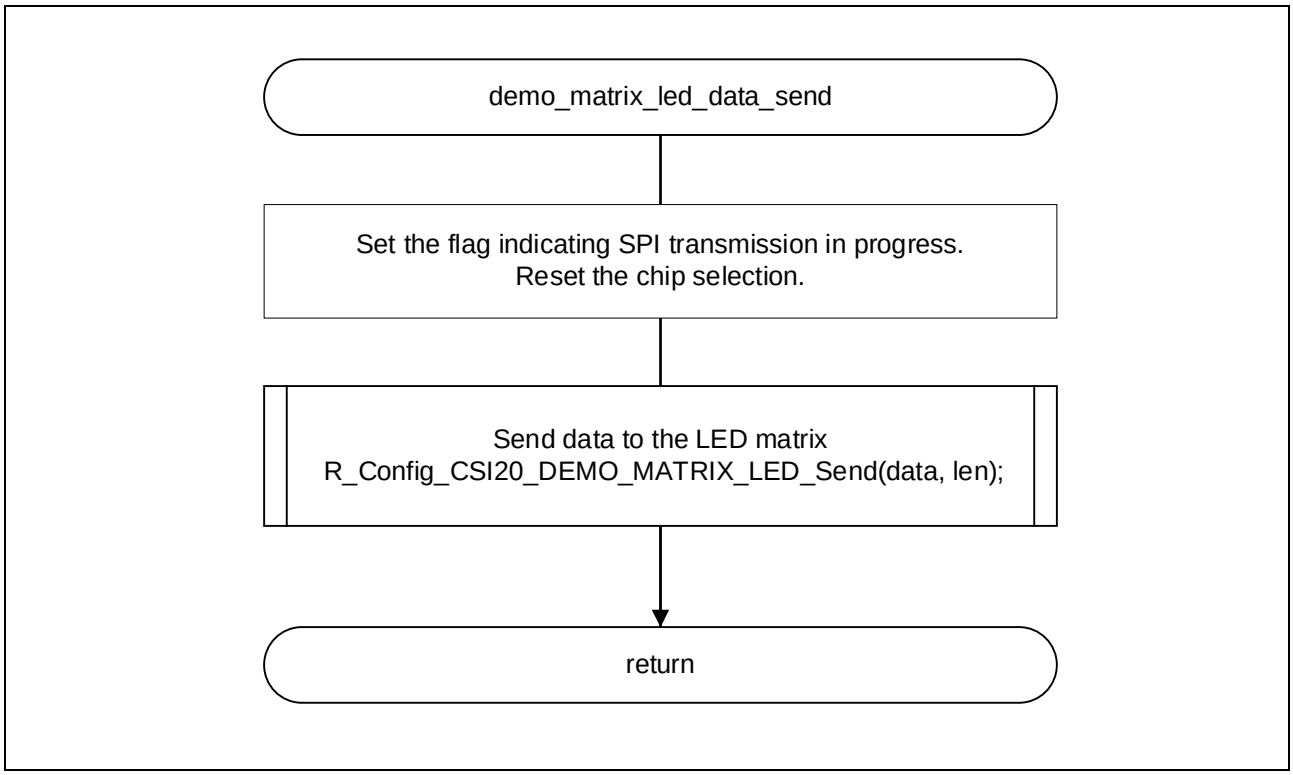



Figure 3-14 LED Matrix Data Transmission Processing

### 3.9.8 Processing to check the end of LED matrix data transmission

Figure 3-15 shows the flowchart for processing to check the end of LED matrix data transmission.

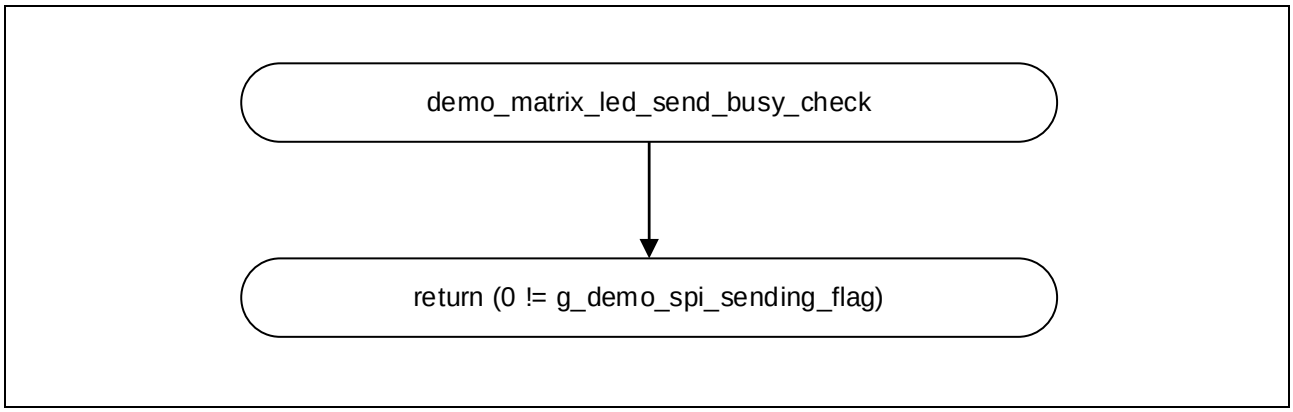



Figure 3-15 Processing to Check the End of LED Matrix Data Transmission

### 3.9.9 TAU0_3 interrupt processing

Figure 3-16 shows the flowchart for TAU0_3 interrupt processing.

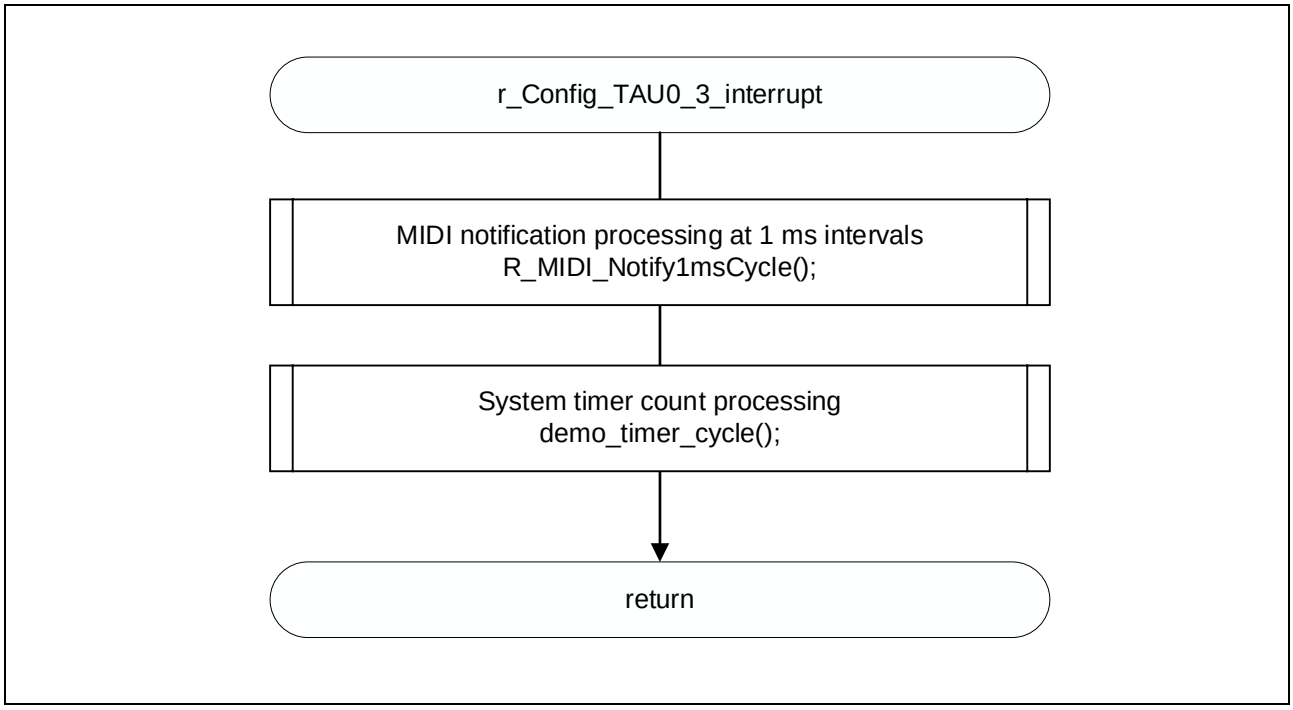



Figure 3-16 TAU0_3 Interrupt Processing

### 3.9.10 AD conversion end interrupt processing

Figure 3-17 shows the flowchart for AD conversion end interrupt processing.

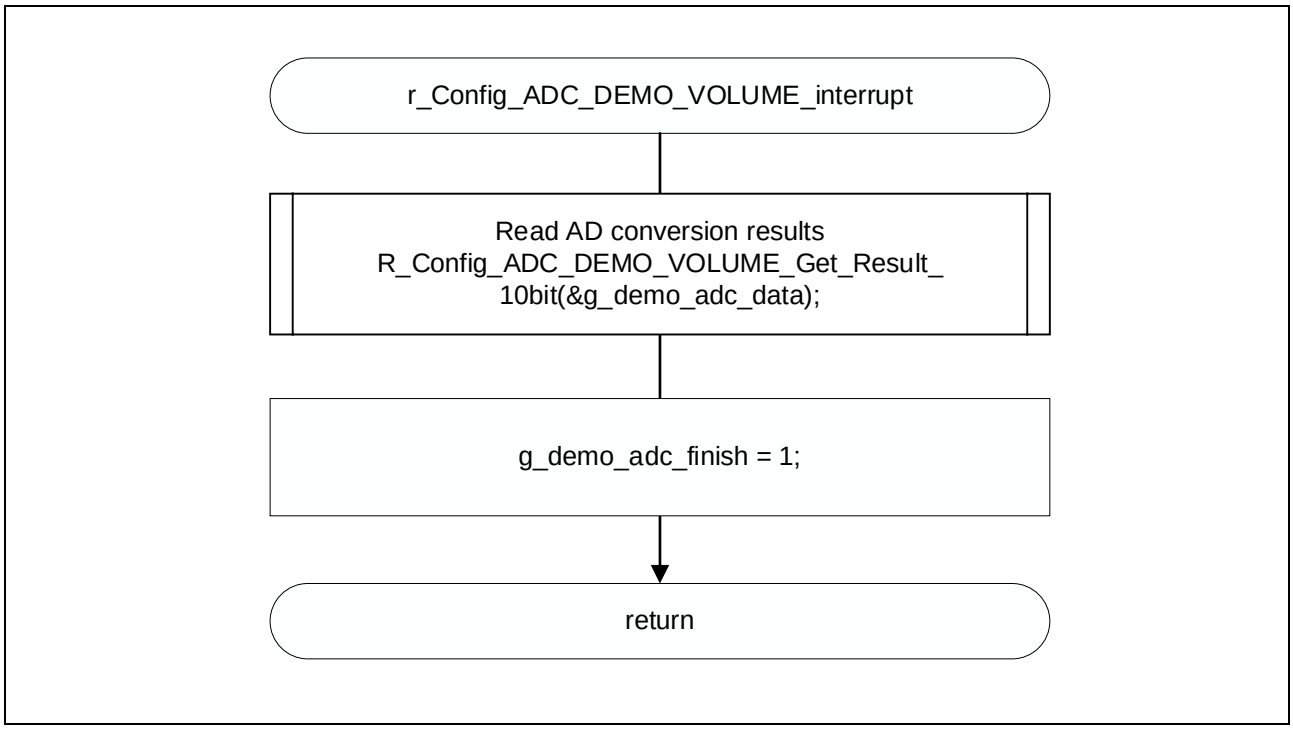



Figure 3-17 AD Conversion End Interrupt Processing

### 3.9.11 LED matrix data CSI transmission end processing

Figure 3-18 shows the flowchart for CSI transmission end processing for LED matrix data.

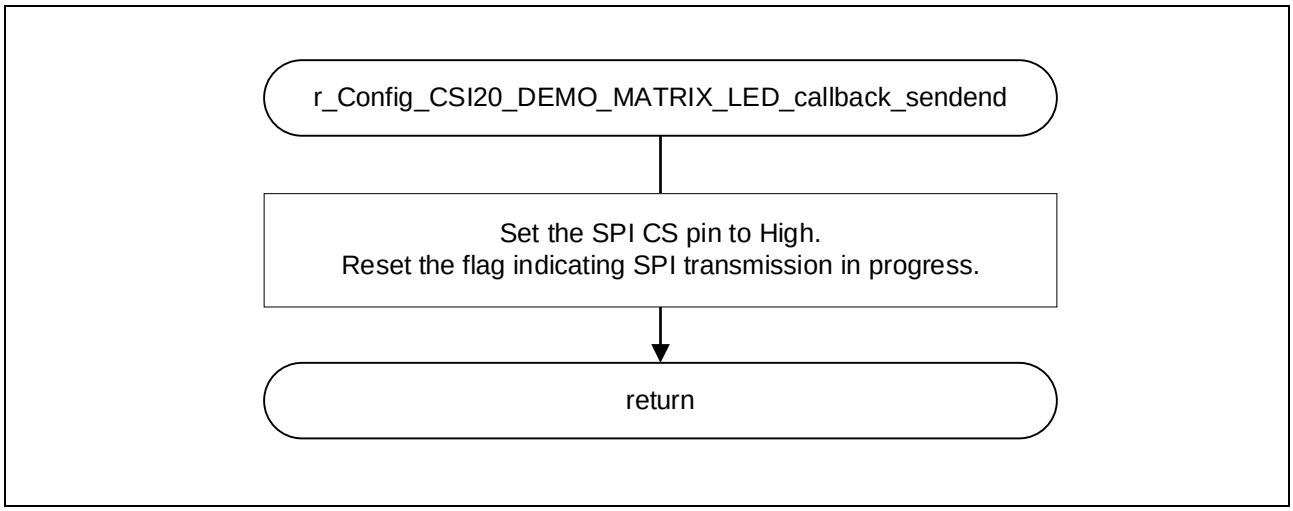



Figure 3-18 LED Matrix Data CSI Transmission End Processing

### 3.9.12 UART0 transmission end processing

Figure 3-19 shows the flowchart for UART0 transmission end processing.

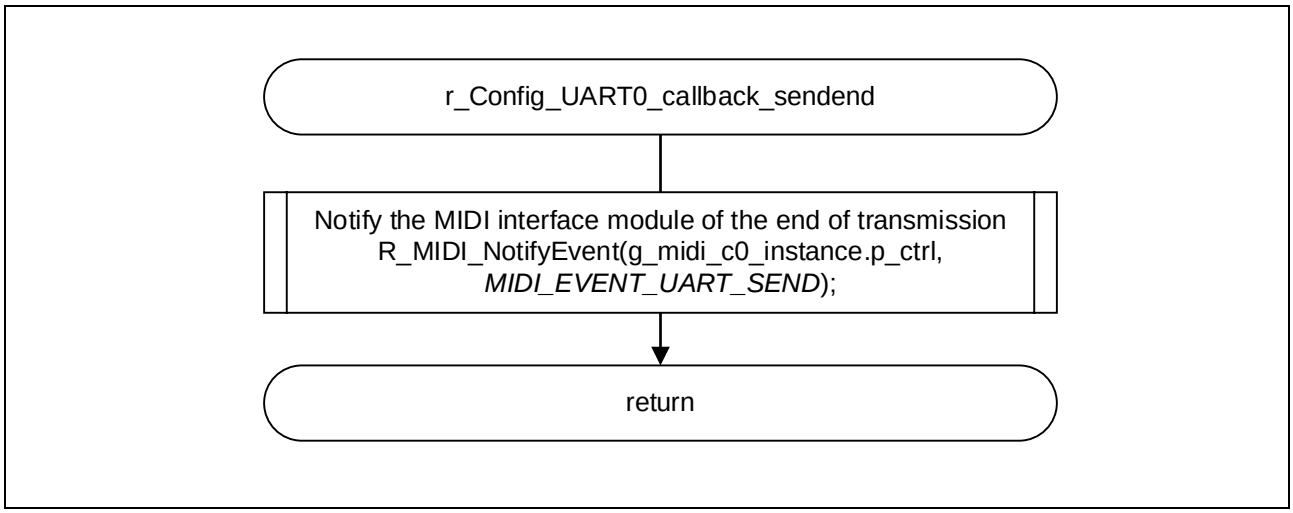



Figure 3-19 UART0 Transmission End Processing

### 3.9.13 UART0 reception end processing

Figure 3-20 shows the flowchart for UART0 reception end processing.

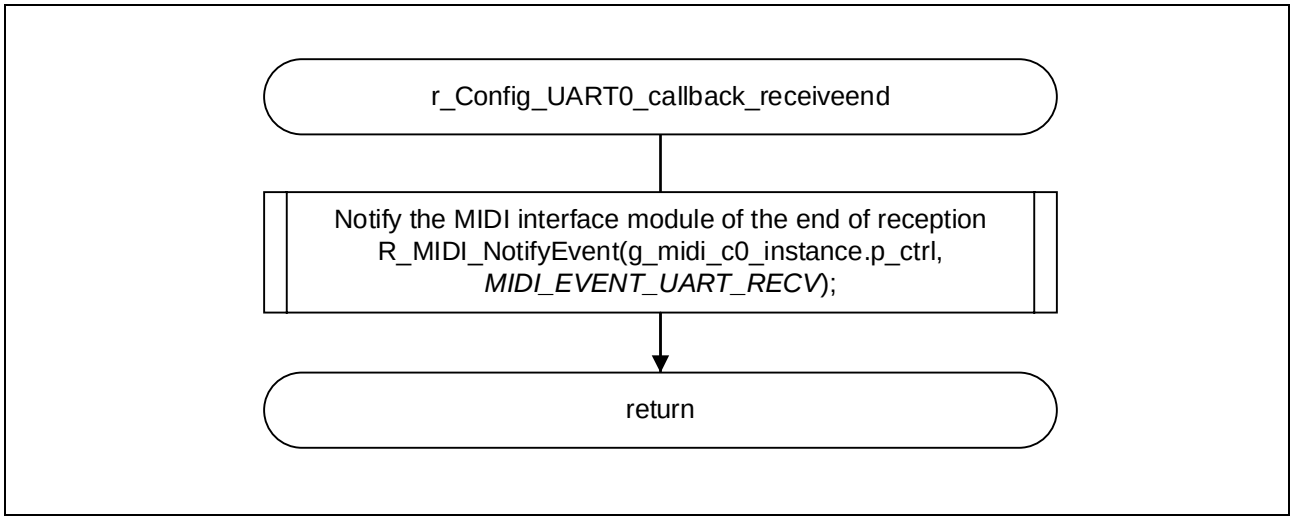



Figure 3-20 UART0 Reception End Processing

### 3.9.14 Hexadecimal keyboard key input processing

Figure 3-21 shows the flowchart for hexadecimal keyboard key input processing.

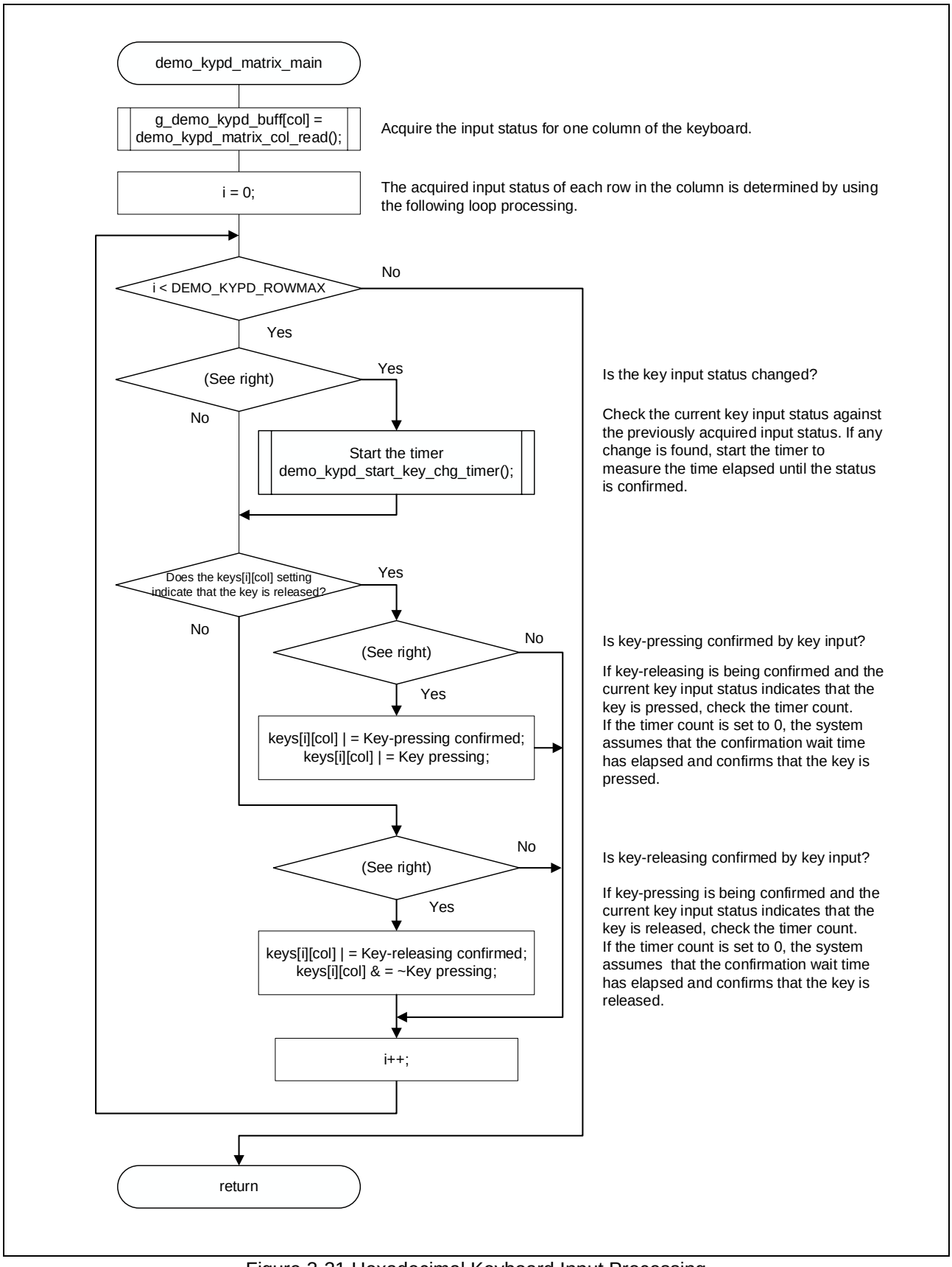



Figure 3-21 Hexadecimal Keyboard Input Processing

### 3.9.15 Input processing for one column of the hexadecimal keyboard

Figure 3-22 shows the flowchart for input processing for one column of the hexadecimal keyboard.

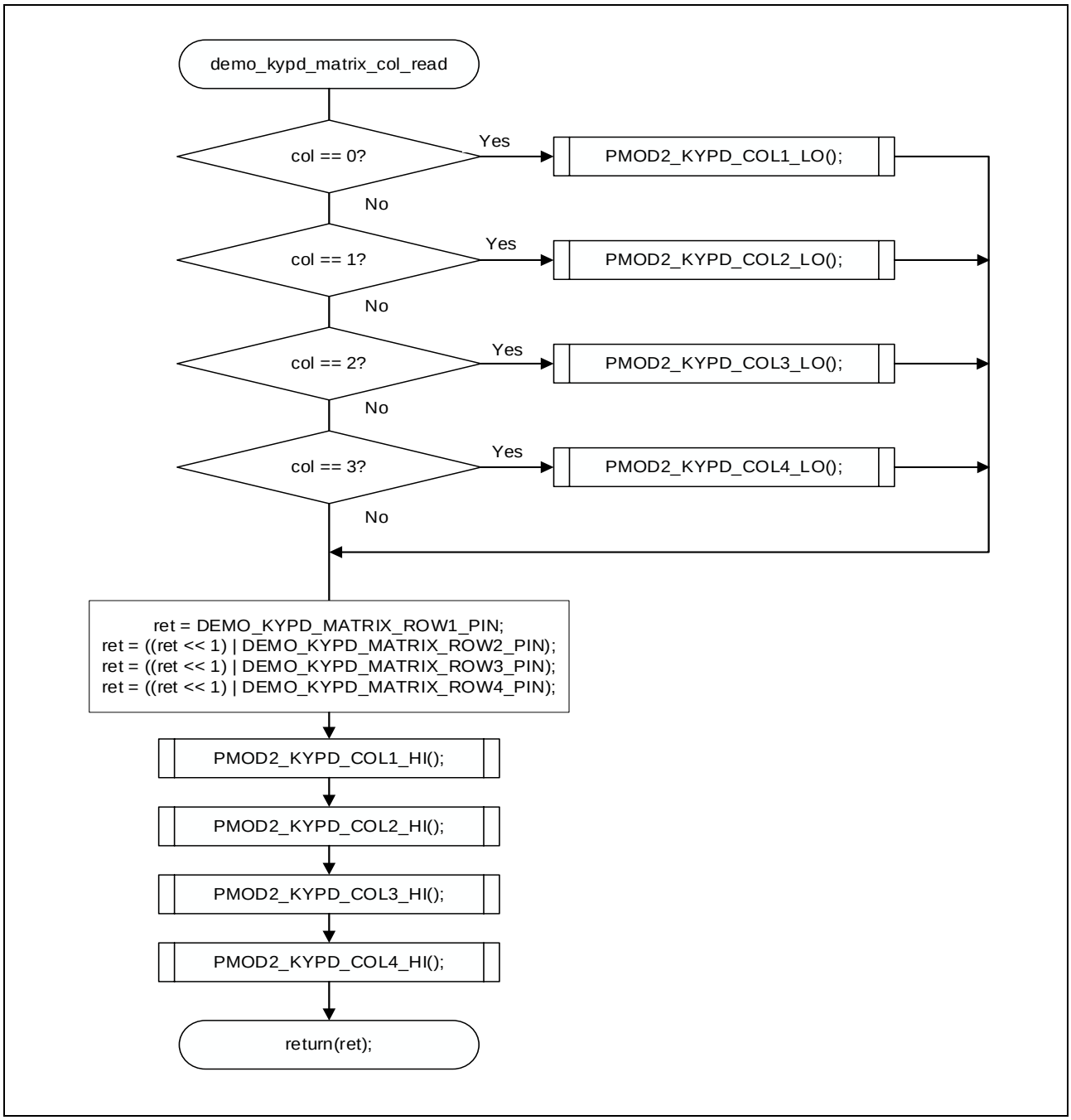



Figure 3-22 Input Processing for One Column of the Hexadecimal Keyboard

### 3.9.16 Key input determination processing

Figure 3-23 shows the flowchart for key input determination processing.

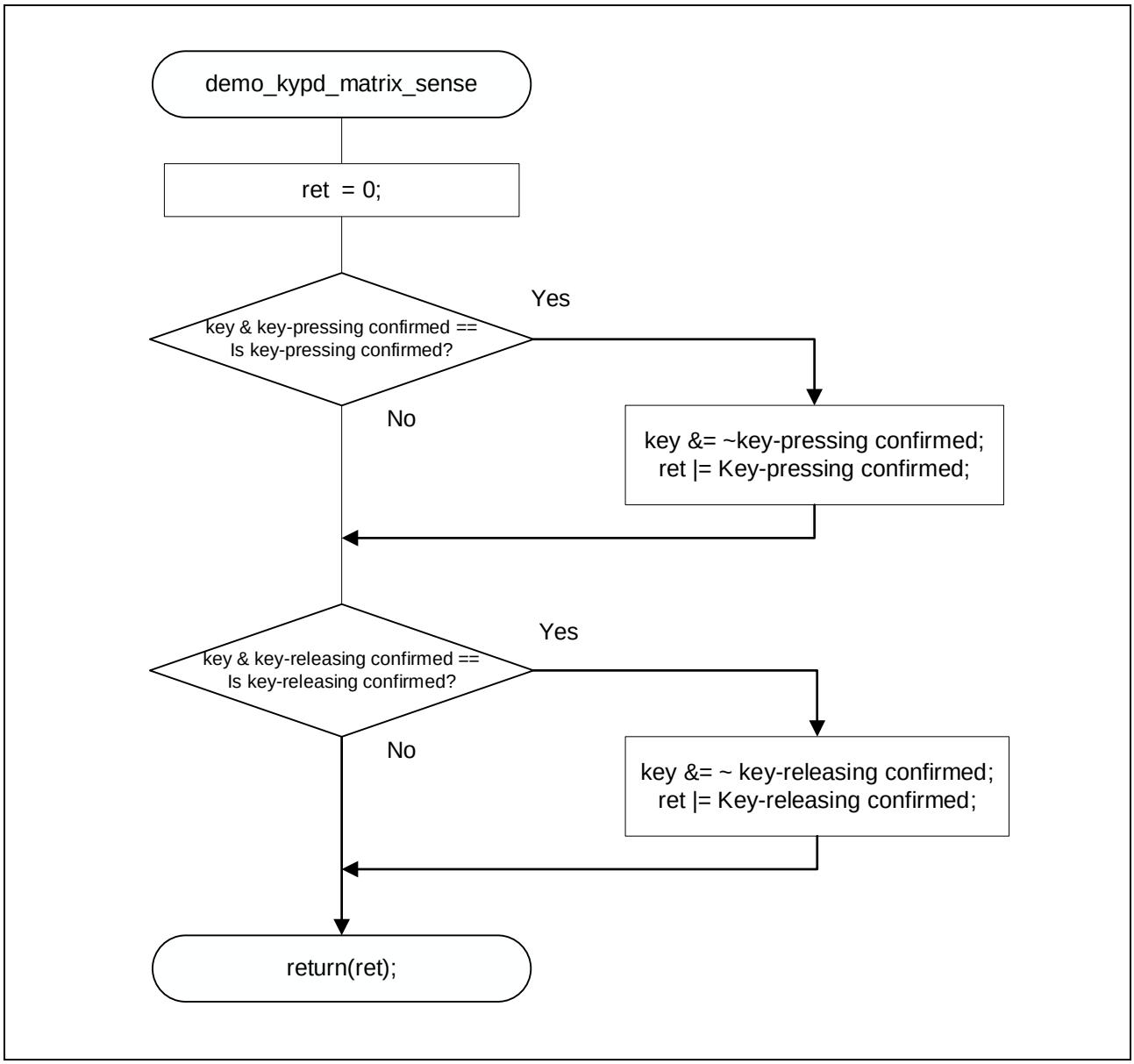



Figure 3-23 Key Input Determination Processing

### 3.9.17 Count processing for the key input confirmation timer

Figure 3-24 shows the flowchart for count processing for the key input confirmation timer.

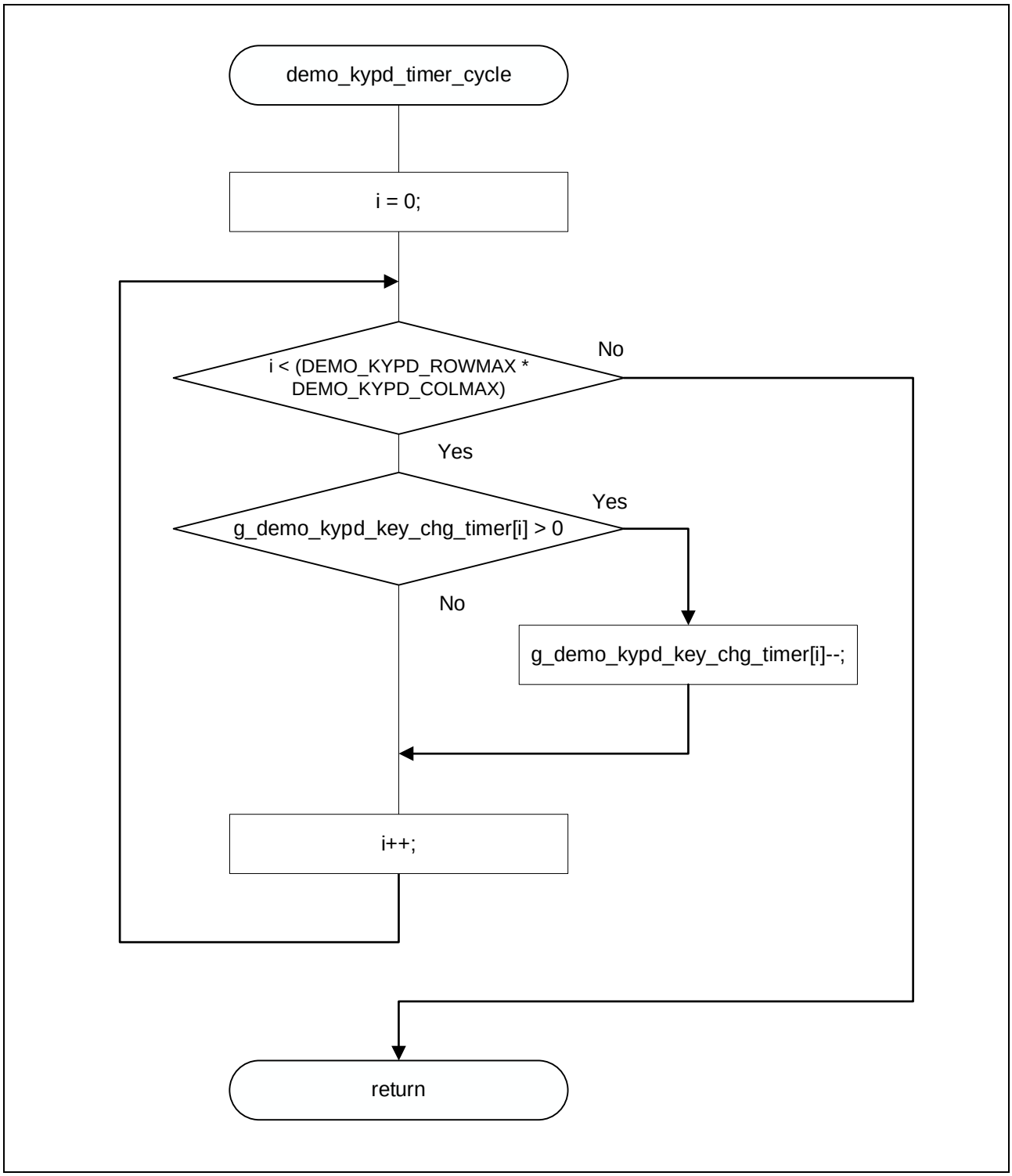



Figure 3-24 Count Processing for the Key Input Confirmation Timer

### 3.9.18 Processing to start the key input confirmation timer

Figure 3-25 shows the flowchart for processing to start the key input confirmation timer.

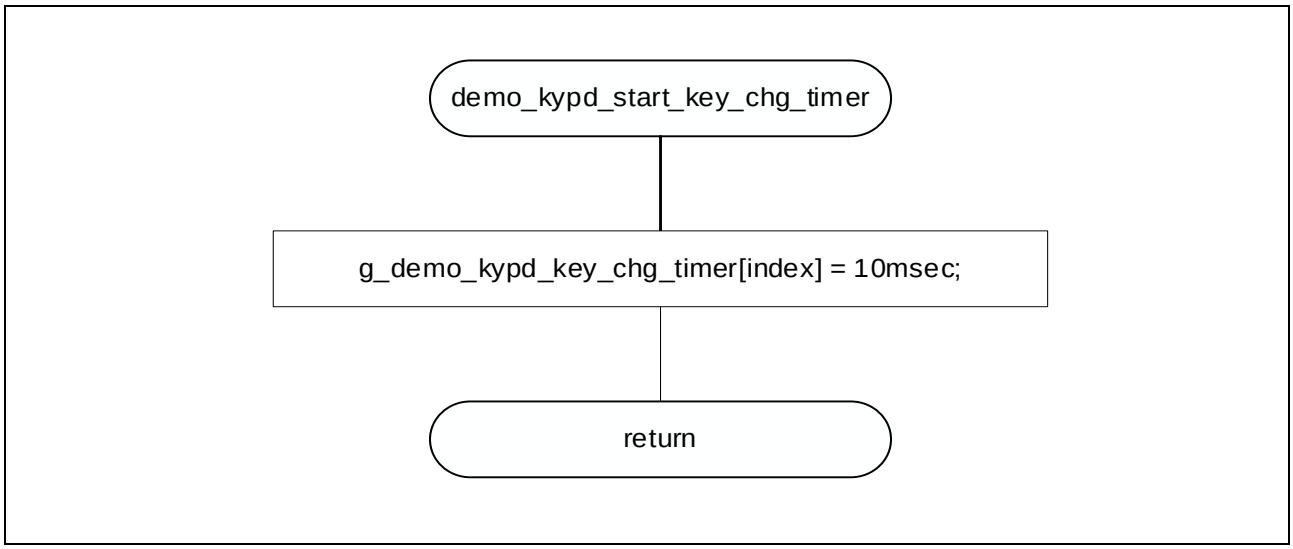



Figure 3-25 Processing to Start the Key Input Confirmation Timer

### 3.9.19 Processing to acquire the key input confirmation timer count

Figure 3-26 shows the flowchart for processing to acquire the key input confirmation timer count.

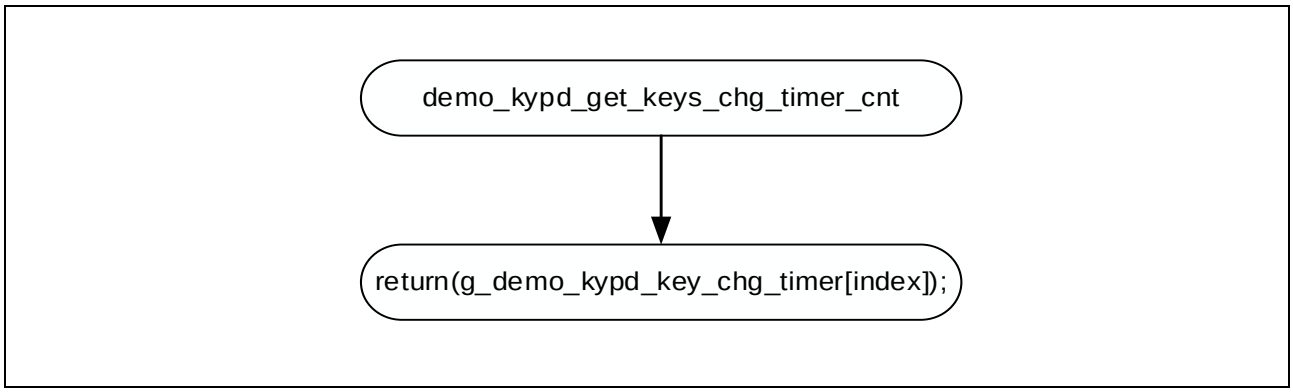



Figure 3-26 Processing to Acquire the Key Input Confirmation Timer Count

# 4. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

# 5. Notes

## 5.1 Operation of the Hexadecimal Keyboard

Due to the specifications of the hexadecimal keyboard used in this sample program, pressing two or more keys at the same time might result in unexpected value input. To prevent problems, press one key at a time.

# 6. Reference Documents

RL78 Family MIDI Interface Module Software Integration System (R01AN7265E)
RL78 Family MIDI Linked Illumination Control Sample Software Using SIS (R01AN7463E)
RL78/G16 User's Manual: Hardware (R01UH0980E)
RL78 family user's manual: software (R01US0015E)
RL78/G16 Fast Prototyping Board User's Manual (R12UM0048E)
(The latest versions can be downloaded from the Renesas Electronics website.)

Technical update
(The latest versions can be downloaded from the Renesas Electronics website.)

MIDI Shield (SparkFun MIDI Shield)

https://www.sparkfun.com/products/12898

LED matrix

https://wiki.52pi.com/index.php?title=EP-0075

Hexadecimal keyboard

https://digilent.com/reference/pmod/pmodkypd/start

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

## Revision History

| Rev. | Date | Description | |
|---|---|---|---|
| | | Page | Summary |
| 1.00 | 2024.12.04 | — | First Edition |
| | | | |

# 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. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time 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 time 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 time 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 time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. 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 the 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.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is 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. Additionally, 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.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between $V_{IL}$ (Max.) and $V_{IH}$ (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between $V_{IL}$ (Max.) and $V_{IH}$ (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of 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.

# Notice

1. 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 or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving 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, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, 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 and impractical, you are responsible for evaluating the safety of the final products or systems 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. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall 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. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

## Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

## Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.

## Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

© 2024 Renesas Electronics Corporation. All rights reserved.