RL78 Family

MIDI Linked Illumination 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 MIDI interface SIS module is used to control the LED matrix display in accordance with a MIDI melody (MIDI message) sent from a PC. It is also possible to transfer MIDI messages to the sound module to play music.

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. Specifications

1.1 Overview of Specifications

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

The receive function is used to control the LED matrix display in accordance with a MIDI melody (MIDI message) sent from a PC. MIDI messages can be transferred (passed through) from the MIDIOUT pin to the sound module to play music 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 P03/TxD0、P04/RxD0	UART communication with MIDI devices
A/D Converter P05/ANI4	Volume switch input for selecting display channels
Serial Array Unit CSI20 P13/SCK20、P16/GPIO、P15/SO20	SPI communication with the LED matrix module

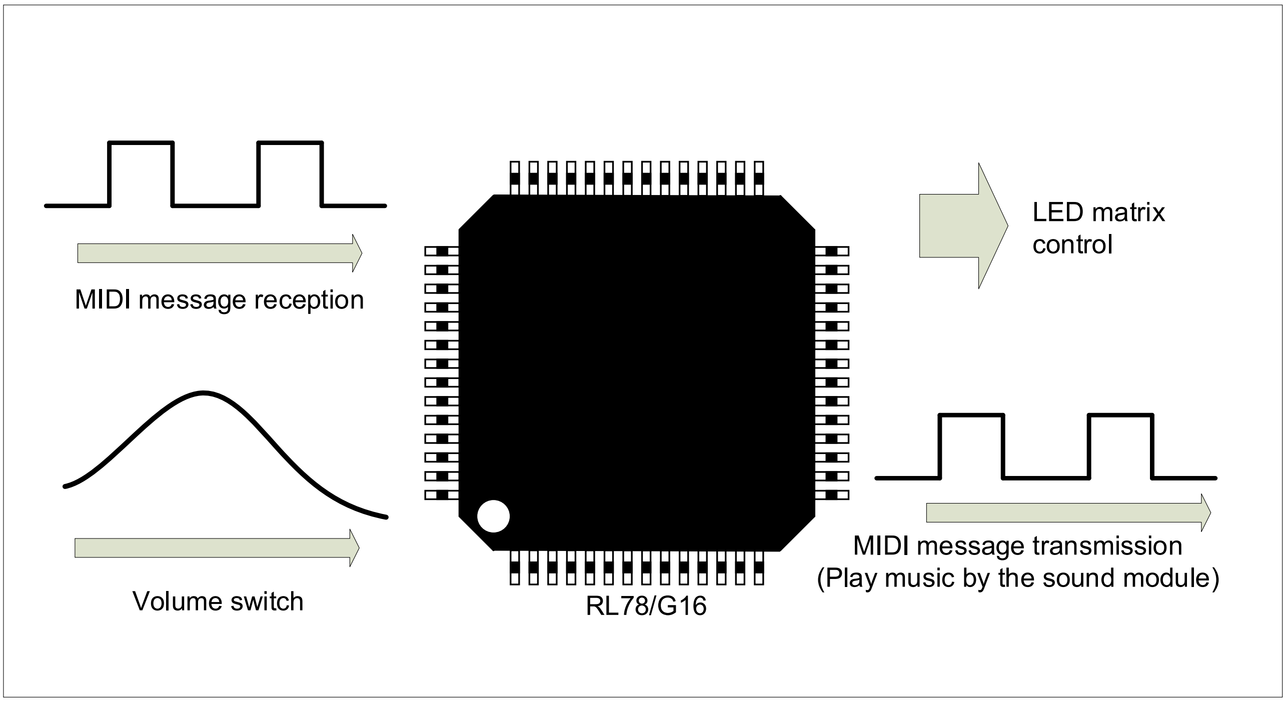

Figure 1-1 Overview of Sample Program Operation

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 K bit/second. Each byte to be sent consists of ten bits (one start bit, eight data bits, and one stop bit).

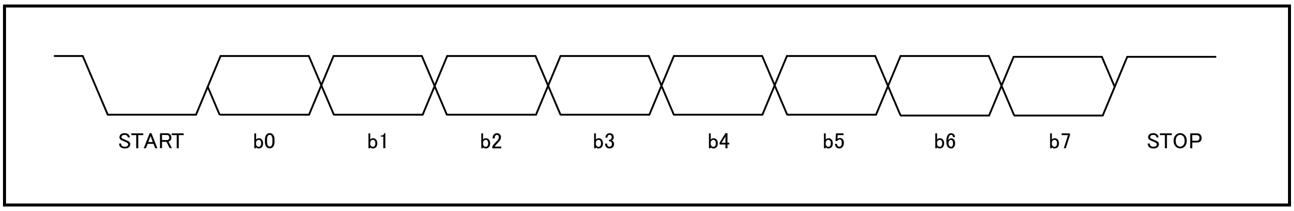

Figure 1-2 MIDI Data Column

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 a 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.

Status byte

Channel Message Structure

b7	b6	b5	b4	b3	b2	b1	b0
1	0	0	1	Channel#			

Data byte 1

b7	b6	b5	b4	b3	b2	b1	b0
0	Note#						

Data byte 2

b7	b6	b5	b4	b3	b2	b1	b0
0	Velocity						

Figure 1-4 NoteOn (Keystoke) Message

1.2 Operation Details

This sample program receives MIDI messages sent from the PC. When a NoteOn (keystroke) message is received, the LED matrix is turned on according to the message contents. Other MIDI messages including the NoteOn (keystroke) message are transferred to the sound module to play music.

The PC sends a MIDI message by reproducing any MIDI file from the DAW.

DAW: Digital Audio Workstation

Software that allows a wide range of production, including composition, recording, and mixing, to be performed on PCs

(1) Specifying and changing display target channels

- The MIDI standard defines 1 to 16 channels (denoting “chords”), which are referred to as MIDI channels.
- It is possible to display one specific MIDI channel or concurrently display all MIDI channels. (Hereafter, the mode to display one specific MIDI channel is referred to as “specific MIDI channel display mode”, and the mode to concurrently display all MIDI channels is referred to as “all MIDI channel concurrent display mode”.)
- A MIDI channel to be displayed is referred to as “display target channel”.
- These modes are determined according to the status of Volume(A0) of the MIDI shield.
- The user can change the display target channel by turning Volume(A0). (Channels can be changed even during replay of a melody.)
- When Volume(A0) is turned, information for the display target channel is displayed on the LED matrix for 1 second.
- If specific MIDI channel display mode is selected, “Ch” 01 to “Ch 16” are displayed.
- With “Ch 16” displayed, turning Volume(A0) enables selection of all MIDI channel concurrent display mode.
- When all MIDI channel concurrent display mode is selected, “Ch All” is displayed.
- Figure 1-5 MIDI Channel Displays shows how the display target channels are displayed on the LED matrix. (Black is off. Red and green are lit.)

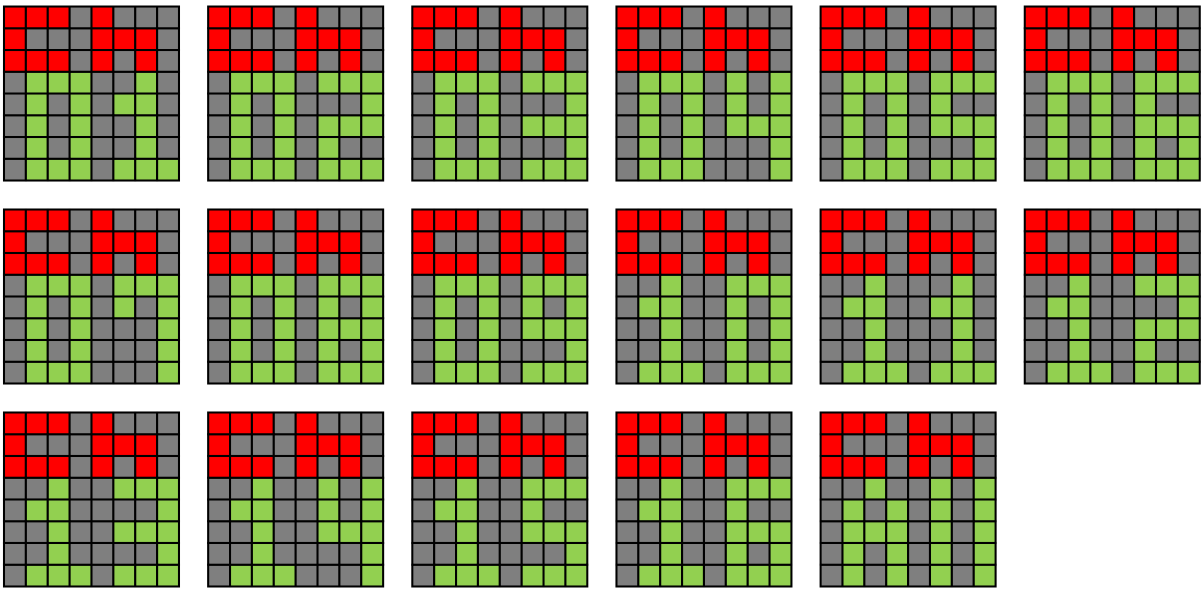

Figure 1-5 MIDI Channel Displays

(2) LED matrix display during melody display (specific MIDI channel display mode)

- The LED matrix is turned on based on the note, channel number, and velocity information contained in the NoteOn (keysroke) message among the received MIDI messages. That is, the LED matrix is turned on at the beginning of the sound.
- Colors are assigned to each note (do, re, mi, fa, so, la, ti) as shown in Figure 1-6. As an example, for piano, the received keystroke messages are shown in a single color for the white keys and in two colors for the black keys.

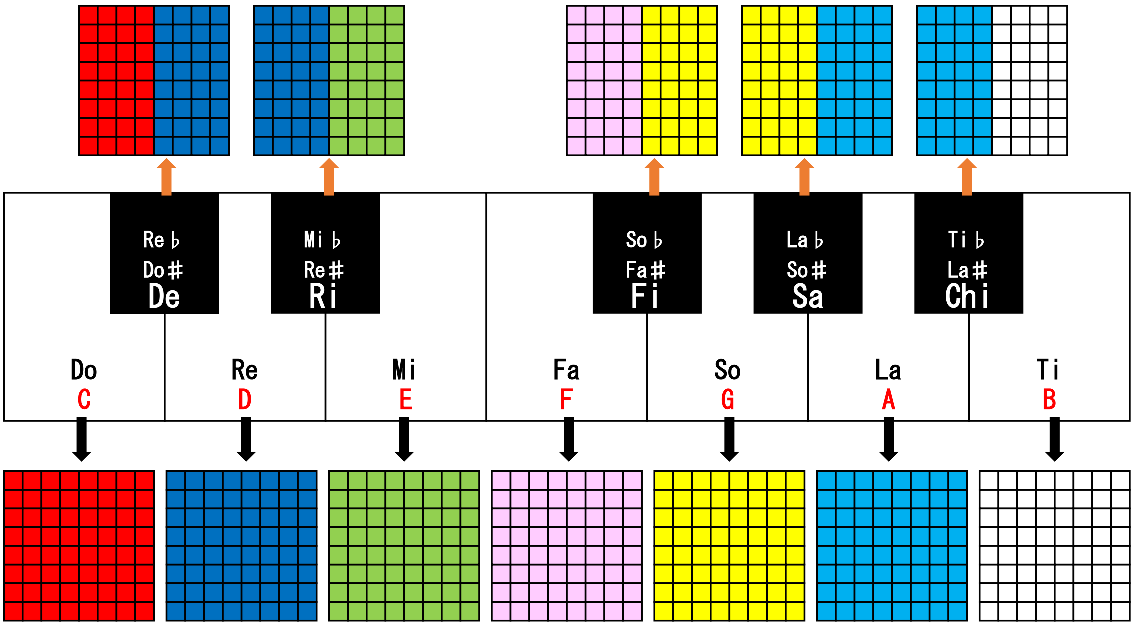

Figure 1-6 Display Color of Each Note (Specific MIDI Channel Display Mode)

- As shown in Figure 1-7, colors assigned to each note are displayed each time a NoteOn (keystroke) message of the display target channel is received. The display range is gradually reduced at regular intervals.
- When the display starts, the size is determined by referring to the velocity value contained in the NoteOn (keystroke) message.
  Figure 1-7, shows that the velocity value of the “Fa#” NoteOn message is the maximum value and therefore, the display starts from the maximum size. The “La#” NoteOn message contains a smaller velocity value, and therefore, the display starts from the value smaller by one frame size..
- Brightness adjustment is not performed, providing a constant brightness.

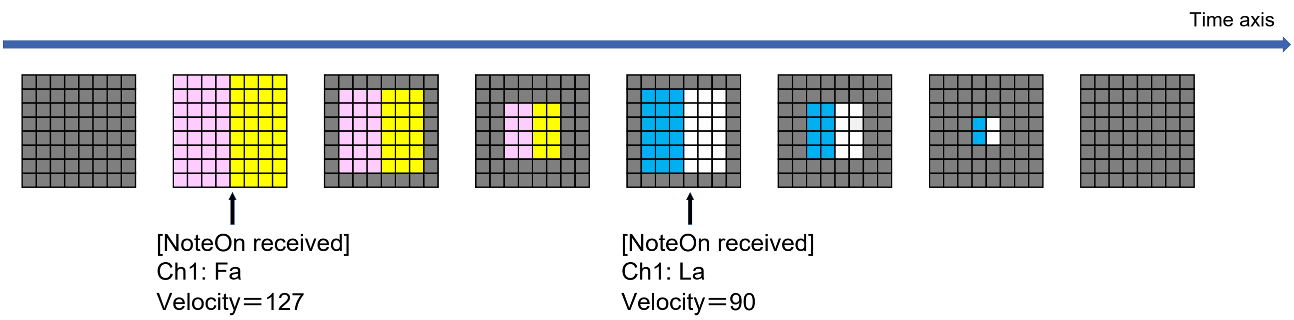

Figure 1-7 Display Example (Specific MIDI Channel Display Mode)

(3) LED matrix display during melody display (all MIDI channel concurrent display mode)

- The 8 x 8 LED matrix is divided into 16 blocks of 2 x 2 LED matrices, and then 16 MIDI channels are assigned.
- The relevant block of the LED matrix is turned on based on the note and channel number information contained in the NoteOn (keysroke) message among the received MIDI messages. The layout of the blocks based on the channel numbers are shown in Figure 1-8.

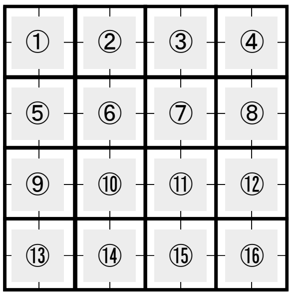

Figure 1-8 Display Location Assigned by MIDI Channel

- Color assignment by note of each channel shall be the same as that for specific MIDI channel display mode. The display frames are reduced as shown in Figure 1-9.

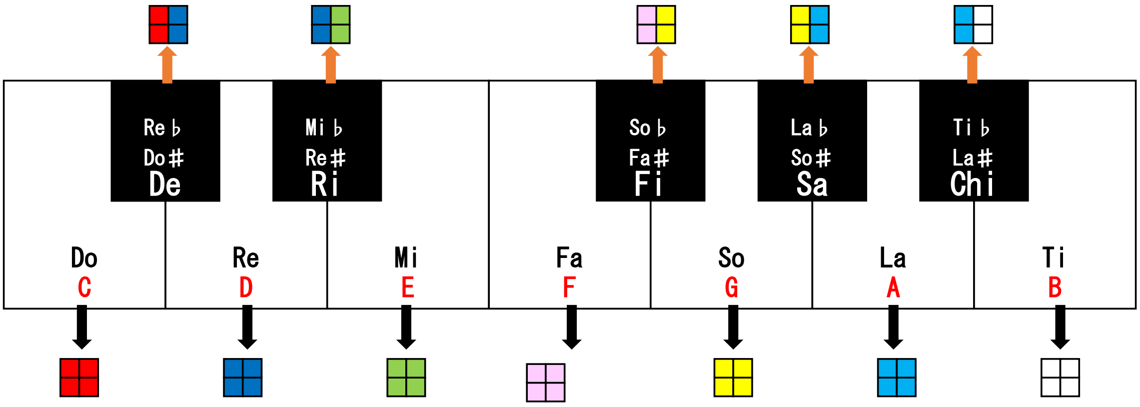

Figure 1-9 Display Color for Each Note (All MIDI Channels Concurrent Display Mode)

- As shown in Figure 1-10, each time a NoteOn (keystroke) message is received, the colors assigned by note are brightly lit in the LED matrix block assigned by MIDI channel. Then, the LED lights are gradually diminished.

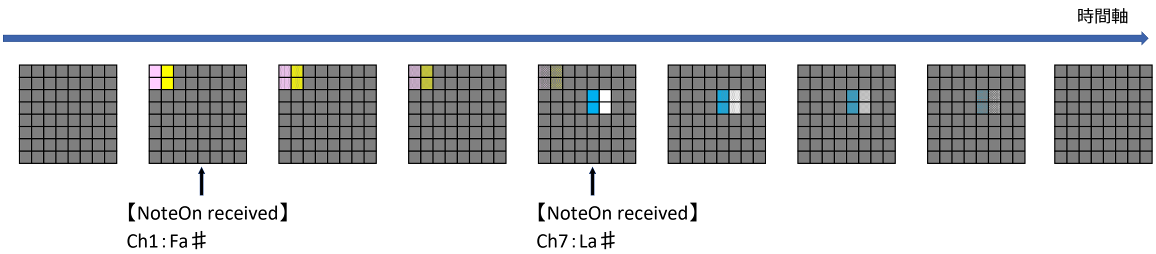

Figure 1-10 Display Example (All MIDI Channels Concurrent Display Mode)

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	Renesas Electronics RL78/G16 Fast Prototyping Board (RTK5RLG160C00000BJ)
MCU used	RL78/G16 (R5F121BCAFP)
Operating frequency	High-speed on-chip oscillator clock (fIH): 16 MHz
Operating voltage	5.0V
MIDI shield board	SparkFun MIDI Shield
MIDI to USB cable	Roland UM-ONE
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

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

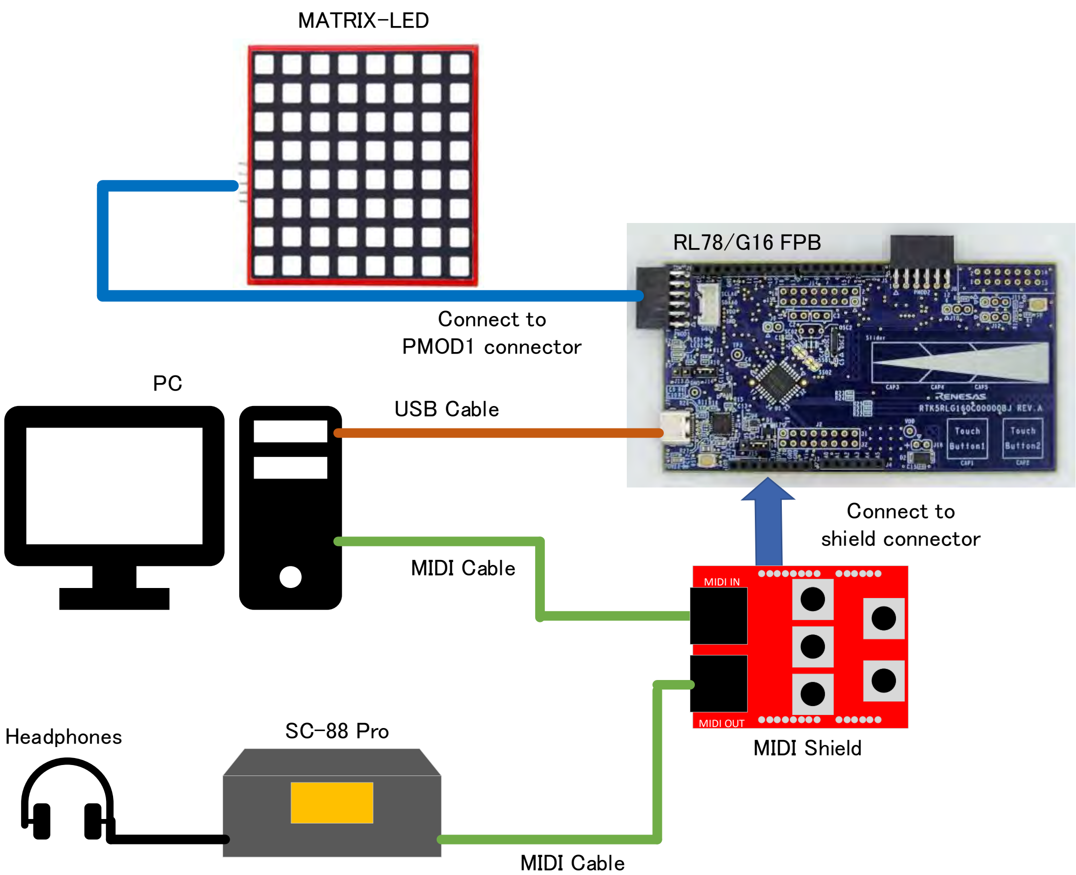

Figure 2-1 Hardware Configuration

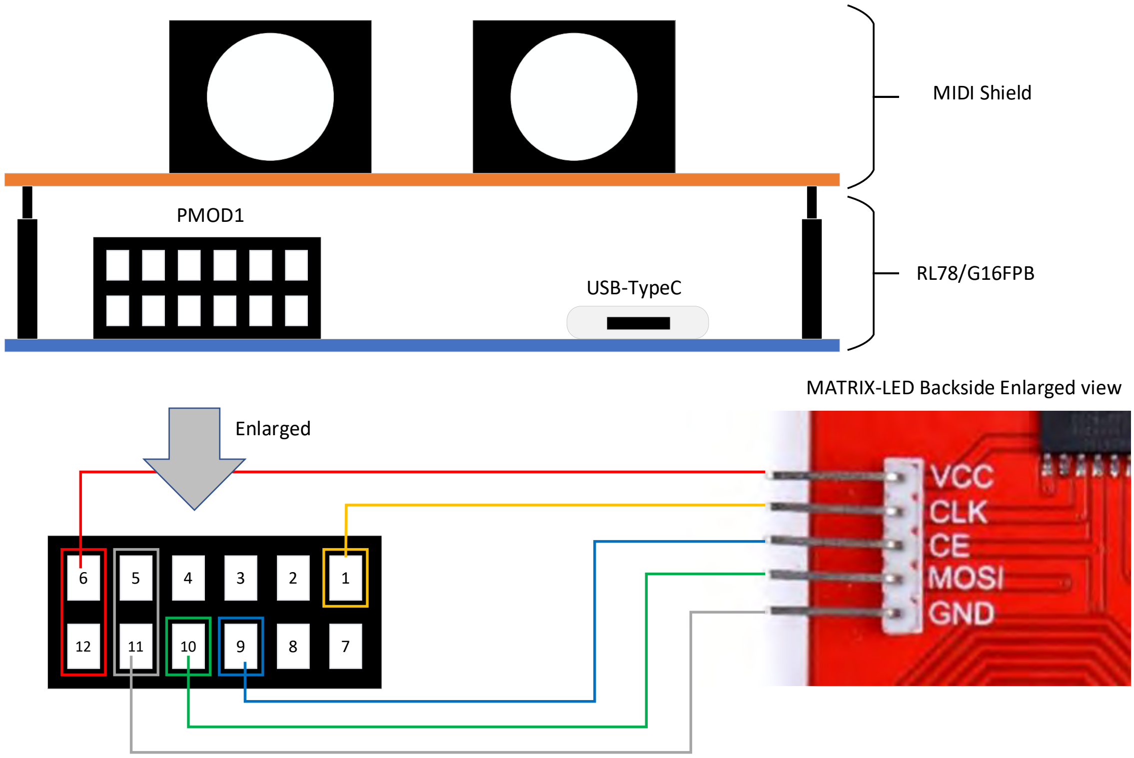

Figure 2-2 Wiring Between PMOD1 and MATRIX-LED

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.

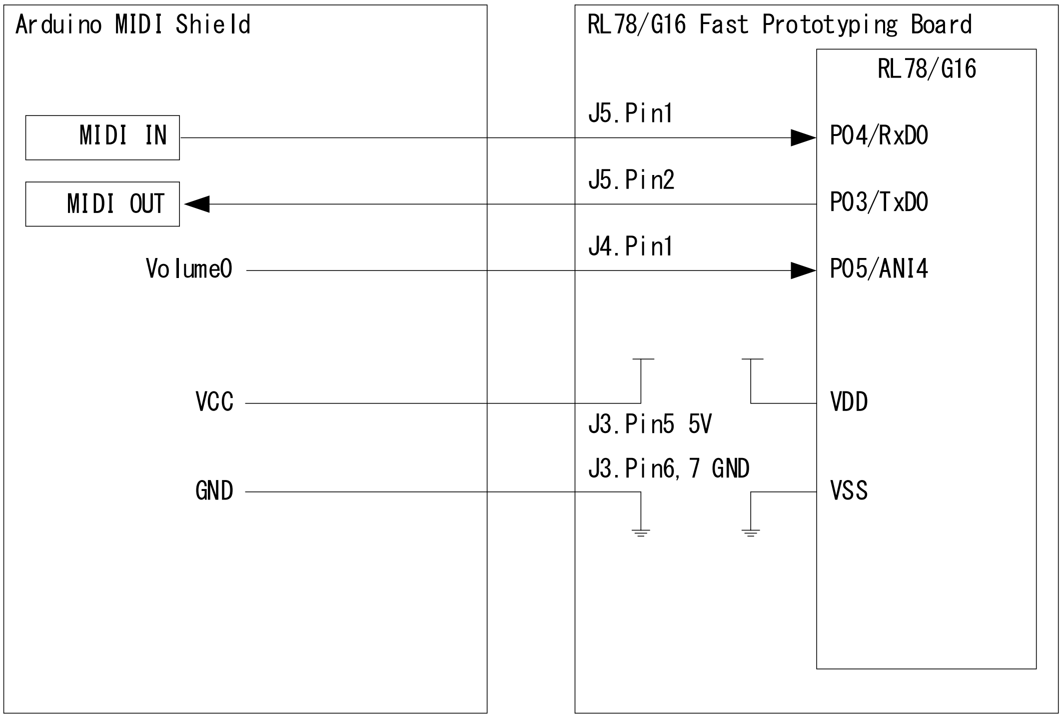

Figure 2-3 Pin Connection Diagram Between RL78/G16 FPB and MIDI Shield

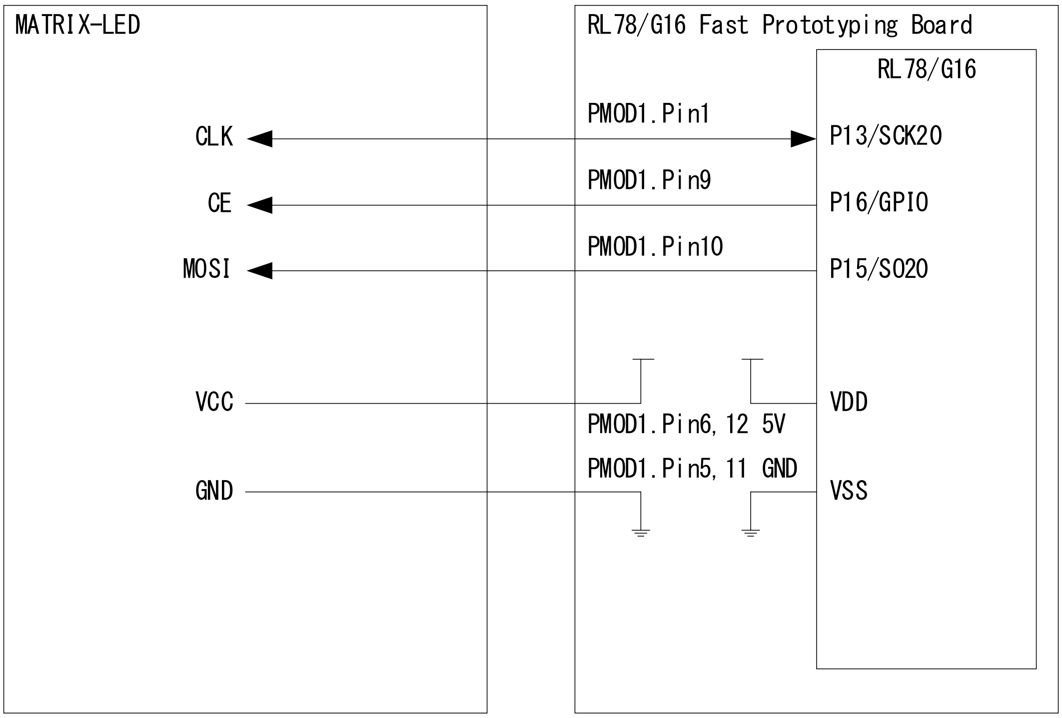

Figure 2-4 Pin Connection Diagram Between RL78/G16 FPB and MATRIX-LED

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
P04/RxD0	Input	MIDI message reception
P05/ANI4	Input	MIDI channel selection
P13/SCK20	Output	MATRIX-LED SPI clock
P16/GPIO	Output	MATRIX-LED SPI chip selection
P15/SO20	Output	MATRIX-LED SPI MOSI

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	Content
Integrated development environment	Manufactured by Renesas Electronics e^2 studio 2024-04
C compiler	Manufactured by Renesas Electronics C Compiler Package for RL78 Family [CC-RL] V1.13.00
Smart Configurator (SC)	Smart Configurator for RL78 V1.10.0
Board Support Package (BSP)	Manufactured by Renesas Electronics V1.62
SIS module	MIDI Interface Module

Note. Please use the latest version of the MIDI interface module.
Use the latest version by adding the MIDI interface module again.
Refer to the following document for the steps to add the module.

When adding in e2studio
RL78 Smart Configurator User's Guide: e^2 studio (R20AN0579)
When adding in IAR
RL78 Smart Configurator User's Guide: IAR (R20AN0581)

3.2 Peripheral Function Settings

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

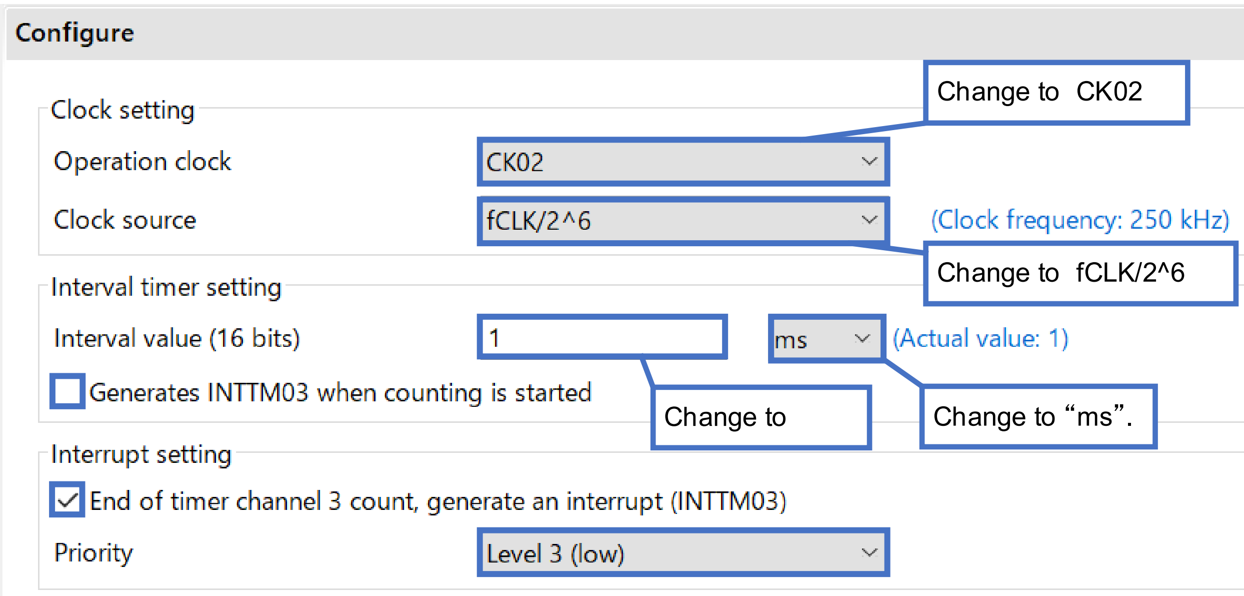

Figure 3-1 1-ms Interval Timer Settings

Figure 3-2 shows the settings of the 4-ms interval timer.

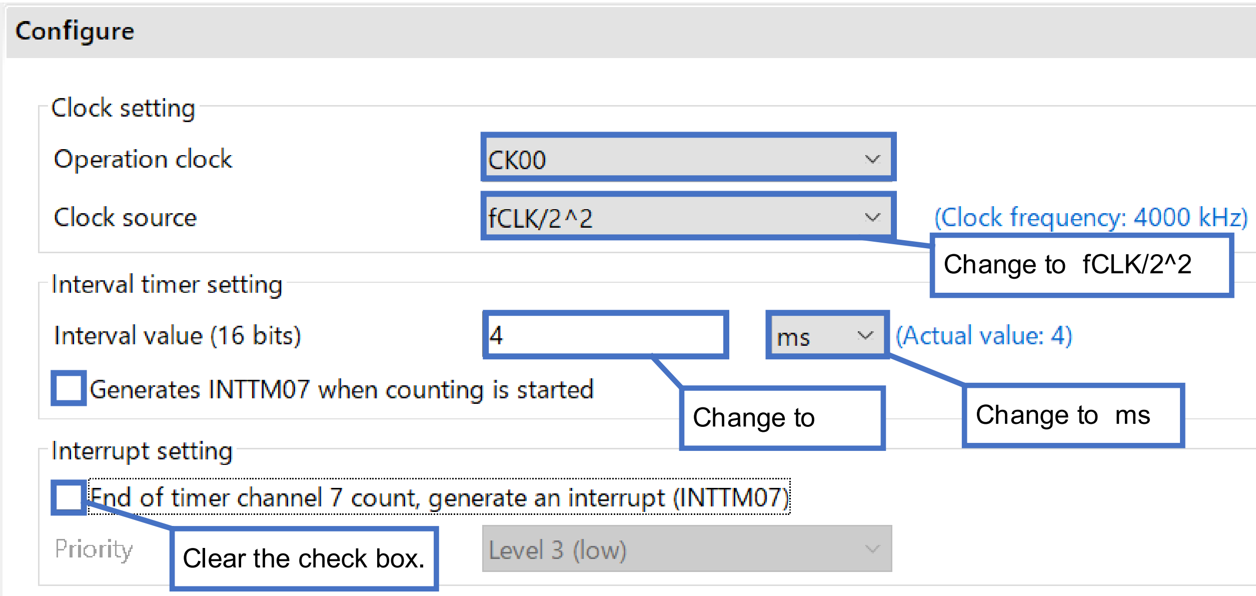

Figure 3-2 4-ms Interval Timer Settings

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

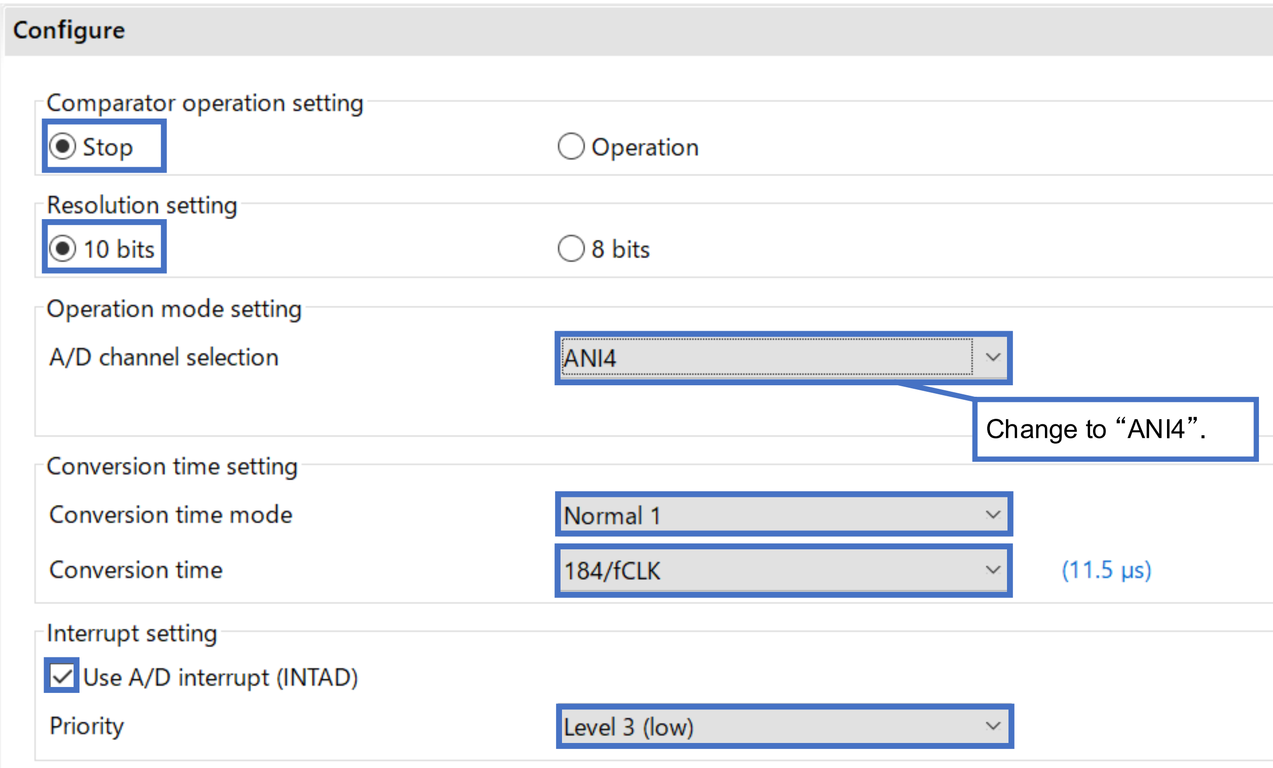

Figure 3-3 Analog Input Settings

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

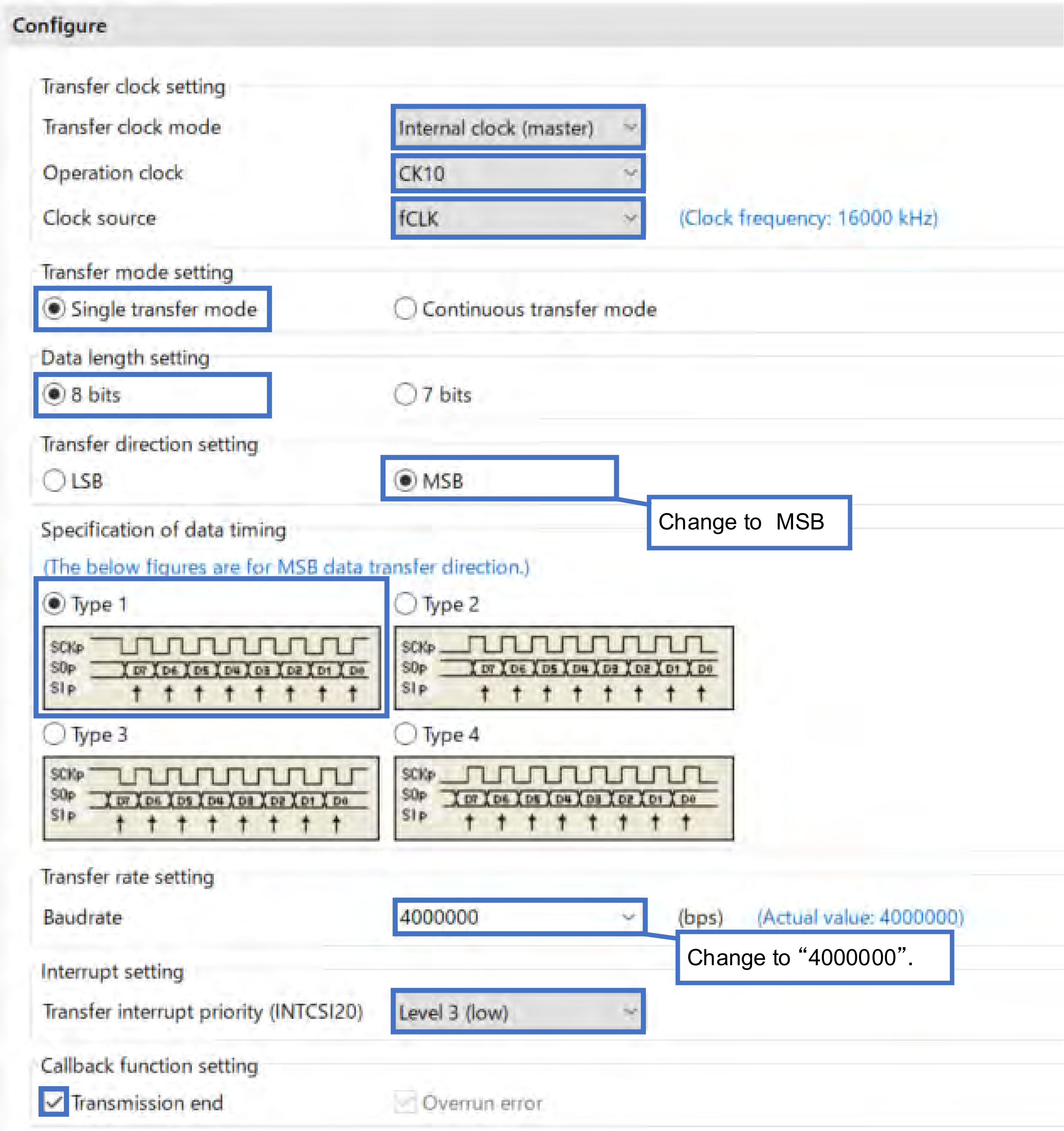

Figure 3-4 SPI Communication Settings

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

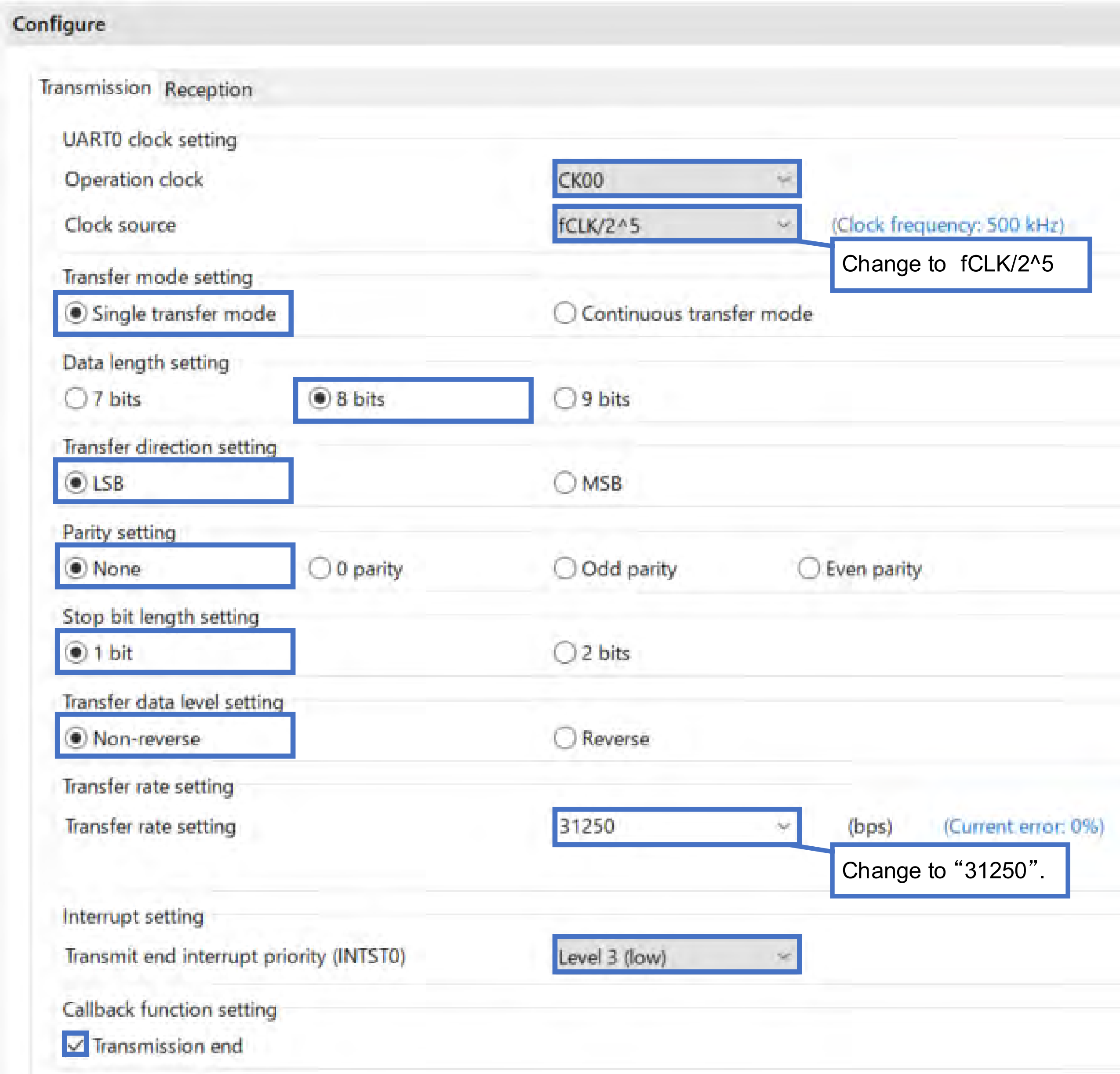

Figure 3-5 Communication Settings for MIDI Transmission

Figure 3-6 shows the communication settings for UART reception that comply with the MIDI communication standard.

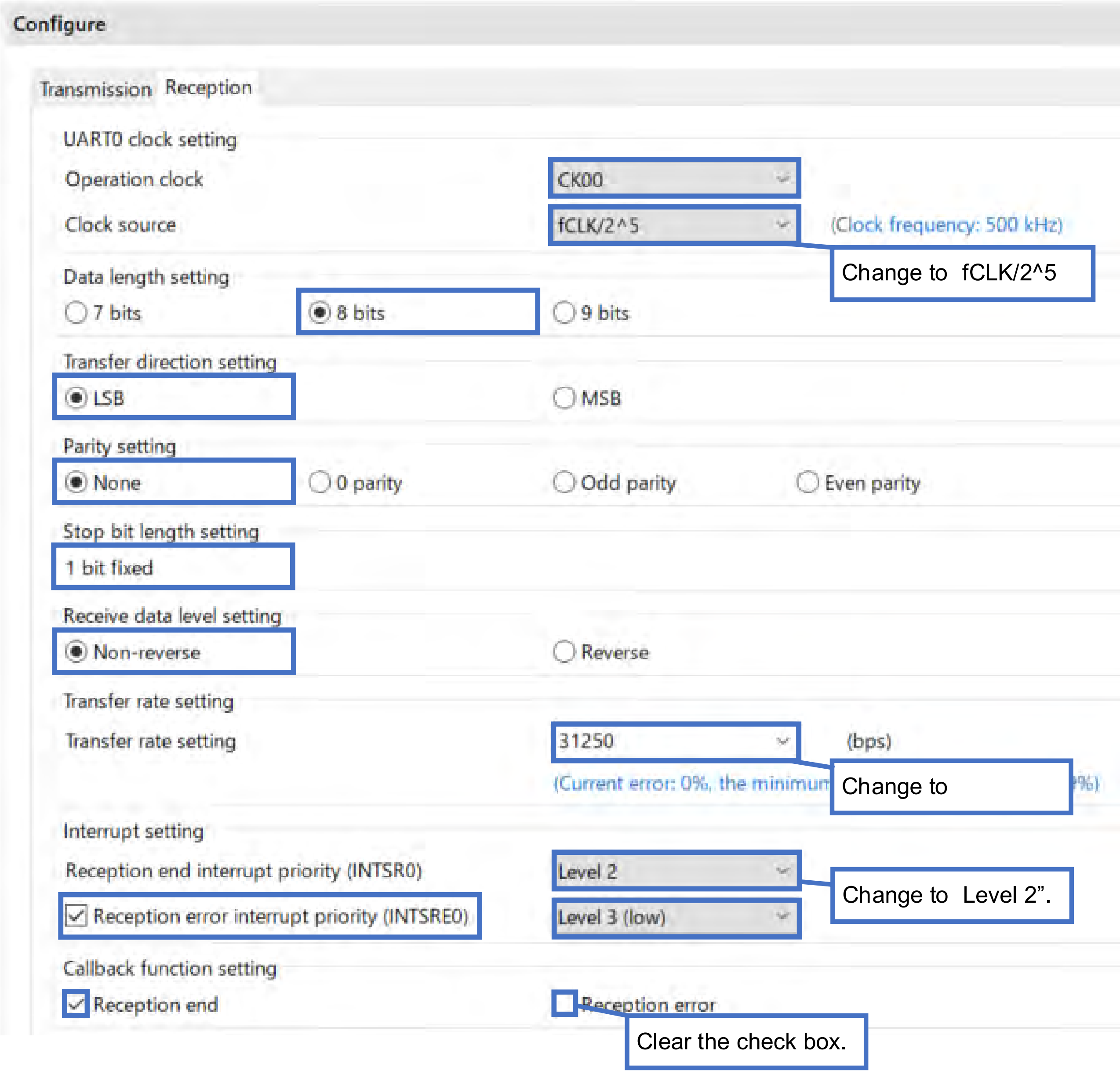

Figure 3-6 Communication Settings for MIDI Reception

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. (Counting stopped after reset)
000C1H	11110111B	SPOR operations (VSPOR) At rising edge TYP. 2.90V (2.76 V ～ 3.02 V) 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

Macro Name	Set Value	Description
DEMO_ALLCH_LIGHT_DIM	3	Number of LED dimming levels in all channel display mode
DEMO_MIDI_BRIGHTNESS_DIVISION	120	Resolution value used to calculate the LED dimming time
DEMO_MIDI_PITCH_NUM	12	Number of notes in one octave
DEMO_MIDI_DISPLAY_ALLCH	16	Number of MIDI channels displayed in all channel display mode
DEMO_MIDI_CH_MAX	16	Maximum MIDI channel number
DEMO_MIDI_CH_MIN	1	Minimum MIDI channel number
DEMO_MIDI_DISPLAY_CH_MAX	17	Number of channels (Ch1 to Ch16, or all channels)
DEMO_ALLCH_LIGHT_KEEP_THD	40	Time threshold by which the brightness is retained during LED dimming in all channel display mode
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
DEMO_DISPLAY_CH_SET_TIME	1000	Channel 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	5	Number of volume levels: 5
DEMO_SOUND_VOLUME_PAT_LARGE	4	Sound volume type: Large
DEMO_SOUND_VOLUME_PAT_MEDIUM	3	Sound volume type: Medium
DEMO_SOUND_VOLUME_PAT_SMALL	2	Sound volume type: Small
DEMO_SOUND_VOLUME_PAT_MICRO	1	Sound volume type: Micro
DEMO_SOUND_VOLUME_PAT_NONE	0	Sound volume type: None
DEMO_SYSTEM_TIMER_START_FUNC	-	Alias of R_Config_TAU0_3_Start
DEMO_MATRIX_LED_PWM_START_FUNC	-	Alias of R_Config_TAU0_7_DEMO_MATRIX_LED_PWM_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_PWM_PERIOD_REG	-	Alias of the timer data register TDR07
DEMO_PWM_COUNT_REG	-	Alias of the timer count register TCR07
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

3.5 List of Constants

Table 3-4 lists the constants.

Table 3-4 Constants

Type	Constant Name	Description	Used Functions
const uint8_t	g_color_table_single_ch	Display color set for each RDB in display mode for each MIDI channel	demo_display_main
const uint8_t	g_sound_volume_digit_table	Sound volume setting table	demo_display_main
const uint8_t	g_sound_volume_msk	LED mask table by sound volume in specific channel display mode	demo_display_main
const uint8_t	g_disp_volume_start	Velocity-to-volume conversion table	demo_convert_velocity_to_graph_pattern
const uint8_t	g_anti_blur	Anti-blur LED data	demo_display_main
const uint8_t	g_color_table_all_ch	LED data table by note in all channel display mode	demo_display_main
const uint8_t	g_display_ch_graph	LED data table to display channels when changing channels	demo_display_main

3.6 List of Variables

Table 3-5 and Table 3-6 list global variables.

Table 3-5 Global Variables (1/2)

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
ch_blink_t	g_demo_matrix_ch_info	Display information by note	main、demo_display_main demo_convert_velocity_to_graph_pattern
uint16_t	g_demo_pwm_period	PWM timer period	Main
uint16_t	g_demo_pwm_period_sub	Duty cycle reduction	Main
uint16_t	g_demo_pwm_period_base	Number of brightness levels of display	main
uint16_t	g_demo_pwm_count	PWM timer count value	main、demo_display_main
uint16_t	g_demo_pwm_count_bak	Previous count value of the PWM timer	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 3-6 Global Variables (2/2)

Type	Variable Name	Description	Used Functions
uint32_t	g_demo_display_ch_start_time	Channel display start time when changing channels	main、 demo_display_main
uint8_t	g_demo_display_mode	LED display mode: DEMO_DISPLAY_MODE_MELODY = Note display, or DEMO_DISPLAY_MODE_CH_SET = Channel display	main、 demo_display_main
uint8_t	g_demo_display_ch	Display target channel: 0 to 15 = Ch1 to Ch16, 16 = All channels	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_read_ch	MIDI channel from the NoteOn (keystroke) message: 0 to 15 = Ch1 to Ch16	Main
uint8_t	g_demo_read_pitch	Note from the NoteOn (keystroke) message (octave information is deleted)	Main
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

3.7 List of Functions

Table 3-7 lists the functions.

Table 3-7 Functions

Function Name	Overview
demo_volume_monitor_main()	Monitor channel 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_convert_velocity_to_graph_pattern()	Conversion processing from velocity to LED display pattern size
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_TAU0_7_DEMO_MATRIX_LED_ PWM_Start()	TAU0_7 timer start processing
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

3.8 Function Specifications

This section describes the function specifications of the sample program.

demo_volume_monitor_main()	
Overview	MIDI channel selection processing
Header	-
Declaration	uint8_t demo_volume_monitor_main(void);
Description	This function updates the MIDI channel to be monitored according to the change in the volume switch.
Arguments	None
Return values	0: The MIDI channel to be monitored is not changed. 1: The MIDI channel to be monitored 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. 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_convert_velocity_to_graph_pattern()	
Overview	Conversion from velocity to LED display pattern size
Header	-
Declaration	void demo_convert_velocity_to_graph_pattern(uint8_t ch);
Description	This function references the ever-changing velocity value retained by each MIDI channel and determines the display size of the LED matrix in specific MIDI channel display mode.
Arguments	uint8_t ch: MIDI channel
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 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 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. This function calls a MIDI 1-ms interval notification function. It also calls a system timer count function.
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. This function reads the AD conversion results and then saves them in the buffer (g_demo_adc_data). 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. This function sets the SPI CS pin to High. 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. 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 reception 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_receiveend(void);
Description	This function is called to perform callback processing when UART0 reception ends. This function calls (R_MIDI_NotifyEvent) to notify the MIDI interface SIS (Software Integration System) module of the end of reception.
Arguments	None
Return values	None

3.9 Flowcharts

3.9.1 Main processing

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

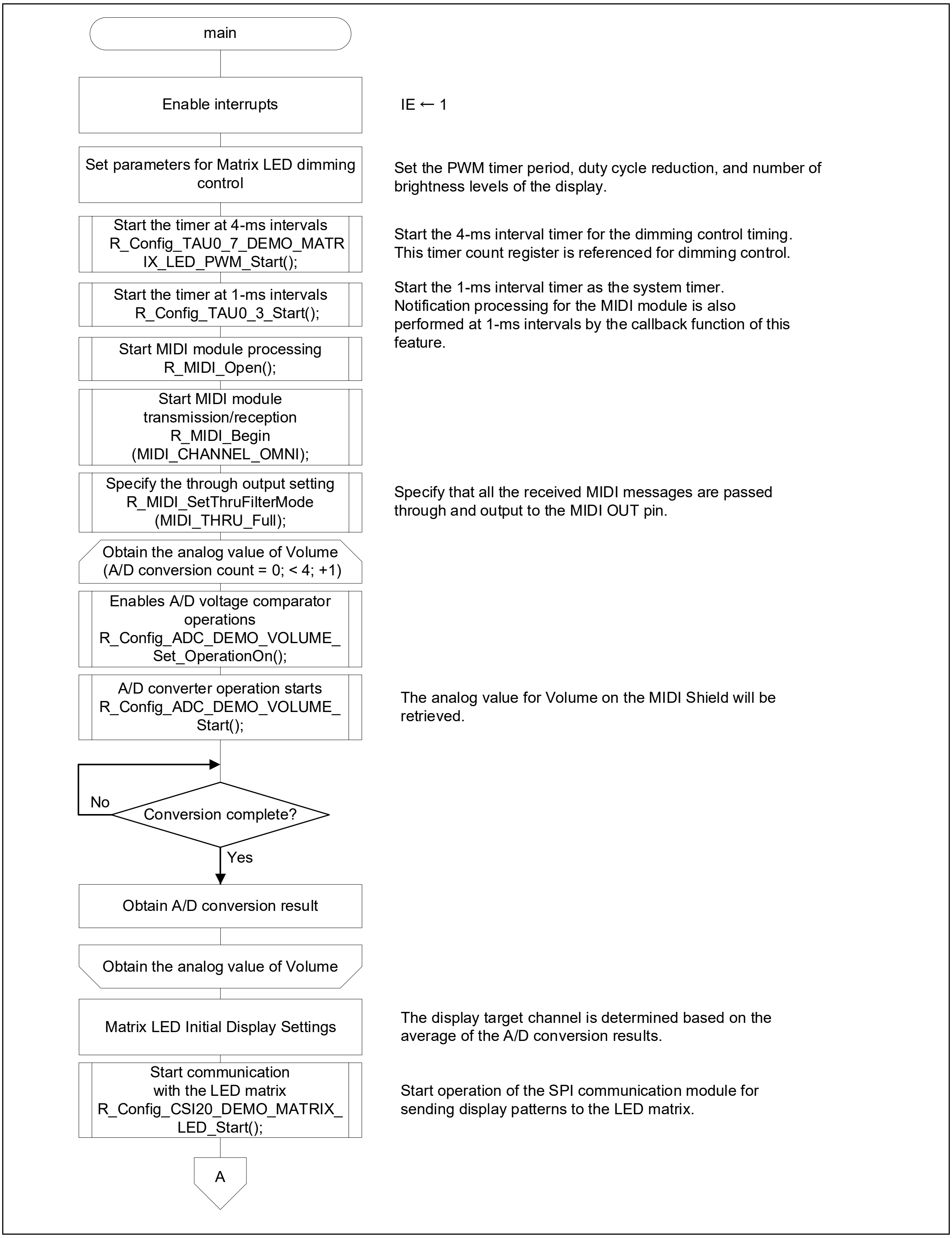

Figure 3-7 Main Processing (1/2)

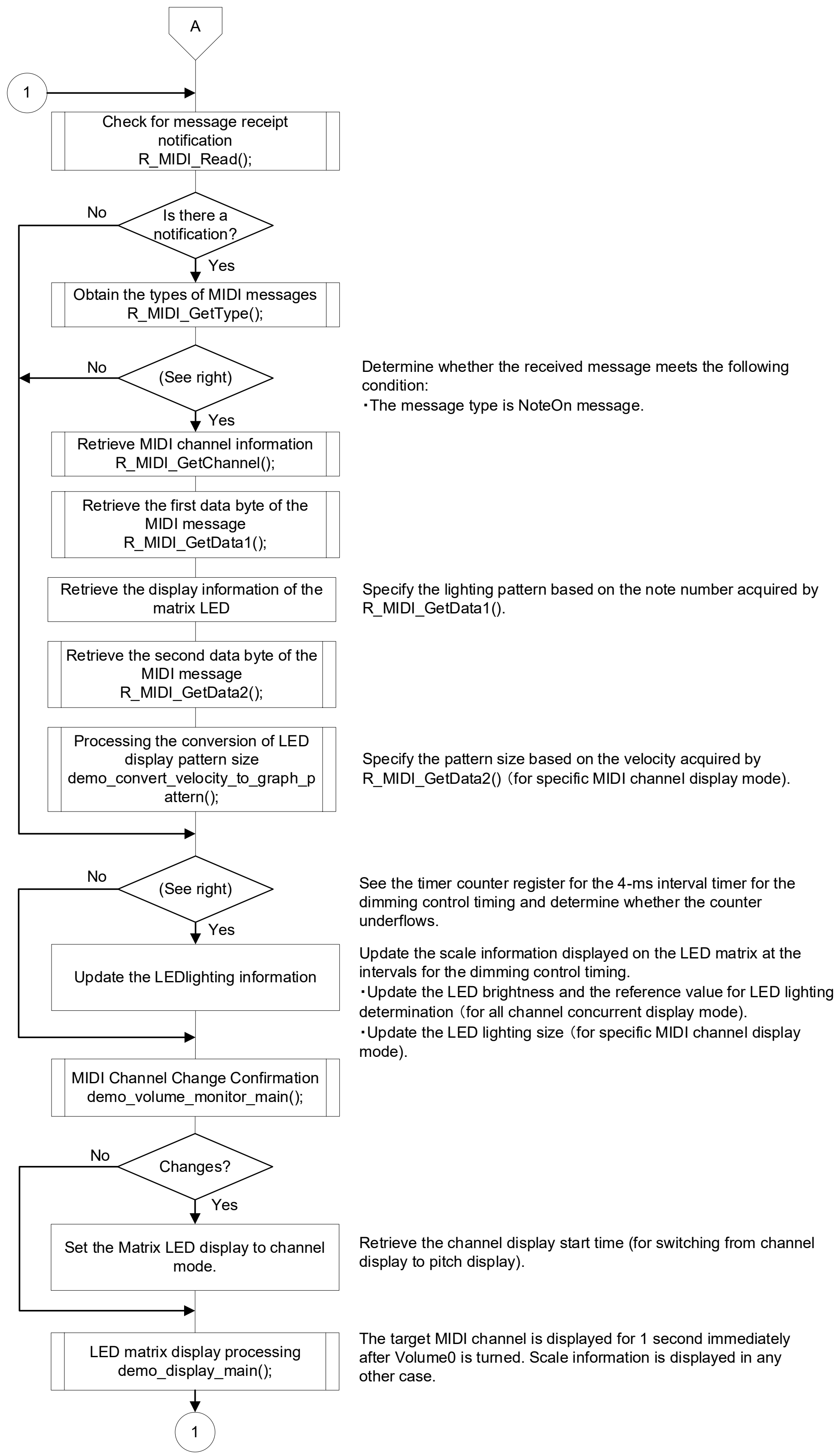

Figure 3-8 Main Processing (2/2)

3.9.2 Processing to determine the display target MIDI channel from the volume switch

Figure 3-9 shows the flowchart for processing to determine the display target MIDI channel from the volume switch.

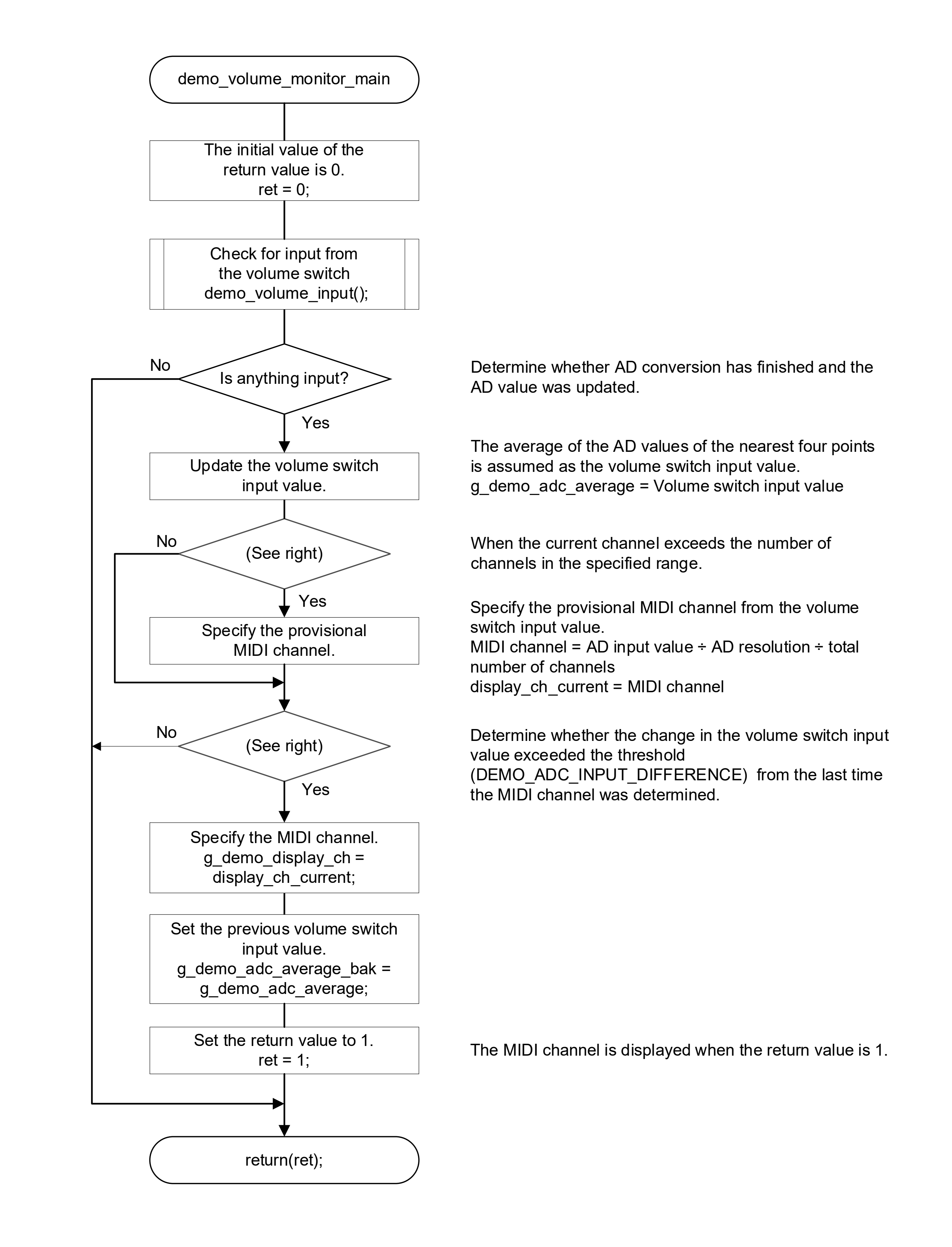

Figure 3-9 Processing to Determine the Display Target MIDI Channel from the Volume Switch

3.9.3 Processing to acquire volume switch input

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

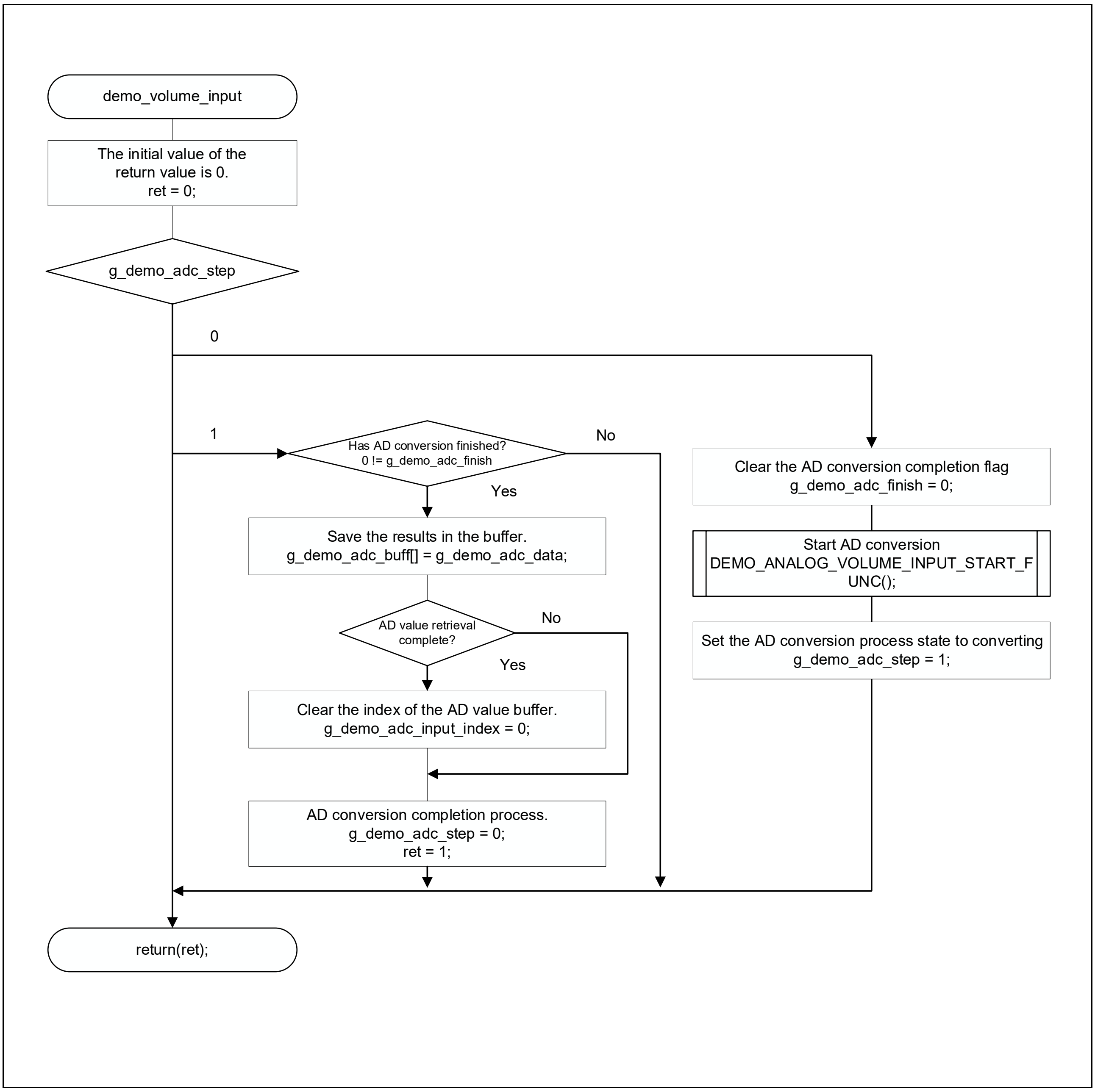

Figure 3-10 Processing to Acquire Volume Switch Input

3.9.4 LED matrix display processing

Figure 3-11 and Figure 3-12 shows the flowchart for LED matrix display processing.

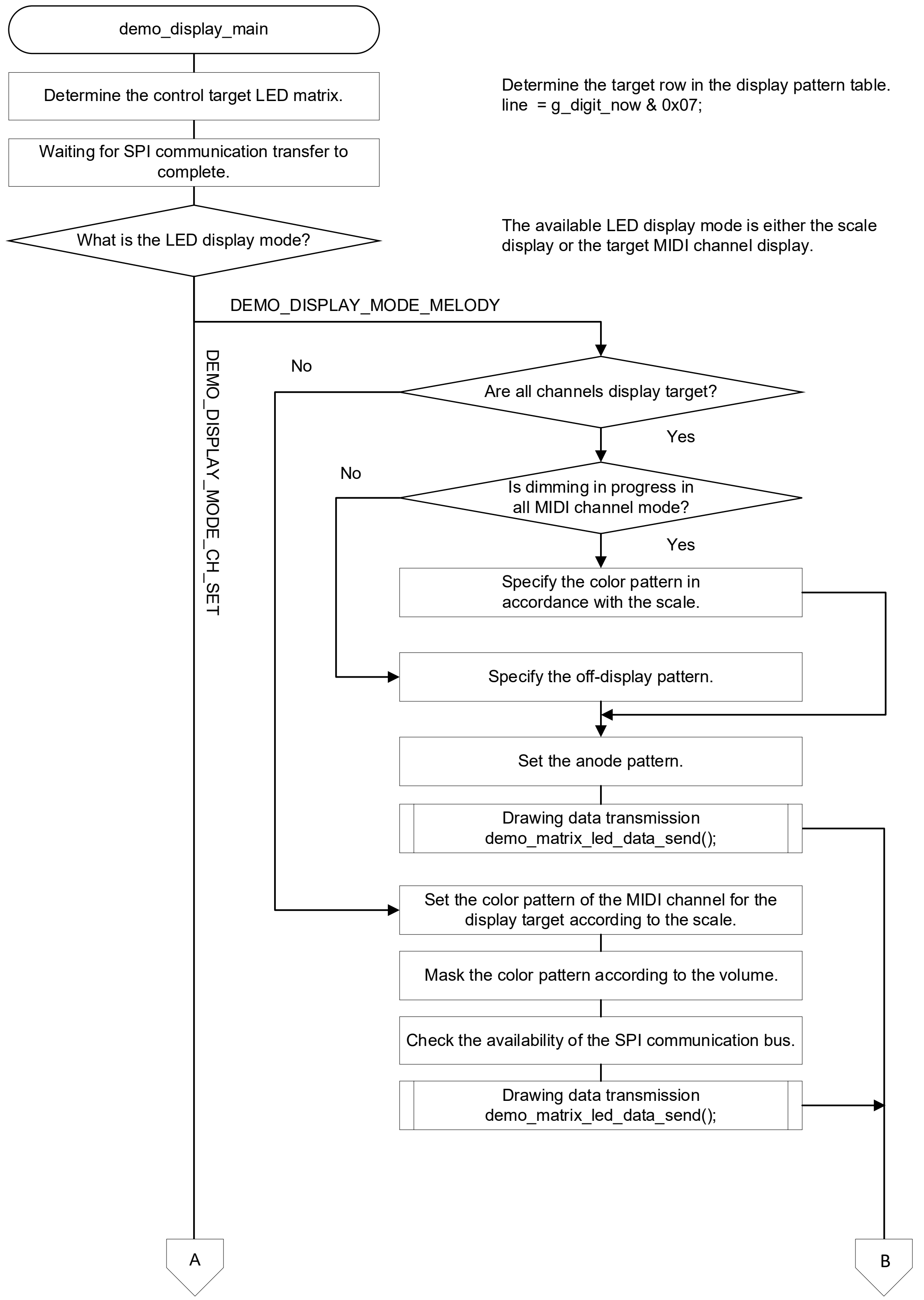

Figure 3-11 LED Matrix Display Processing (1/2)

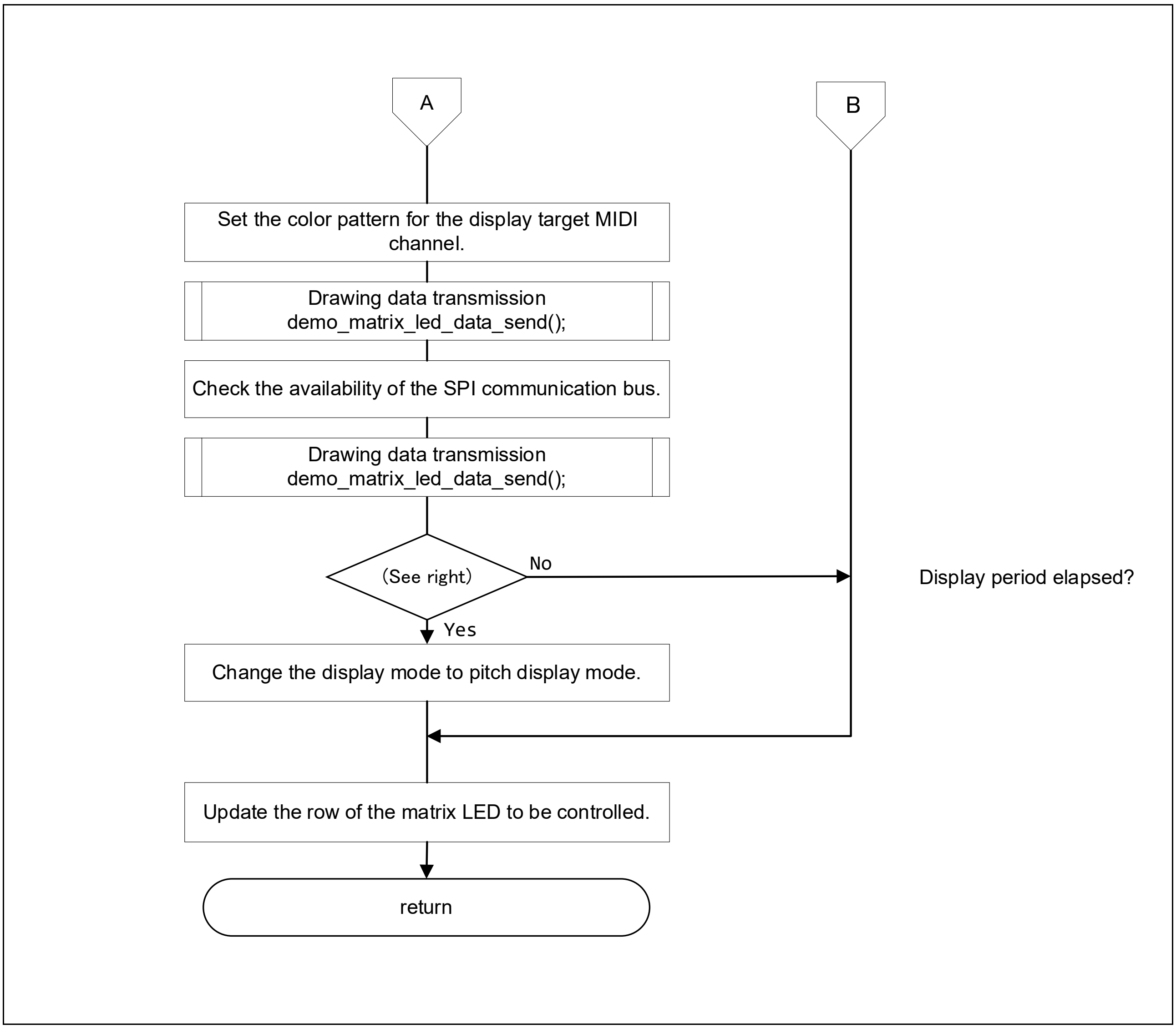

Figure 3-12 LED Matrix Display Processing (2/2)

3.9.5 System timer acquisition processing

Figure 3-13 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.

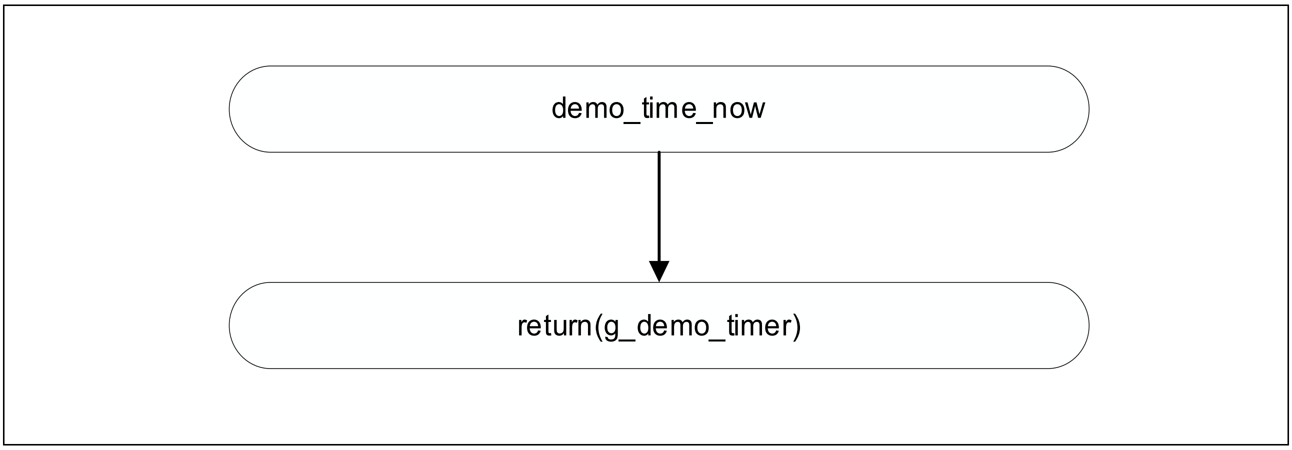

Figure 3-13 System Timer Acquisition Processing

3.9.6 System timer count processing

Figure 3-14 shows the flowchart for system timer count processing.

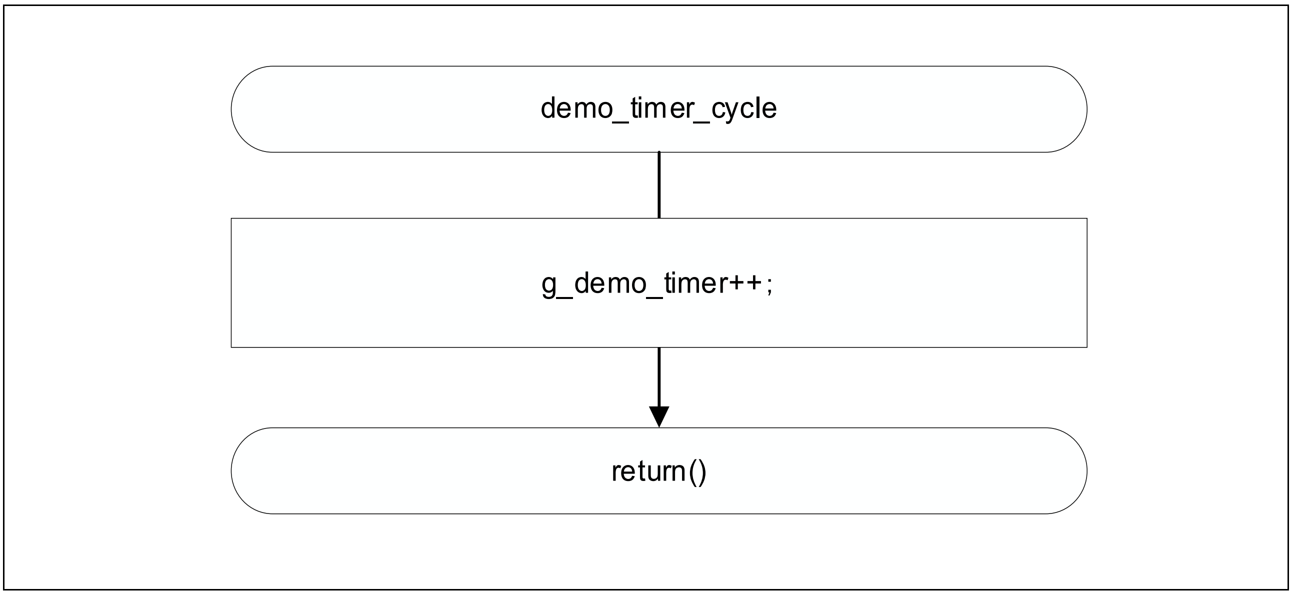

Figure 3-14 System Timer Count Processing

3.9.7 Processing to determine the display size of LED matrix data

Figure 3-15 shows the flowchart for processing to determine the display size of LED matrix data.

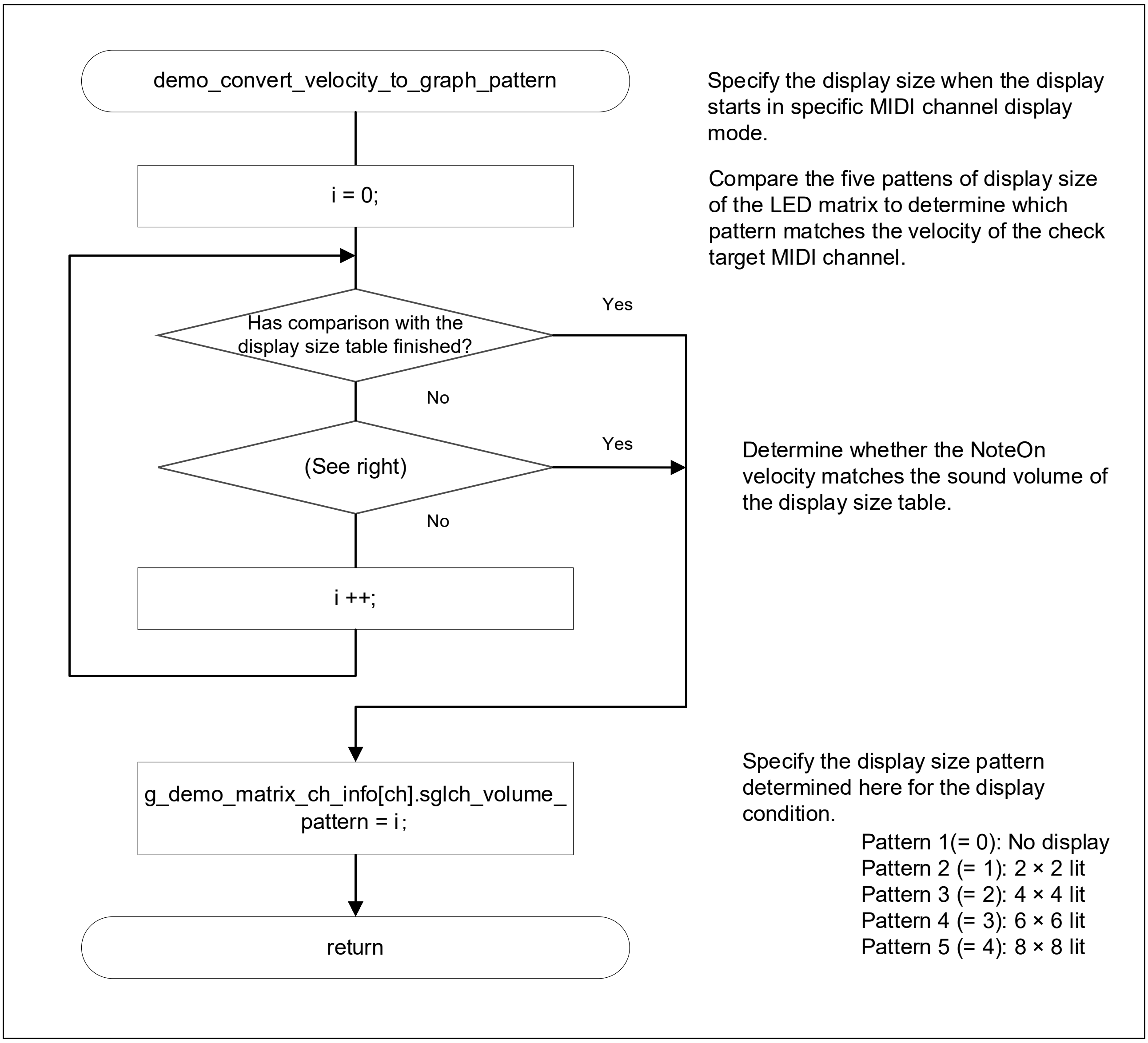

Figure 3-15 Processing to Determine the Display Size of LED Matrix Data

3.9.8 LED matrix data transmission processing

Figure 3-16 shows the flowchart for LED matrix data transmission processing.

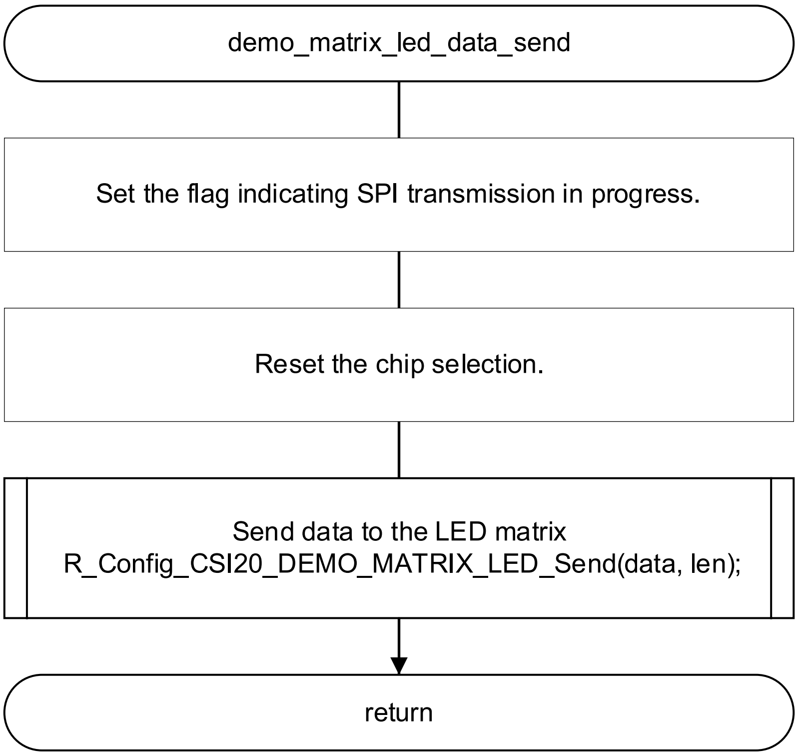

Figure 3-16 LED Matrix Data Transmission Processing

3.9.9 Processing to check the end of LED matrix data transmission

Figure 3-17 shows the flowchart for processing to check the end of LED matrix data transmission.

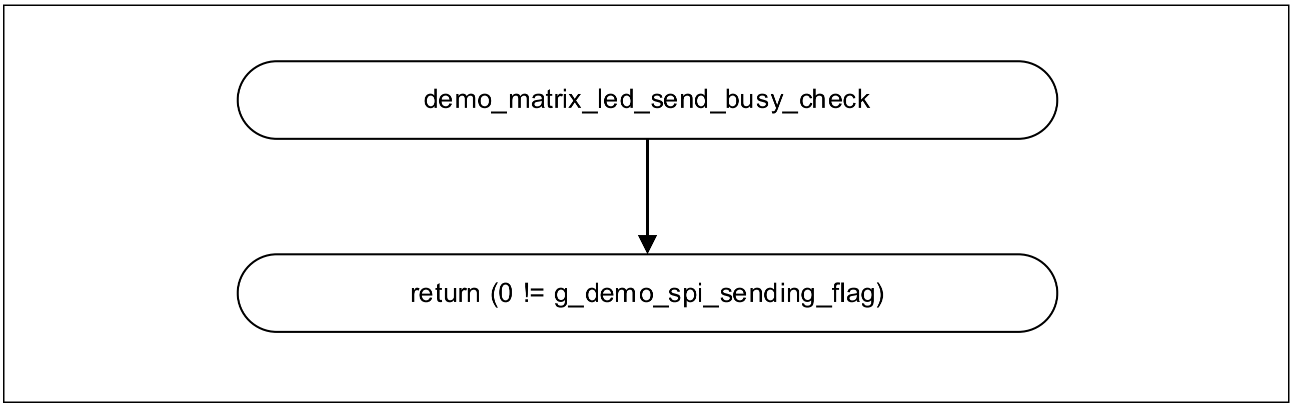

Figure 3-17 Processing to Check the End of LED Matrix Data Transmission

3.9.10 TAU0_3 interrupt processing

Figure 3-18 shows the flowchart for TAU0_3 interrupt processing.

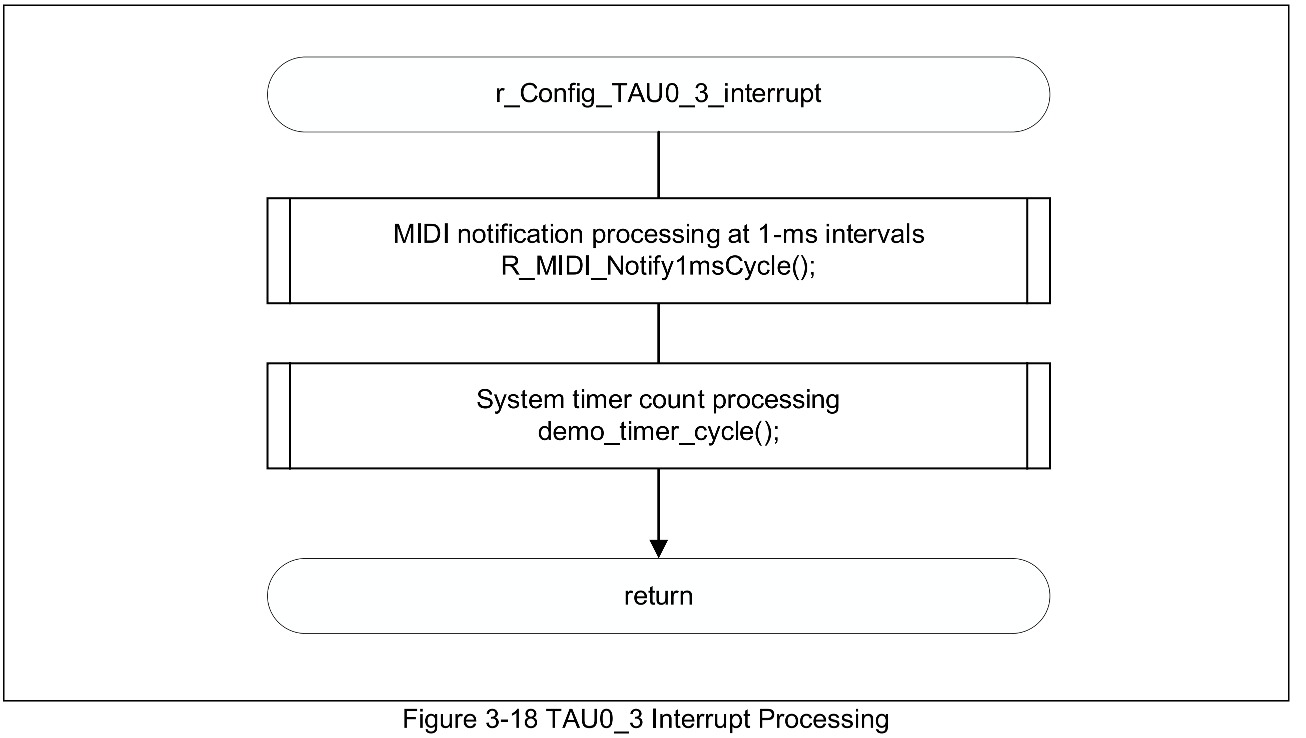

Figure 3-18 TAU0_3 Interrupt Processing

3.9.11 AD conversion end interrupt processing

Figure 3-19 shows the flowchart for AD conversion end interrupt processing.

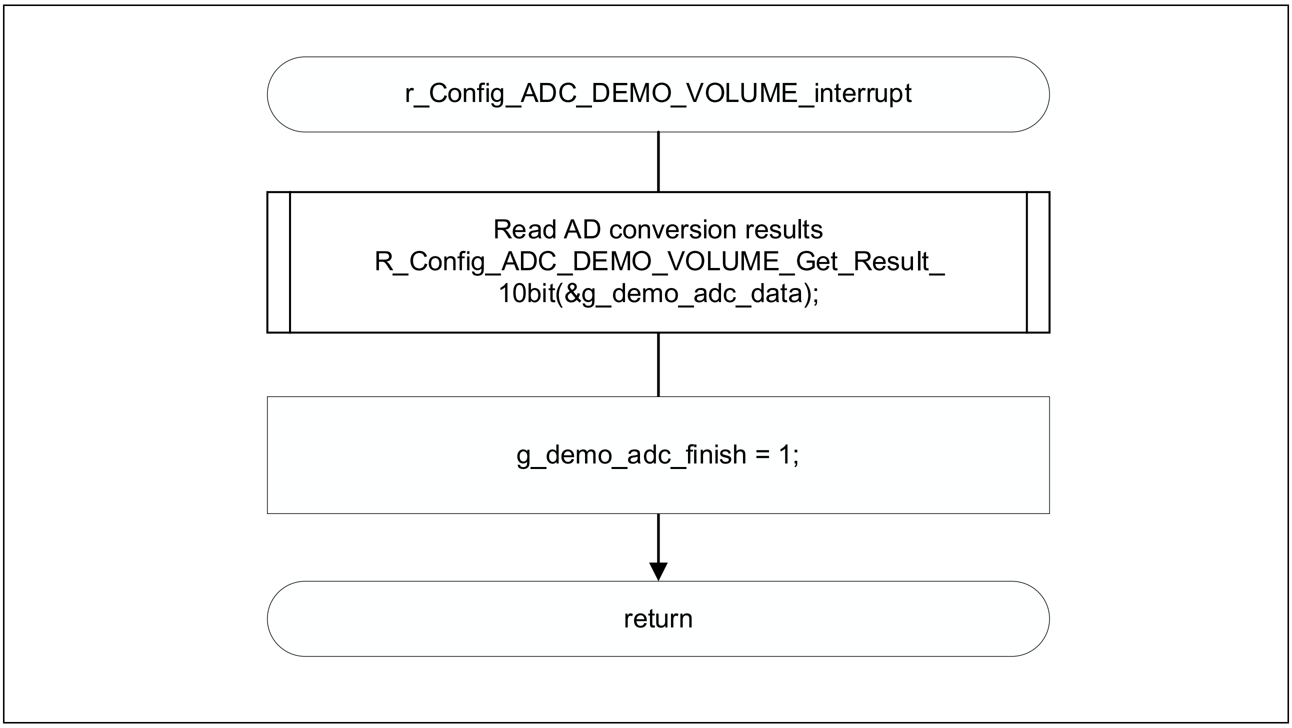

Figure 3-19 AD Conversion End Interrupt Processing

3.9.12 LED matrix data CSI transmission end processing

Figure 3-20 shows the flowchart for CSI transmission end processing for LED matrix data.

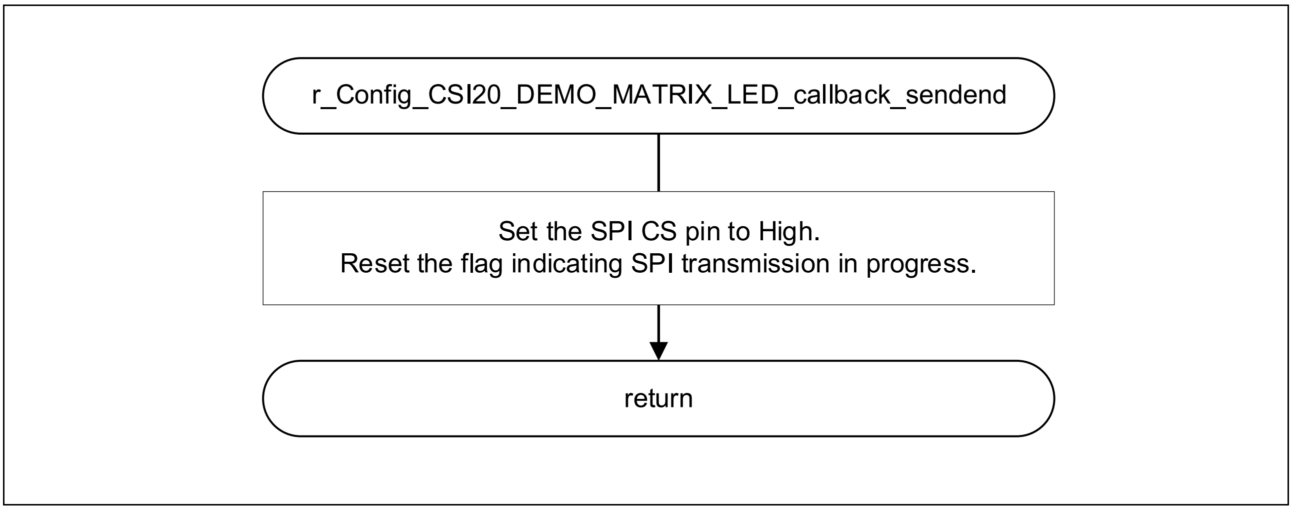

Figure 3-20 LED Matrix Data CSI Transmission End Processing

3.9.13 UART0 transmission end processing

Figure 3-21 shows the flowchart for UART0 transmission end processing.

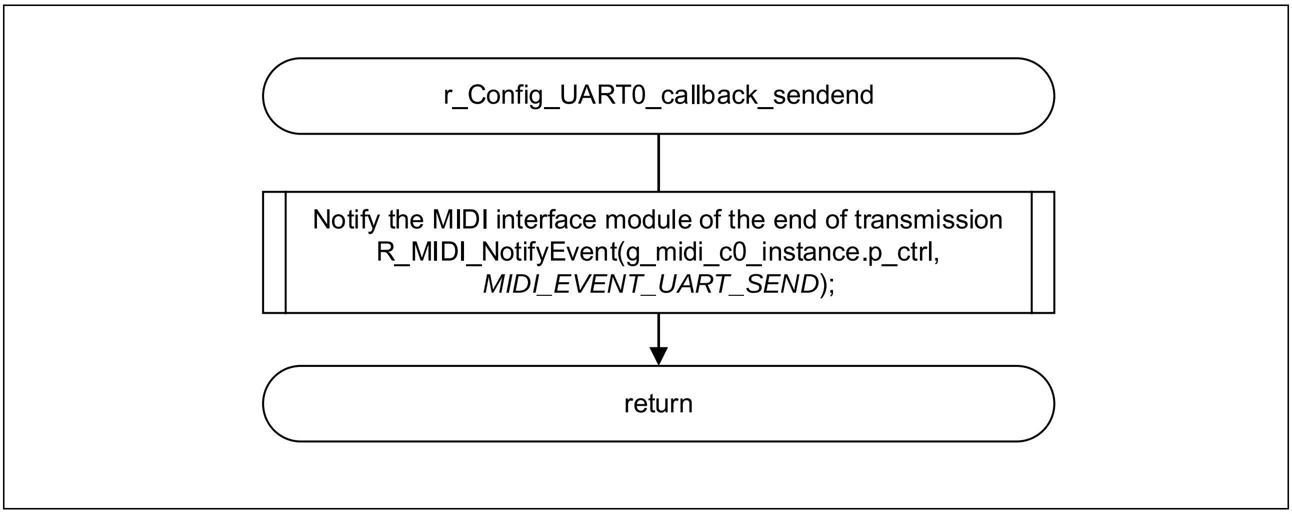

Figure 3-21 UART0 Transmission End Processing

3.9.14 UART0 reception end processing

Figure 3-22 shows the flowchart for UART0 reception end processing.

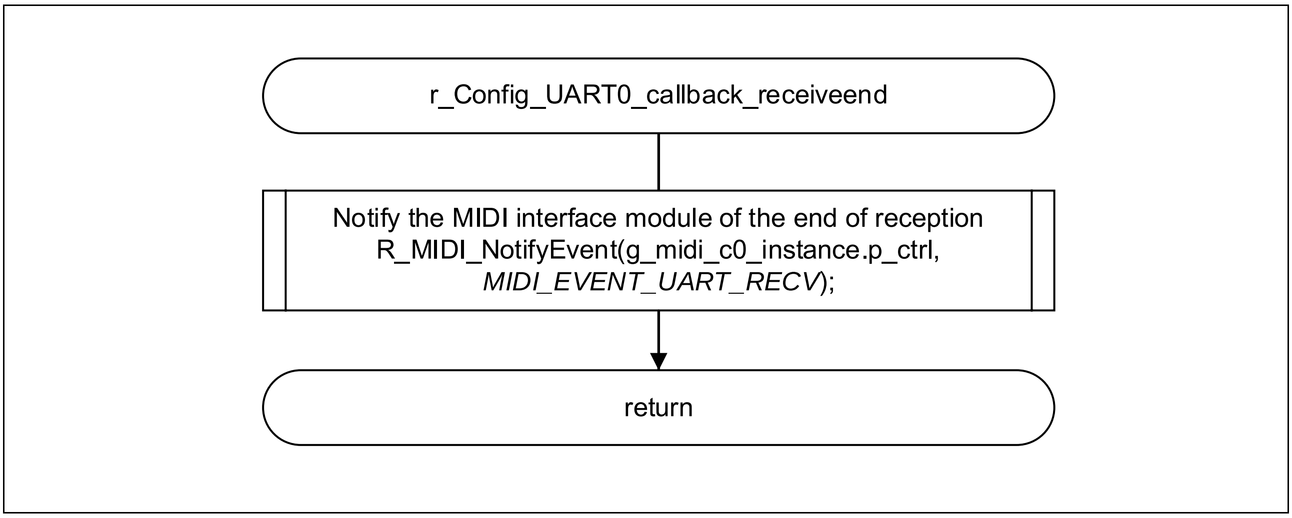

Figure 3-22 UART0 Reception End Processing

4. Configuring the DAW

This section describes the procedure by using Domino (https://takabosoft.com/domino) as a DAW.

Unzip the downloaded file to extract it as shown in Figure 4-1.

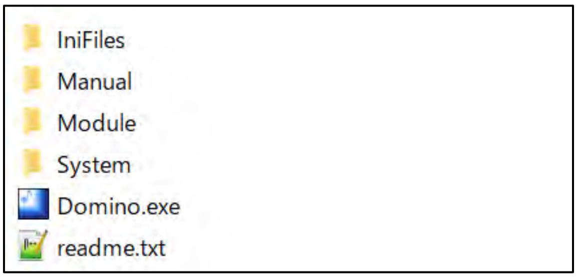

Figure 4-1 Extracted Domino Files

Among the extracted files, execute Domino.exe to launch the application as shown in Figure 4-2.

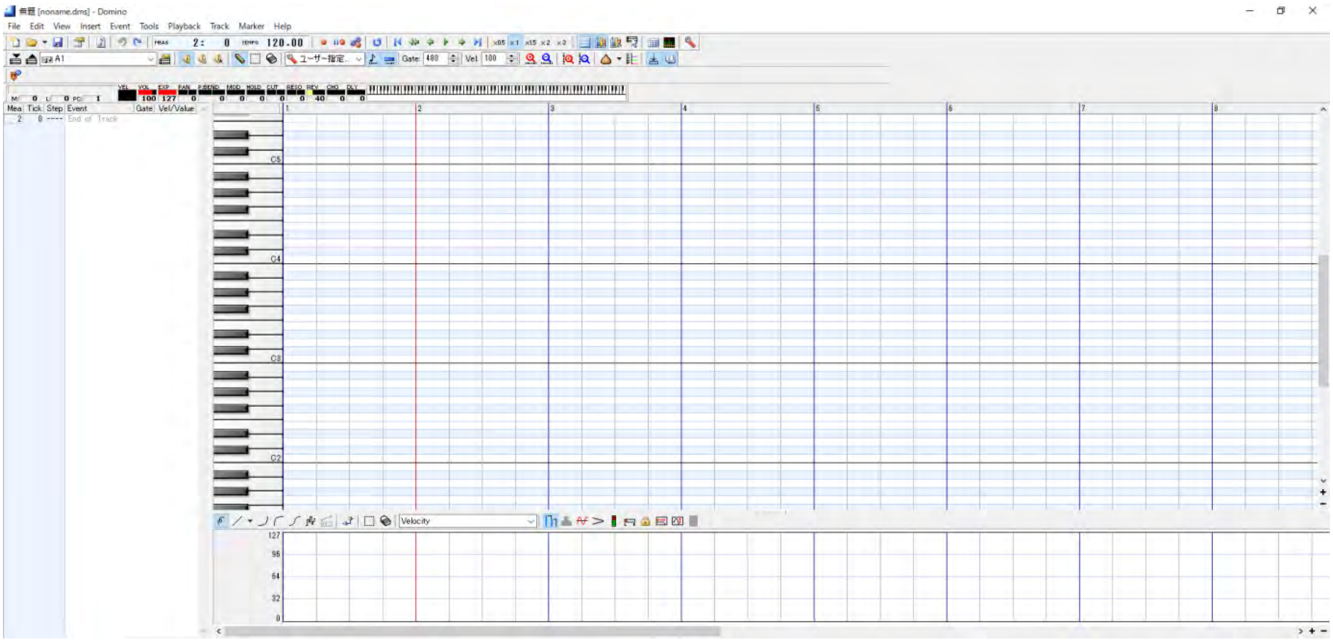

Figure 4-2 Domino Main Window

The UM-ONE manufactured by Roland is used as shown in Table 2-1 Hardware List. Therefore, specify the connection between the PC and the MIDI IN pin by using the Environment Settings dialog box (Figure 4-3) displayed by clicking [Files] and then [Environment Settings] or by pressing the F12 key. In addition, specify SC-88Pro as the sound module.

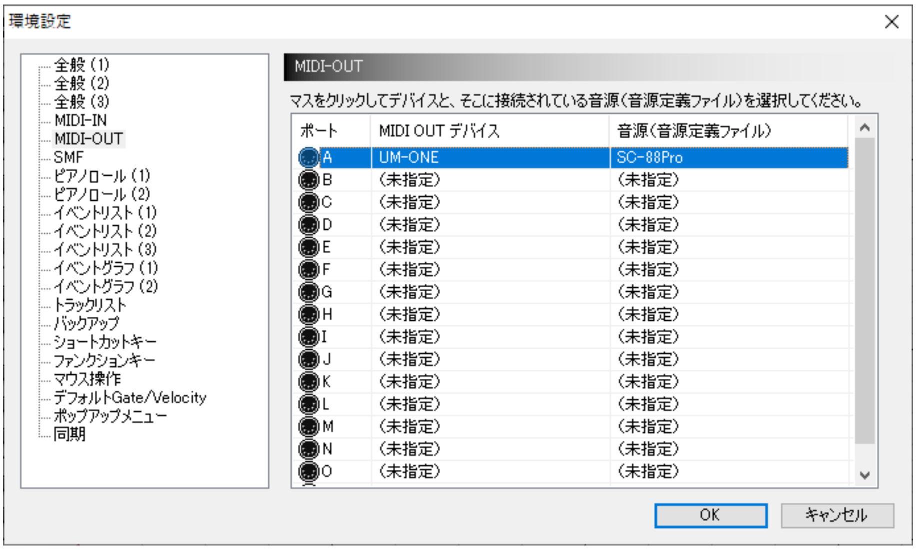

Figure 4-3 Domino Environment Settings

In the Open File dialog box (Figure 4-4) displayed by clicking [Files] and then [Open] or by pressing the Ctrl + O keys, open the file you want to play.

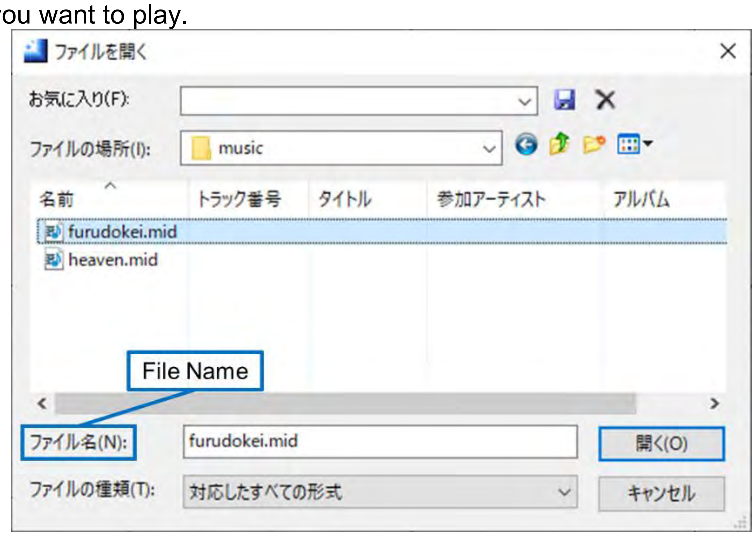

Figure 4-4 Dialog Box Used to Open the Domino File

Figure 4-5 indicates that the MIDI file has been read and is ready to play music.

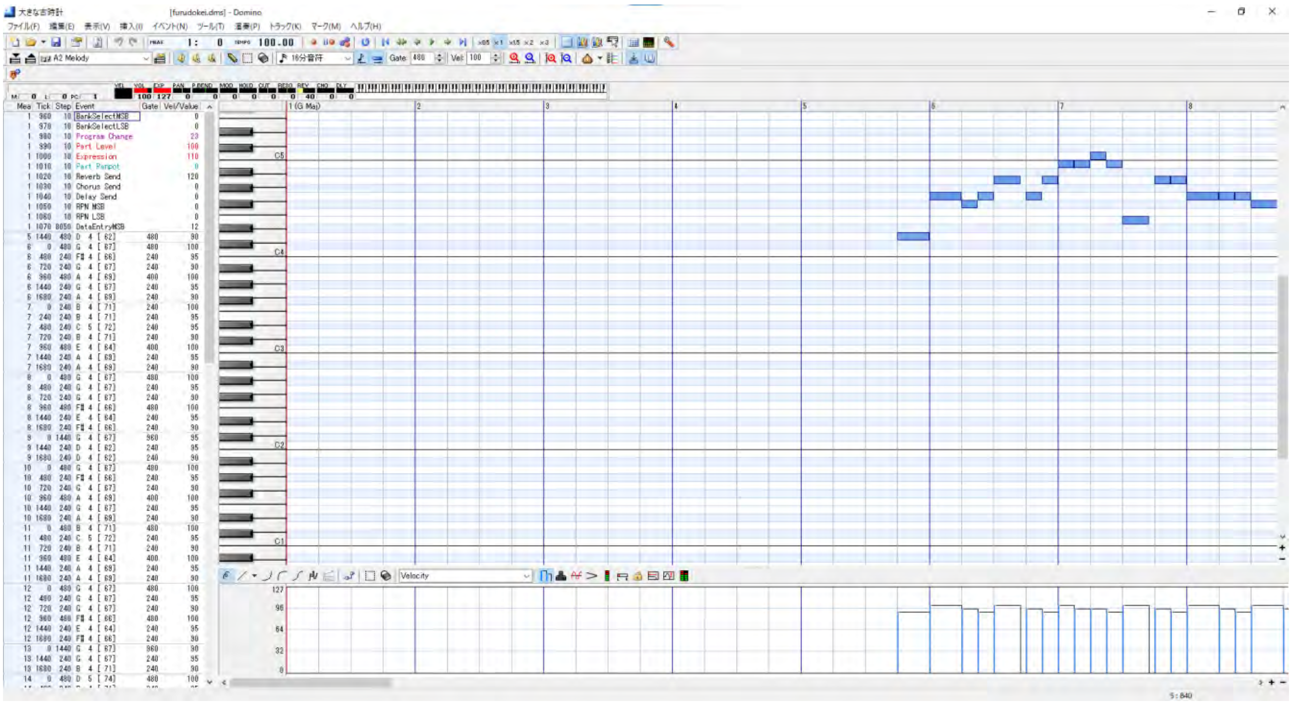

Figure 4-5 Domino Main Window (MIDI File Opened)

To start playback, click [Play] and then click the start/stop button shown in Figure 4-6, or press the space key.

Figure 4-6 Domino Play

5. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

6. Reference Documents

RL78 Family MIDI Interface Module Software Integration System (R01AN7265E)
RL78 Family MIDI Performance Control Sample Software Using SIS (R01AN7491E)
RL78/G16 User’s Manual: Hardware (R01UH0980E)
RL78 family user's manual: software (R01US0015E)
RL78/G16 Fast Prototyping Board User’s Manual (R12UM0048)
(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

Matrix LED

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

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	2024.11.29	—	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.

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/.