RL78/G13

Software UART CC-RL

R01AN4671EJ0100
Rev.1.00
May. 7, 2019

Introduction

This application note describes how to implement the software UART communication functions by using the external interrupts and timer array unit.

Target Device

RL78/G13

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

This application system uses the external interrupts and timer array unit to implement the UART communication functions by programming. Specifically, the system analyzes the ASCII characters sent from the opposite device and makes responses.

Table 1.1 lists peripheral functions to be used and their uses.

Table 1.1 Peripheral Functions and Applications

Peripheral Function	Application
External interrupt	Detects the start bit on the receiving pin.
Timer array unit channel0	Counts the data output timing for transmission.
Timer array unit channel1	Counts the latch timing for reception.

1.1 Setting Software UART Communication Operations

Table 1.2 shows the available software UART communication operations.

The software UART communication operations can be set by the #define declaration statements on the 40th to 51st lines of the r_cg_userdefine.h file. Validating the #define declaration statements allows the settings to be reflected. Only one statement should be validated for each item.

Table 1.2 Software UART Communication Operations

Item	Contents	Setting method
Data transfer order	MSB first	Enable #define MSB_FIRST
	LSB first	Enable #define LSB_FIRST
Parity	Non-parity	Enable #define NON_PARITY
	0 parity	Enable #define ZERO_PARITY
	Odd parity	Enable #define ODD_PARITY
	Even parity	Enable #define EVEN_PARITY
Data output	Data invert	Enable #define INVERT
	Data non-invert	Enable #define NON_INVERT
Data length	8bit	—
Start bit length	1bit	—
Stop bit length	1bit	—

1.2 Baud Rate

The software UART counts the receive data latch timing and transmit data output timing by using the interval timer of the timer array unit. Calculate the appropriate interval time value (TDRmn register) to be set based on the baud rate according to the equation below. Set the calculated value (TDRmn register) in #define BAUD_RATE of the r_cg_userdefine.h file.

Interval time value to be set (TDRmn register) = (CPU clock frequency / baud rate)

For reception, the start bit is latched when the time corresponding to baud rate/2 elapses after detection of the valid edge of the external interrupt. Therefore, the interval time for the start bit latch timing is different from that for the other bits. Note that the latch timing is delayed because latch operation is performed in the interrupt processing routine. For this application system, subtract 67 as shown in the equation below and set the calculated value as the interval time (TDRmn register) to be set in order to adjust the latch timing delay. Set the calculated value (TDRmn register) in #define BAUD_RATE_START_BIT of the r_cg_userdefine.h file.

Interval time value to be set (TDRmn register) = (CPU clock frequency / (baud rate / 2)) – 67

1.3 Software UART Reception

This section describes reception by the software UART.

Reception starts upon detection of the valid edge of the external interrupt. The latch timing on the receiving pin is counted by the interval timer of the timer array unit. The pin level is latched when baud rate/2 elapses after the input level changes. The value latched is stored in the receive data variable and is shifted.

When the stop bit latch is complete, it is checked whether a framing error or a parity error has occurred. If no error has occurred, the received data is processed to create an 8-bit data according to the software UART communication operation. When the 8-bit received data is stored in RAM, reception of one unit of data is completed.

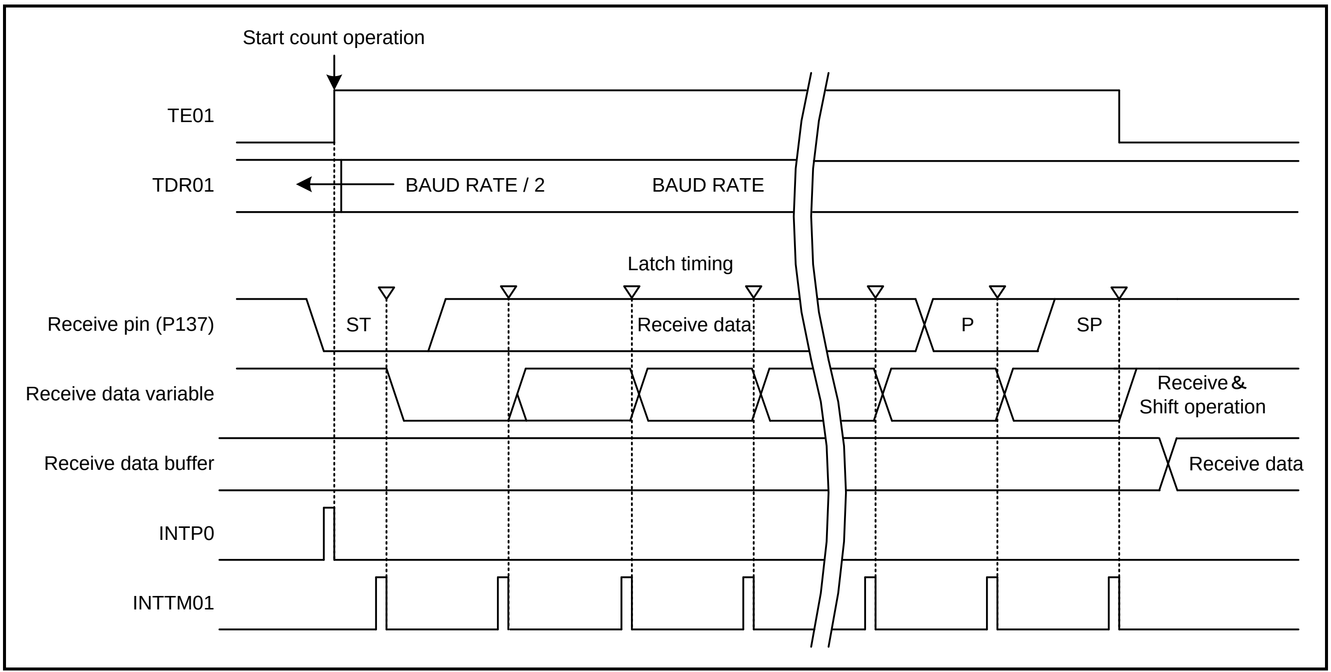

Figure 1.1 Software UART reception timing chart

1.4 Software UART Transmission

This section describes transmission by the software UART.

According to the software UART communication operation (data transfer sequence, parity bit, data output), 16-bit transmit data is created which consists of a start bit as the LSB, data, a parity bit, and a stop bit in this order.

Data output timing is counted by the interval timer of the timer array unit. Each time an interrupt is generated, data is sequentially output to P130 starting with the LSB. When the stop bit is output, a transmission of one unit of data is completed.

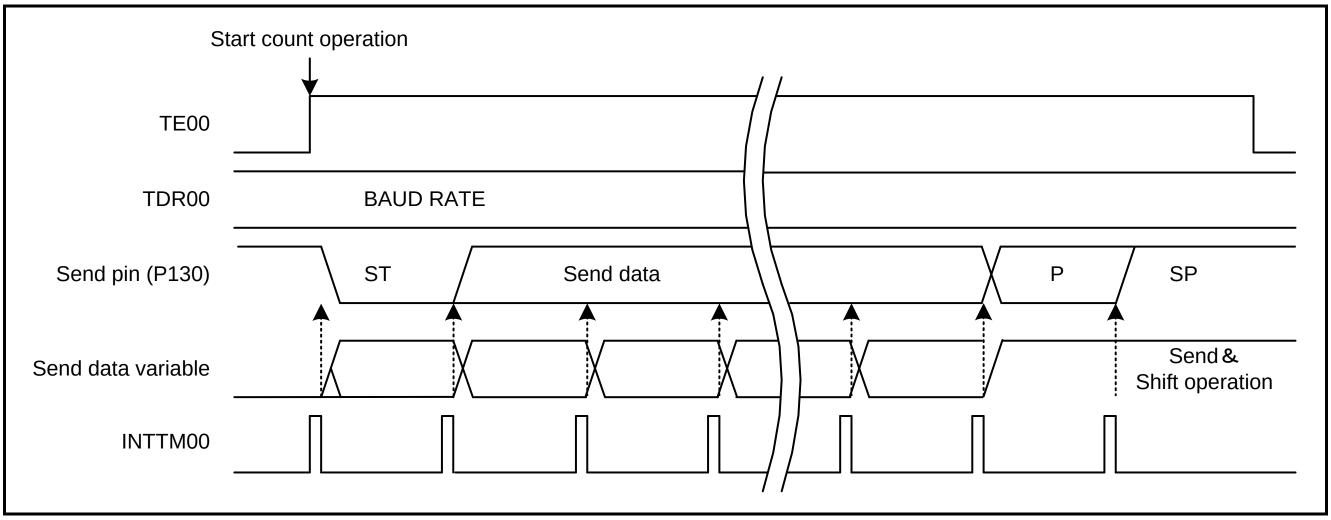

Figure 1.2 Software UART transmission timing chart

1.5 CPU Utilization during Software UART Transmission and Reception

The software UART uses the CPU during transmission and reception. Table 1.3 shows the CPU utilization during transmission and reception. If the CPU clock or baud rate is changed, careful evaluation is required.

Table 1.3 CPU utilization

CPU clock	Baud rate	CPU utilization
32MHz	9600bps	Send: about 1%、Receive: about 1%
32MHz	19200bps	Send: about 2%、Receive: about 2%
16MHz	19200bps	Send: about 4%、Receive: about 4%

2. Operation Check Conditions

The sample code described in this application note has been checked under the conditions listed in the table below.

Table 2.1 Operation Check Conditions

Item	Description
MCU used	RL78/G13 (R5F100LE)
Operating frequencies	● High-speed on-chip oscillator clock: 32MHz ● CPU/peripheral hardware clock: 32MHz
Operating voltage	3.3V (operating range 1.8V to 5.5V) LVD operations (V_{LVD}): Reset mode 2.75V (2.70V to 2.81V)
Integrated development environment (CS+)	CS+ for CC V8.01.00 from Renesas Electronics Corp.
C compiler (CS+)	CC-RL V1.08.00 from Renesas Electronics Corp.
Integrated development environment (e^2 studio)	e^{2}studio V7.3.0 from Renesas Electronics Corp.
C compiler (e^2 studio)	CC-RL V1.08.00 from Renesas Electronics Corp.

3. Related Application Notes

Application notes related to this document are shown below. Please refer to these as needed.

RL78/G13 Initialization CC-RL (R01AN2575EJ) Application note
RL78/G13 Serial Array Unit (UART Communication) CC-RL (R01AN2517EJ) Application note

4. Hardware Explanation

4.1 Hardware Configuration Example

Figure 4.1 shows an example of the hardware configuration used in this application note.

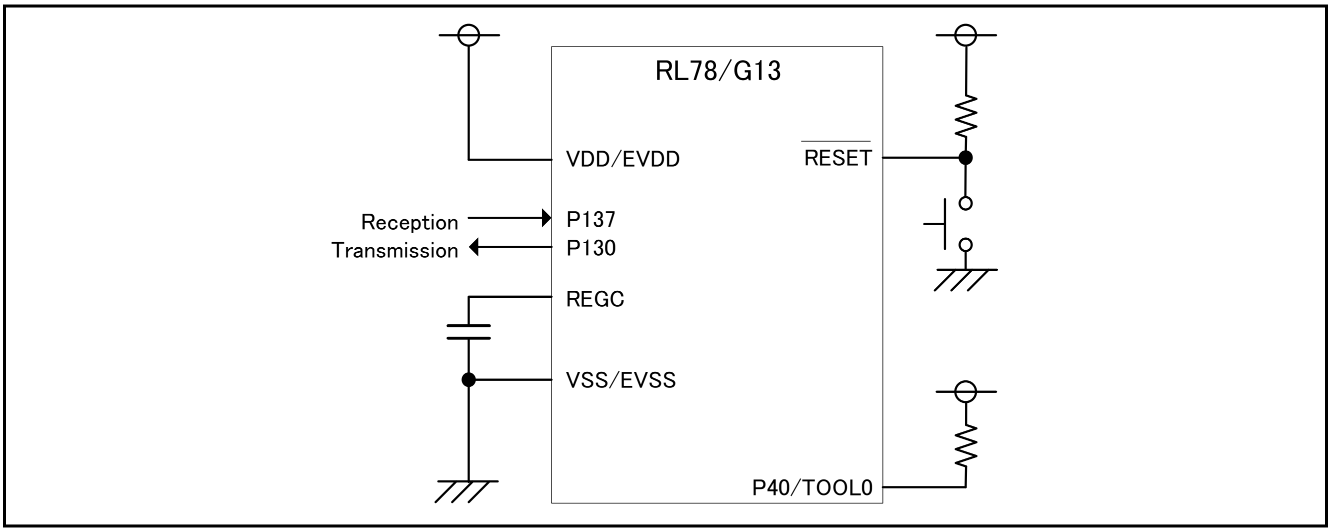

Figure 4.1 Hardware Configuration

Note: 1. This simplified circuit diagram was created to show an overview of connections only.
When actually designing your circuit, make sure the design includes sufficient pin processing and meets electrical characteristic requirements. (Connect each input-only port to V_{DD} or V_{SS} through a resistor.)

4.2 Used Pin List

Table 4.1 provides List of Pins and Functions.

Table 4.1 List of Pins and Functions

Pin name	Input/Output	Function
P130	Outut	Output data
P137/INTP0	Input	Input data

5. Description of the Software

5.1 Operation Outline

The sample code described in this application note receives data from the device on the opposite side, and transmits the data corresponding to the received data to the device on the opposite side. If a communication error occurs, it transmits the data corresponding to the error to the device on the opposite side. Table 5.1 and Table 5.2 show the correspondence between receive data and transmit data.

Table 5.1 Correspondence between Receive Data and Transmit Data

Receive Data	Response (Transmit)
T (54H)	O (4FH), K (4BH), ”CR” (0DH), ”LF” (0AH)
t (74H)	o (6FH), k(6BH), ”CR” (0DH), ”LF” (0AH)
Other than above	U (55H), C (43H), ”CR” (0DH), ”LF” (0AH)

Table 5.2 Correspondence between Error and Transmit Data

Error	Response (Transmit)
Parity error	P (50H), E (45H), ”CR” (0DH), ”LF” (0AH)
Framing error	F (46H), E(45H), ”CR” (0DH), ”LF” (0AH)

Details are described in ① to ⑪ below.

① Initialize the CPU clock.

＜CPU clock setup conditions＞

- The 32MHz high-speed on-chip oscillator clock is set as the CPU/peripheral hardware clock (fCLK).
- The high-speed system clock pin operating mode is set to input port mode.
- The subsystem clock pin operating mode is set to input port mode.

② Initialize the input/output ports.

＜Input/output port setup conditions＞

- Set P137 input mode.
- Set P130 output mode.

③ Initialize the timer array unit.

＜Channel 0 setting condition＞

- Use the interval timer mode as the timer operation mode.
- Initialize timer data register 00 (TDR00) to 104us.
- Set the timer output enable register to disable operation.
- Use timer interrupts (INTTM00) from timer channel0.

＜Channel 1 setting condition＞

- Use the interval timer mode as the timer operation mode.
- Initialize timer data register 01 (TDR01) to 104us.
- Set the timer output enable register to disable operation.
- Use timer interrupts (INTTM01) from timer channel1.

④ Performs initial setting for the external interrupt.

<External interrupt setting conditions>

- Sets the valid edge for the INTP0 pin.

  When #define NON_INVERT is enabled: Rising edge

  When #define INVERT is enabled: Falling edge

⑤ Sets the initial output level of the transmitting pin (P130).

  When #define NON_INVERT is enabled: High-level output

  When #define INVERT is enabled: Low-level output

⑥ Initializes the reception status, sets the baud rate, and enables the external interrupt.

⑦ Enters HALT mode and waits for data to receive.

⑧ Performs the data reception process upon generation of the external interrupt. After completion of the data reception process, disables the external interrupt.

⑨ Performs the data transmission process depending on the received data as described below.

- Received data is “T”: Transmits ”OK”.
- Received data is “t”: Transmits ”ok”.
- Received data is neither “T” nor ”t”: Transmits ”UC”.
- A framing error occurs: Transmits ”FE”.
- A parity error occurs: Transmits ”PE”.

⑩ Waits for completion of the transmission process.

⑪ Initializes the reception status and enables the external interrupt.

⑫ Repeats steps ⑦ to ⑪.

Data reception process

① Upon generation of the external interrupt, returns from HALT mode and starts the interval timer counting operation of the timer array unit. Then modifies the interval time setting from “baud rate/2” to “baud rate”.

② When the specified interval time (baud rate/2) passes, the interval timer interrupt is generated. Latches P137 and stores the start bit value in the receive data variable. Shifts the receive data variable to the left by one bit.

③ Enters HALT mode and waits for the interval timer interrupt to be generated.

④ When the specified interval time (baud rate) passes, the interval timer interrupt is generated. Latches P137 and stores the data value in the receive data variable. Shifts the receive data variable to the left by one bit.

⑤ Repeats steps ③ and ④ until receiving the stop bit.

⑥ After receiving all the bits, stops the interval timer counting operation.

⑦ Reverses the receive data variable value when #define INVERT is enabled.

⑧ Checks for a framing error. If a framing error has occurred, sets the framing error as the reception status.

⑨ Checks for a parity error when #define EVEN_PARITY or #define ODD_PARITY is enabled. If a parity error has occurred, sets the parity error as the reception status.

⑩ If no reception error has occurred in steps ⑧ and ⑨, removes the start bit, parity bit, and stop bit from the receive data variable to create 8-bit data, and stores it in the receive data buffer. Rearranges the bit order of the 8-bit data when #define LSB_FIRST is enabled.

⑪ Updates the receive data pointer and receive data number counter.

⑫ Enables the external interrupt and sets “baud rate/2” as the interval time. Then enters HALT mode.

⑬ Repeats steps ① to ⑫ until receiving the number of data specified in the receive data variable. Upon completion of reception, sets the reception end flag to end reception.

Data transmission process

① Reads 8-bit transmit data from RAM and stores it in the transmit data buffer.

② Sets the stop bit and parity bit in the transmit data variables. Sets the parity bit when #define ZERO_PARITY, #define EVEN_PARITY, or #define ODD_PARITY is enabled.

③ Rearranges the bit order of the transmit data buffer when #define MSB_FIRST is enabled.

④ Adds the transmit data buffer value to the transmit data variable and creates the data so that it consists of a start bit as the LSB, data, a parity bit, and a stop bit in this order.

⑤ Reverses the transmit data variable value when #define INVERT is enabled.

⑥ Starts the interval timer counting operation of the timer array unit. An interrupt is generated simultaneously. Sets the LSB value of the transmit data variable to P130 and outputs the data. Shifts the transmit data variable to the right by one bit.

⑦ Enters HALT mode and waits for the interval timer interrupt to be generated.

⑧ When the specified interval time passes, the interval timer interrupt is generated. Sets the LSB value of the transmit data variable to P130 and outputs the data. Shifts the transmit data variable to the right by one bit.

⑨ Repeats steps ⑦ and ⑧ until transmitting the stop bit.

⑩ After transmitting all the bits, stops the interval timer counting operation.

⑪ Enables multiple interrupt servicing.

⑫ Updates the transmit data pointer and transmit data number counter.

⑬ Executes steps ① to ⑤.

⑭ Starts the interval timer counting operation. Prevents interrupt generation at the start of the counting operation in order to ensure the stop bit period for the previous transmission.

⑮ Repeats steps ⑦ to ⑭ until transmitting the number of data specified in the transmit data variable. Upon completion of transmission, sets the transmission end flag to end transmission.

5.2 Option Byte Settings

Table 5.3 lists the option byte settings.

Table 5.3 Option Byte Settings

Address	Setting Value	Contents
000C0H/010C0H	1110 1111B	Watchdog timer operation is stopped (count is stopped after reset)
000C1H/010C1H	0111 1111B	LVD Reset mode 2.75V (2.70V~2.81V)
000C2H/010C2H	1110 1000B	HS mode, High-speed on-chip oscillator clock: 32MHz
000C3H/010C3H	1000 0100B	On-chip debugging enabled

5.3 List of Constants

Table 5.4 lists the constants that are used in this sample program.

Table 5.4 Constants

Constant	Setting Value	Description
LENGTH_8BIT	0x08	Number of bits of data
FRAMING_ERROR	0x01	Framing error
PARITY_ERROR	0x02	Parity error
BAUD_RATE_START_BIT	0x0047	Latch timing setting value
BAUD_RATE	0x0115	Baud rate setting value (9600bps)
BIT_TOTAL	0x10、0x11	Total number of bits
OUTPUT_INIT	0x00、0x01	Initial value of transmit pin
ACTIVE_LEVEL	0x80、0x00	Valid value of receive pin
g_messageOK[4]	"OK¥r¥n"	Response message to reception of “T”
g_messageok[4]	"ok¥r¥n"	Response message to reception of “t”
g_messageUC[4]	"UC¥r¥n"	Response message to reception of characters other than “T” or “t”.
g_messageFE[4]	"FE¥r¥n"	Response message to a framing error
g_messagePE[4]	"PE¥r¥n"	Response message to a parity error

5.4 List of Variables

Table 5.5 lists the global variable that is used by this sample program.

Table 5.5 Global Variable

Type	Variable Name	Contents	Function Used
uint8_t	gp_uarts_tx_address	Transmit data pointer	r_uarts_send()、r_tau0_channel0_interrupt()
uint8_t	gp_uarts_rx_address	Receive data pointer	r_uarts_receive()、r_tau0_channel1_interrupt()
uint8_t	g_uarts_tx_buffer	Transmit data buffer	r_uarts_send()、r_tau0_channel0_interrupt()、r_uarts_tx_set()
uint8_t	g_uarts_rx_buffer	Receive data buffer	main()、R_MAIN_UserInit()、r_tau0_channel1_interrupt()
uint16_t	g_uarts_tx_count	Transmit data number counter	r_uarts_send()、r_tau0_channel0_interrupt
uint16_t	g_uarts_rx_count	Receive data number counter	r_uarts_receive()、r_tau0_channel1_interrupt()
uint16_t	g_uarts_rx_length	Receive data number	r_uarts_receive()、r_tau0_channel1_interrupt()
uint16_t	g_uarts_tx_data	Transmit data	r_tau0_channel0_interrupt()、r_uarts_tx_set()
uint16_t	g_uarts_rx_data	Receive data	r_tau0_channel1_interrupt()
uint8_t	g_uarts_rx_error	Receive error status	main()、r_uarts_receive()、r_tau0_channel1_interrupt()
uint8_t	g_uarts_rx_end	Reception complete flag	main()、r_tau0_channel1_interrupt()
uint8_t	g_uarts_tx_end	transmission complete flag	main()、r_uarts_send()、r_tau0_channel0_interrupt()
uint8_t	g_tx_bit_count	Transmit bit counter	r_uarts_send()、r_tau0_channel0_interrupt()
uint8_t	g_rx_bit_count	Receive bit counter	r_tau0_channel1_interrupt()
uint16_t	g_check	Parity bit comparison value	r_tau0_channel1_interrupt()
uint16_t	g_parity	Parity bit receive value	r_tau0_channel1_interrupt()
uint8_t	g_port	Port read value	r_tau0_channel1_interrupt()

5.5 List of Functions

Table 5.6 lists the functions that are used by this sample program.

Table 5.6 List of Functions

Function Name	Outline
main	Main processing
R_MAIN_UserInit	Main initial setting
R_INTC0_Start	External interrupt enable function
R_INTC0_Stop	External interrupt disable function
r_uarts_send	Software UART data transmission function
r_uarts_receive	Software UART reception status initialization function
r_parity_check	Parity check function
r_bit_swap_8bit	Data order sort function
r_uarts_tx_set	Software UART communication operation setting function
r_intc0_interrupt	External interrupt processing
r_tau0_channel0_interrupt	Timer array unit channel 0 interrupt processing
r_tau0_channel1_interrupt	Timer array unit channel 1 interrupt processing

5.6 Function Specifications

This part describes specifications of the sample code.

[Function Name]	main
Outline	Main processing
Header	r_cg_macrodriver.h、r_cg_cgc.h、r_cg_port.h、r_cg_intc.h、r_cg_timer.h、r_cg_userdefine.h
Declaration	—
Description	After executing the main user initialization function, enters HALT mode and waits for data to receive. When having received 1-byte data, transmits the data corresponding to the received data.
Arguments	None
Remarks	None

[Function Name]	R_MAIN_UserInit
Outline	Main initial setting
Header	r_cg_macrodriver.h、r_cg_cgc.h、r_cg_port.h、r_cg_intc.h、r_cg_timer.h、r_cg_userdefine.h
Declaration	void R_MAIN_UserInit(void);
Description	Sets the initial output value to P130 and then initializes the software UART status. Enables the external interrupts and enables maskable interrupts.
Arguments	None
Remarks	None

[Function Name]	R_INTC0_Start
Outline	External interrupt enable function
Header	r_cg_macrodriver.h、r_cg_intc.h、r_cg_userdefine.h
Declaration	void R_INTC0_Start(void);
Description	Enable external interrupt processing.
Arguments	None
Remarks	None

[Function Name]	R_INTC0_Stop
Outline	External interrupt disable function
Header	r_cg_macrodriver.h、r_cg_intc.h、r_cg_userdefine.h
Declaration	void R_INTC0_Stop(void);
Description	Disable external interrupt processing
Arguments	None
Remarks	None

[Function Name]	r_uarts_send
Outline	Software UART data transmission function
Header	r_cg_macrodriver.h、r_cg_cgc.h、r_cg_port.h、r_cg_intc.h、r_cg_timer.h、r_cg_userdefine.h
Declaration	void r_uarts_send(uint8_t * tx_buf, uint16_t tx_num);
Description	Initial setting of software UART transmission is performed. start sending data.
Arguments	uint8_t * tx_buf : Transmit data buffer address
	uint16_t tx_num : Transmit data buffer size
Remarks	None

[Function Name]	r_uarts_receive
Outline	Software UART reception status initialization function
Header	r_cg_macrodriver.h、r_cg_cgc.h、r_cg_port.h、r_cg_intc.h、r_cg_timer.h、r_cg_userdefine.h
Declaration	void r_uarts_receive(uint8_t * rx_buf, uint16_t rx_num);
Description	Initial setting of software UART reception is performed.
Arguments	uint8_t * rx_buf : Receive data buffer address
	uint16_t rx_num : Receive data buffer size
Remarks	None

[Function Name]	r_parity_check
Outline	Parity check function
Header	r_cg_macrodriver.h、r_cg_cgc.h、r_cg_port.h、r_cg_intc.h、r_cg_timer.h、r_cg_userdefine.h
Declaration	uint16_t r_parity_check(uint16_t data);
Description	Calculate the parity value of the data.
Arguments	uint16_t data : Transmit / receive data
Remarks	uint16_t: Parity value

[Function Name]	r_bit_swap_8bit
Outline	Data order sort function
Header	r_cg_macrodriver.h、r_cg_cgc.h、r_cg_port.h、r_cg_intc.h、r_cg_timer.h、r_cg_userdefine.h
Declaration	uint8_t r_bit_swap_8bit(uint8_t data);
Description	Sort the order of the bits of the data.
Arguments	uint8_t data : Transmit / receive data
Remarks	uint8_t: Sored data

[Function Name]	r_uarts_tx_set
Outline	Software UART communication operation setting function
Header	r_cg_macrodriver.h、r_cg_cgc.h、r_cg_port.h、r_cg_intc.h、r_cg_timer.h、r_cg_userdefine.h
Declaration	void r_uarts_tx_set(void);
Description	Set the communication operation of the software UART.
Arguments	None
Remarks	None

[Function Name]	r_intc0_interrupt
Outline	External interrupt processing
Header	r_cg_macrodriver.h、r_cg_intc.h、r_cg_timer.h、r_cg_userdefine.h
Declaration	#pragma interrupt r_intc0_interrupt(vect=INTP0)
Description	Start counting channel1 of timer array unit. Disable external interrupts.
Arguments	None
Remarks	None

[Function Name]	r_tau0_channel0_interrupt
Outline	Timer array unit channel 0 interrupt processing
Header	r_cg_macrodriver.h、r_cg_intc.h、r_cg_timer.h、r_cg_userdefine.h
Declaration	#pragma interrupt r_tau0_channel0_interrupt(vect=INTTM00)
Description	Sets the transmit data to the P13 register. When having transmitted 1 unit of data, starts transmitting the next data.
Arguments	None
Remarks	None

[Function Name]	r_tau0_channel1_interrupt
Outline	Timer array unit channel 1 interrupt processing
Header	r_cg_macrodriver.h、r_cg_intc.h、r_cg_timer.h、r_cg_userdefine.h
Declaration	#pragma interrupt r_tau0_channel1_interrupt(vect=INTTM01,bank=RB2,enable=true)
Description	Reads the input level from the P13 register. When having received the stop bit, checks for a framing error and a parity error. If no error has occurred, stores the received data in the receive data buffer.
Arguments	None
Remarks	None

5.7 Flowcharts

Figure 5.1 shows an overall from of the sample code.

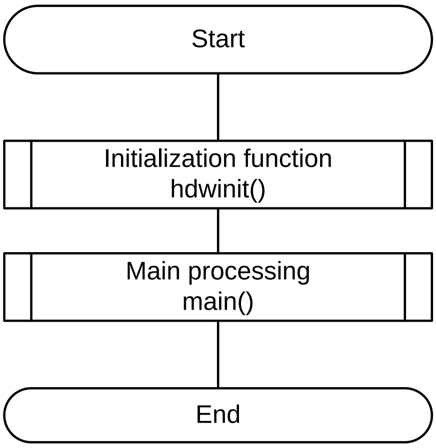

Figure 5.1 Overall Flow

5.7.1 Initialization Function

Figure 5.2 shows the flowchart for the initialization function.

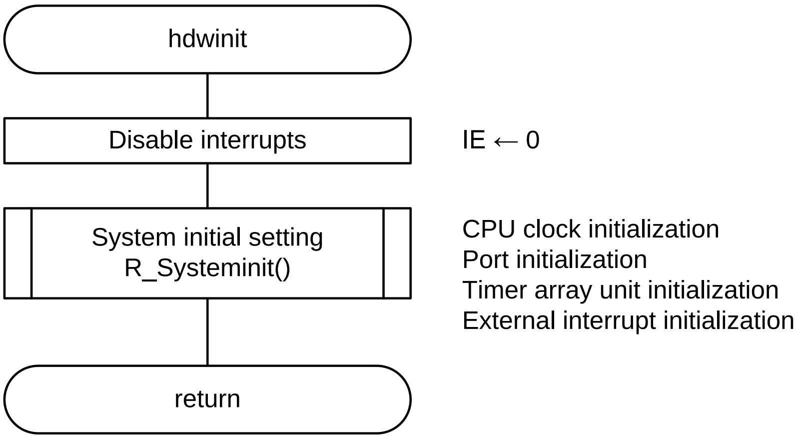

Figure 5.2 Initialization Function

5.7.2 System Initial Setting

Figure 5.3 shows the flowchart for the system initial setting.

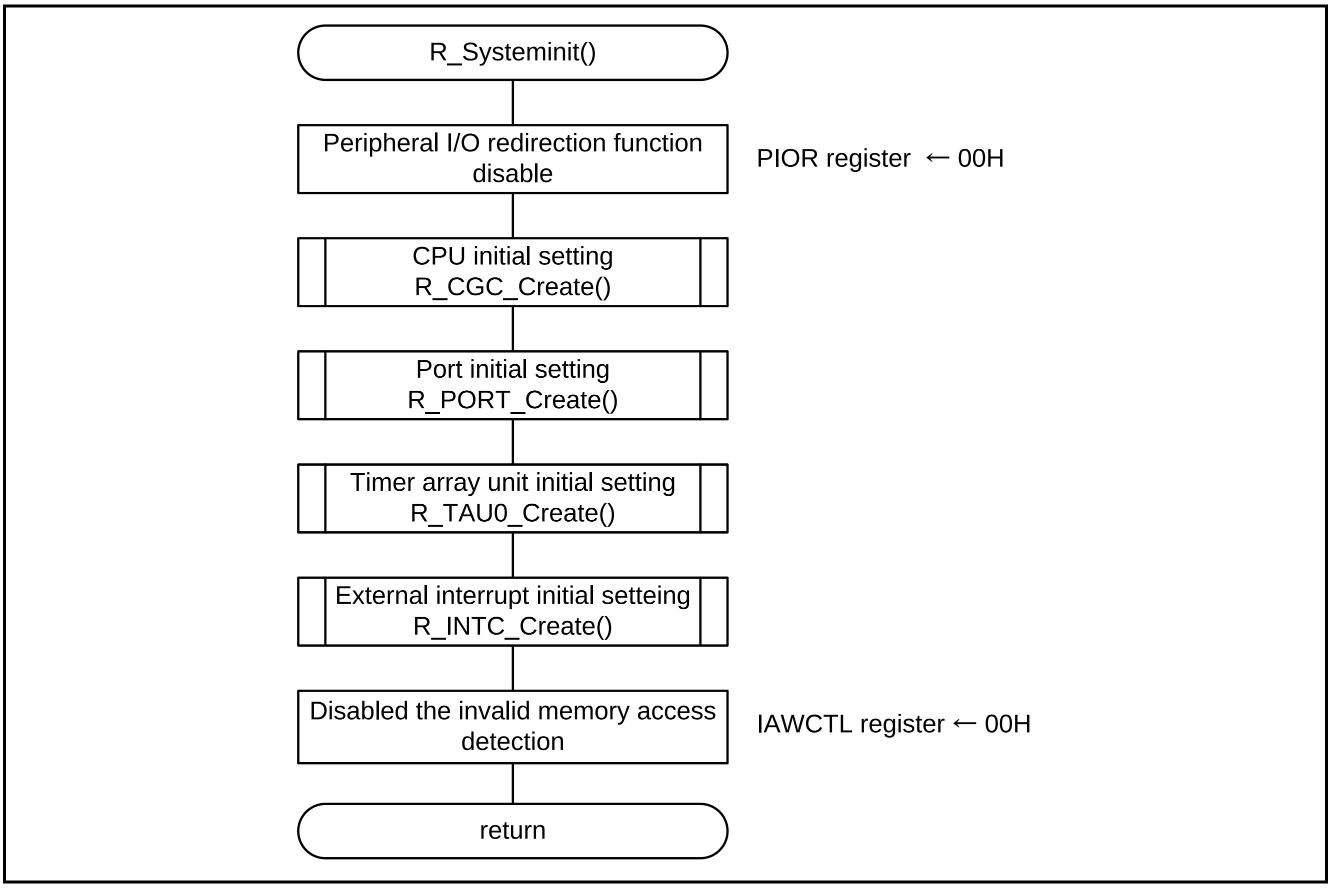

Figure 5.3 System Initial Setting

5.7.3 CPU Initial Setting

Figure 5.4 shows the flowchart for the CPU initial setting.

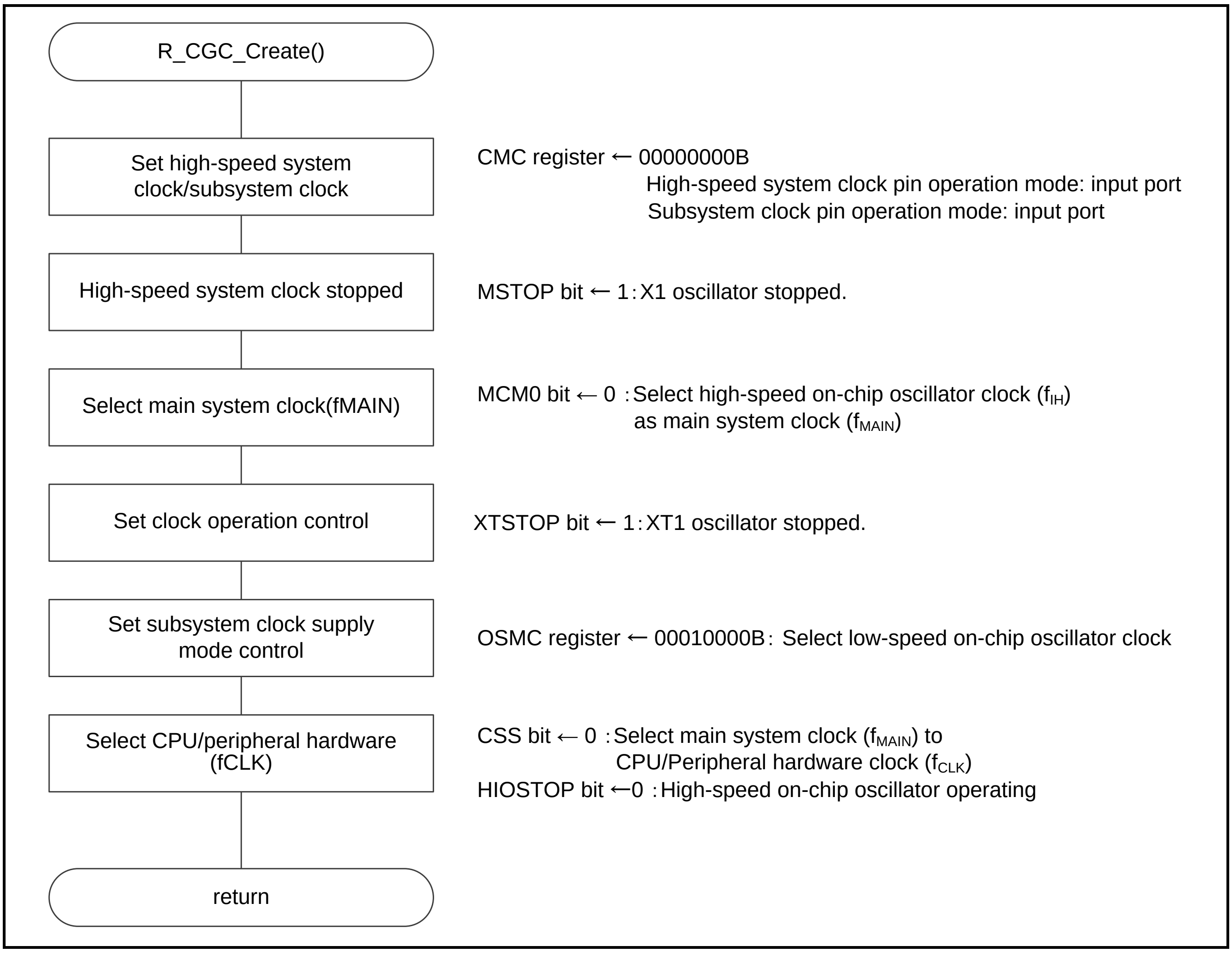

Figure 5.4 CPU Initial Setting

5.7.4 Port Initial Setting

Figure 5.5 shows the flowchart for the port initial setting.

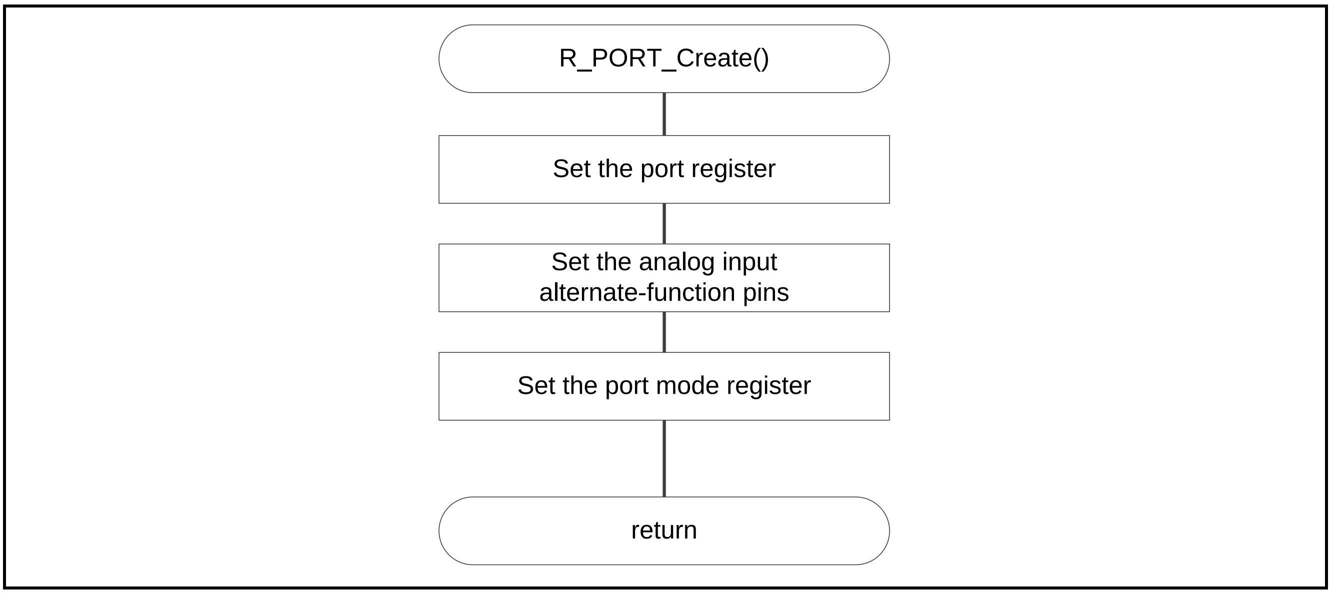

Figure 5.5 Port Initial Setting

Note: Refer to the initialization flowchart in the RL78/G13 Initialization (R01AN2575J) Application Note for details on how to set unused ports.

Caution: When designing circuits, always make sure unused ports are properly processed and all electrical characteristics are met. Also make sure each unused input-only port is connected to V_{DD} or V_{SS} through a resister.

5.7.5 Timer Array Unit Initial Setting

Figure 5.6 shows the flowchart for timer array unit initial setting.

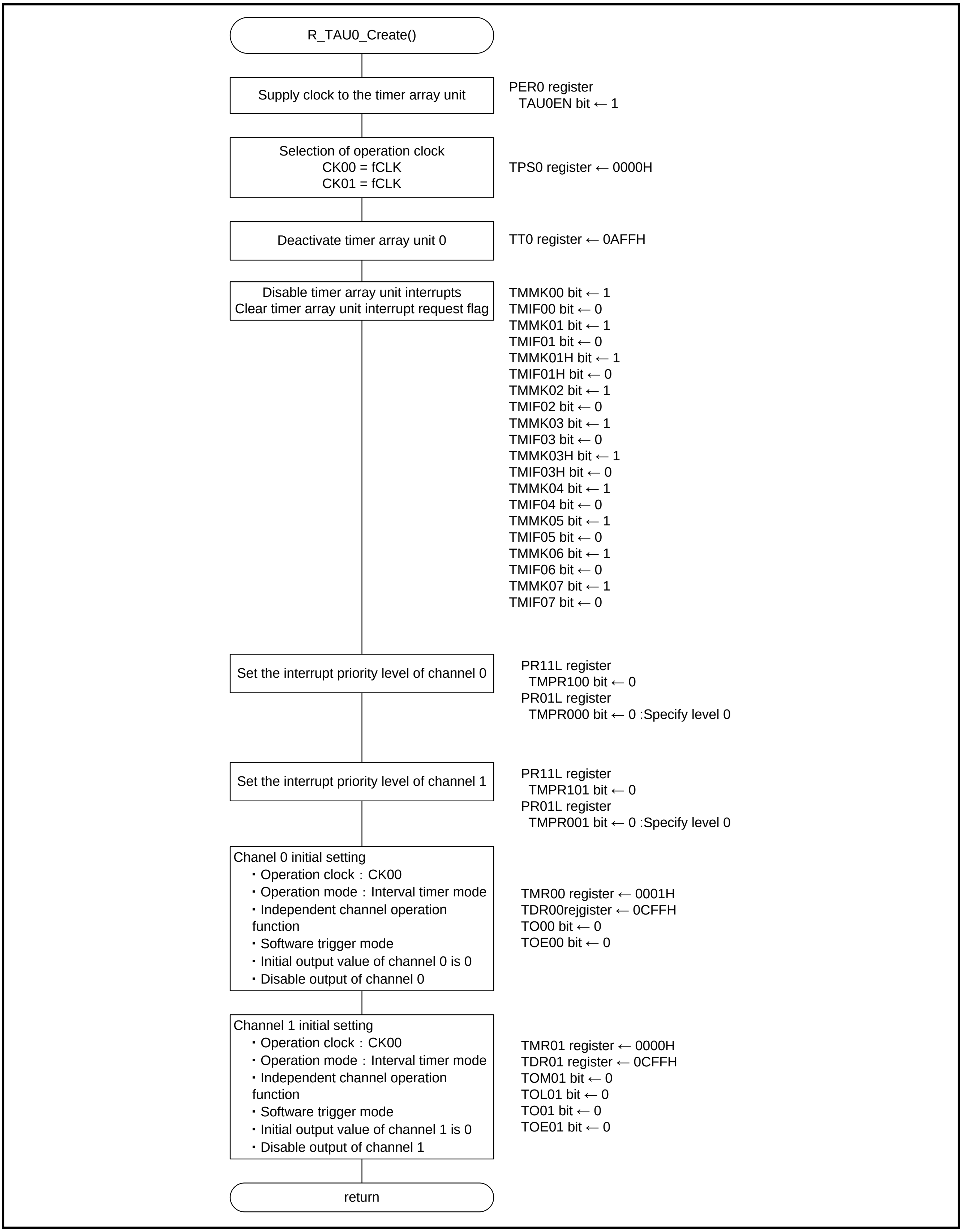

Figure 5.6Timer Array Unit Initial Setting

5.7.6 External Interrupt Initial Setting

Figure 5.7 shows the flowchart for external interrupt initial setting.

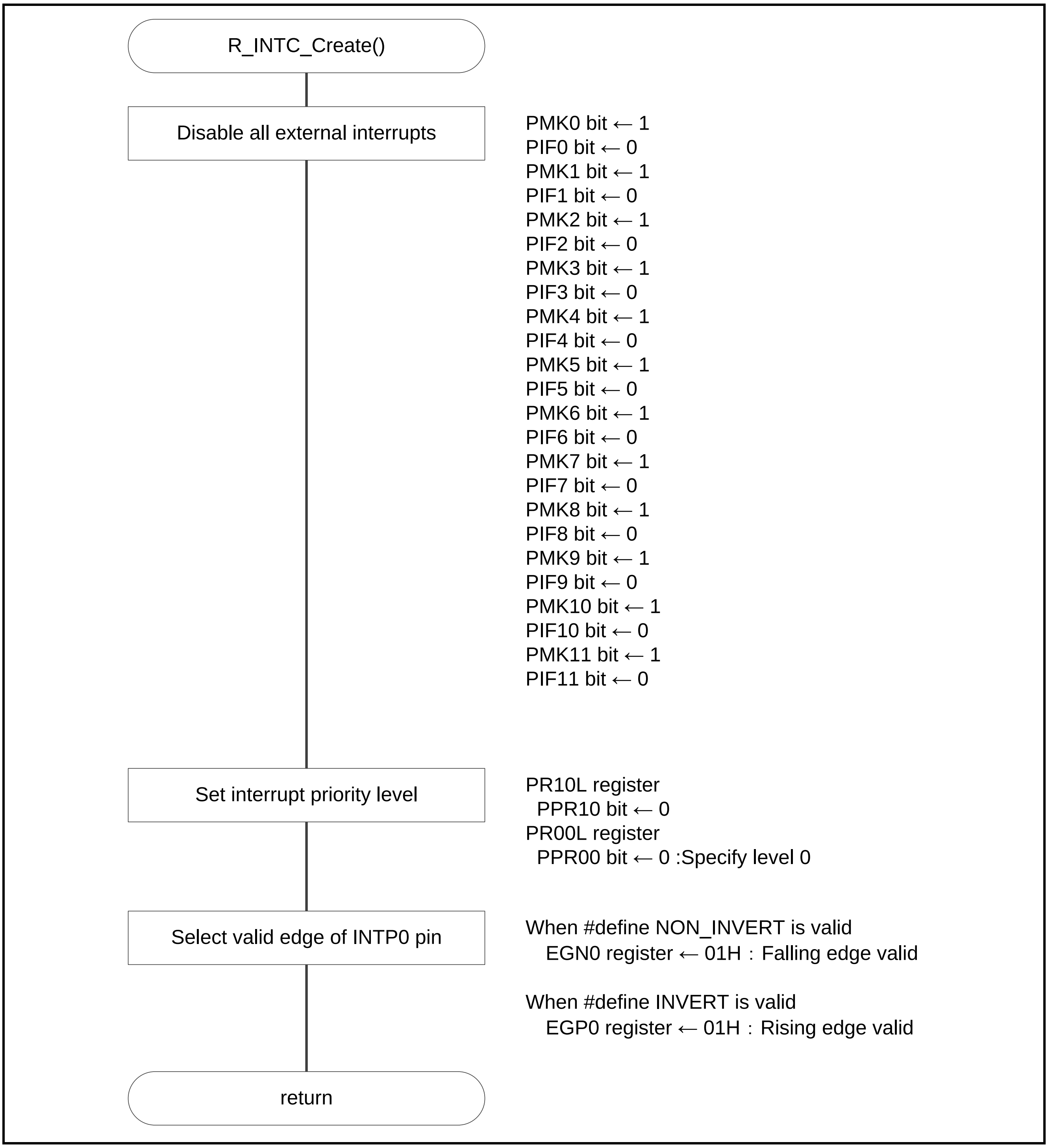

Figure 5.7 External Interrupt Initial Setting

5.7.7 Main Processing

Figure 5.8 and Figure 5.9 shows the flowchart for main processing.

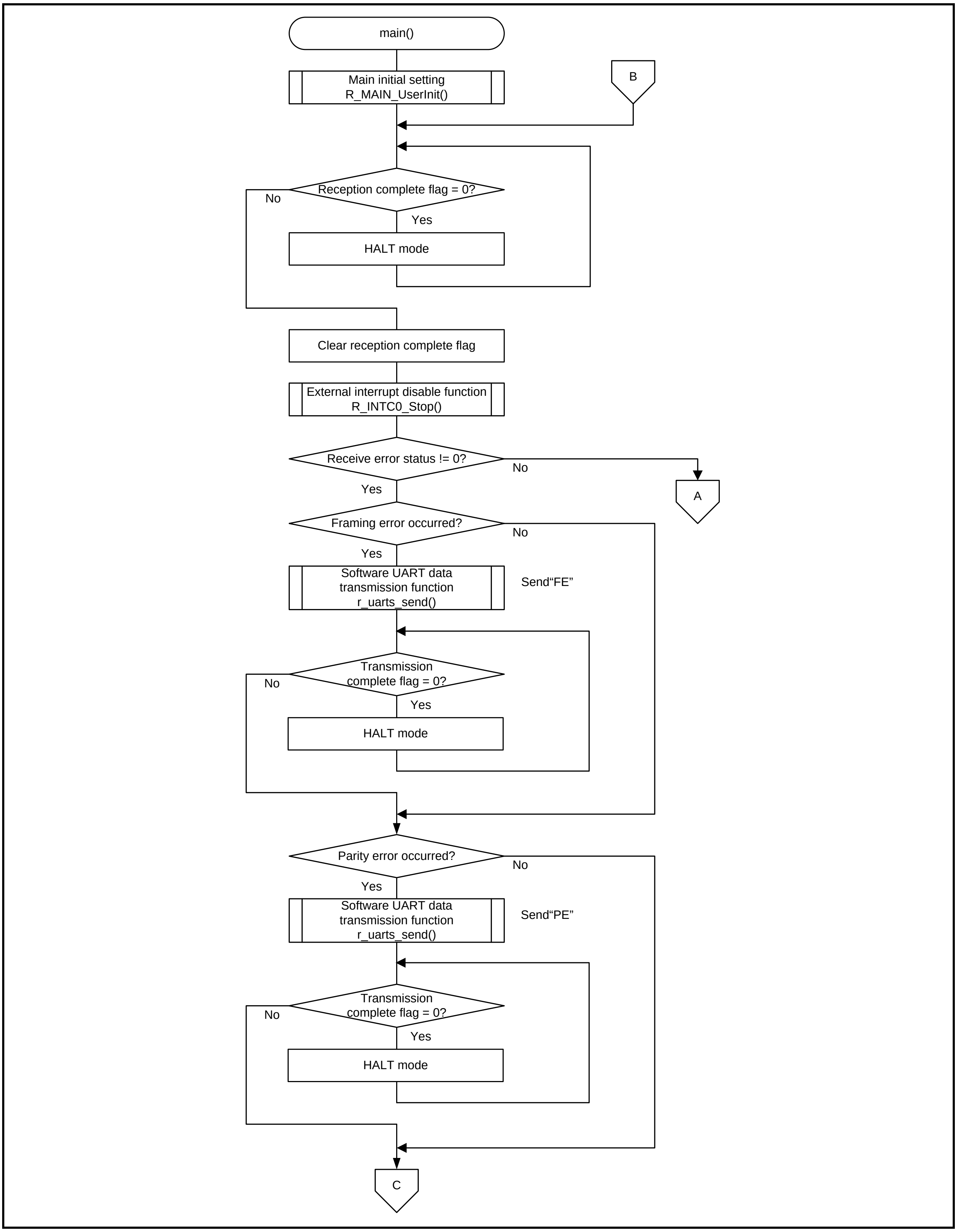

Figure 5.8 Main Processing (1/2)

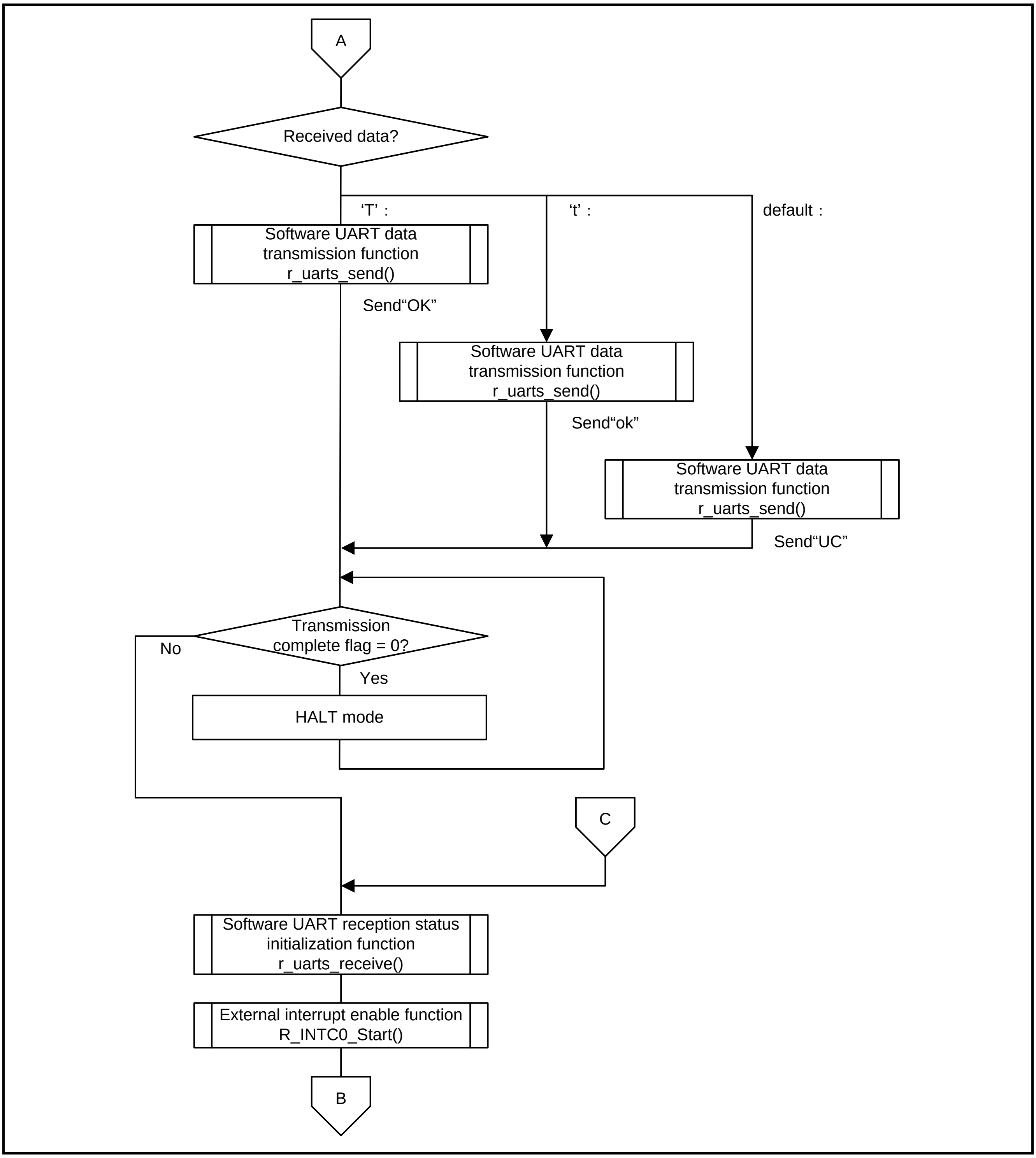

Figure 5.9 Main Processing (2/2)

5.7.8 Main Initial Setting

Figure 5.10 shows the flowchart for main initial setting.

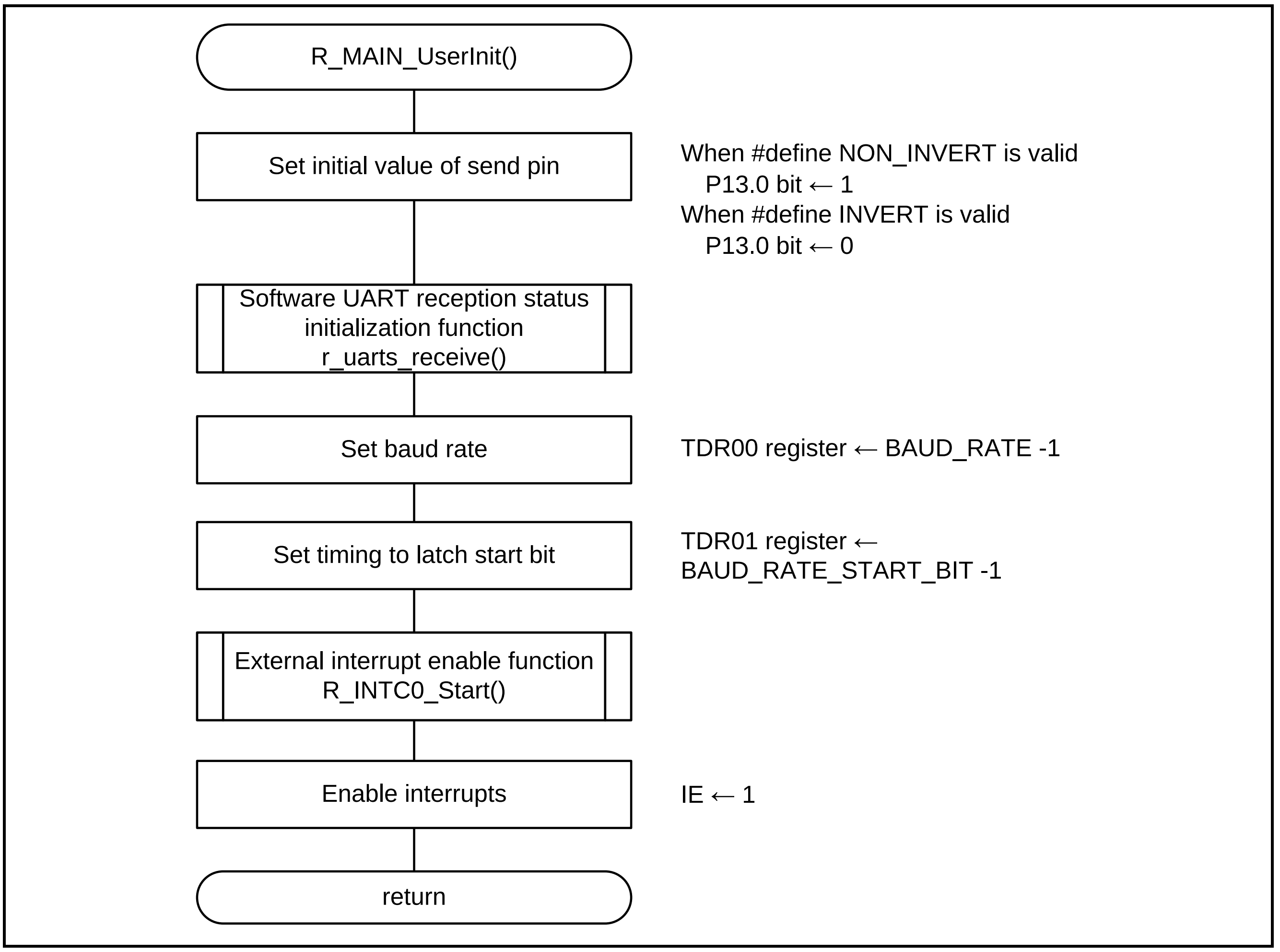

Figure 5.10 Main Initial Setting

5.7.9 Software UART Data Transmission Function

Figure 5.11 shows the flowchart for the software UART data transmission function.

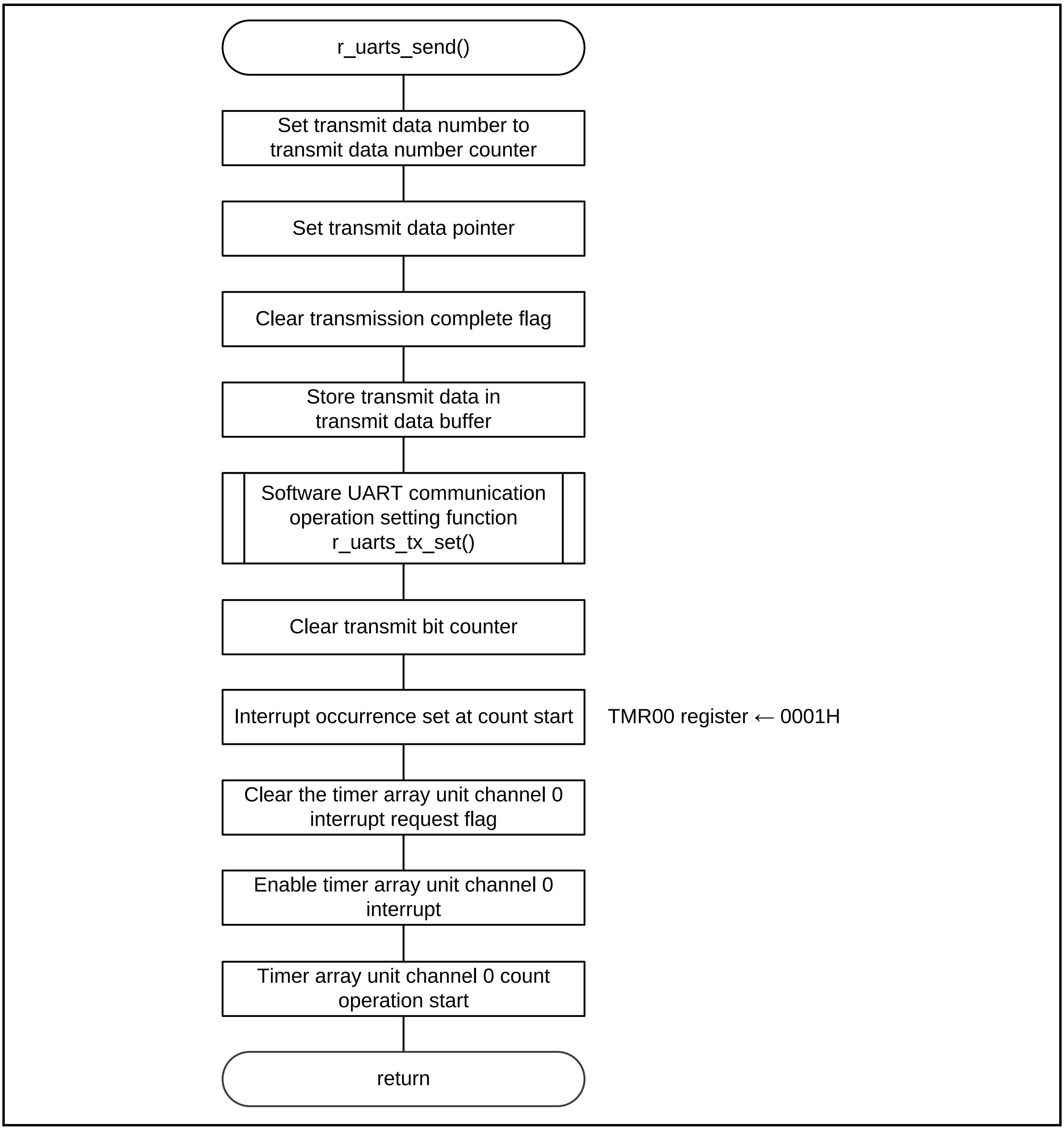

Figure 5.11 Software UART Data Transmission Function

5.7.10 Software UART Communication Operation Setting Function

Figure 5.12 show the flowchart for the software UART communication operation setting function.

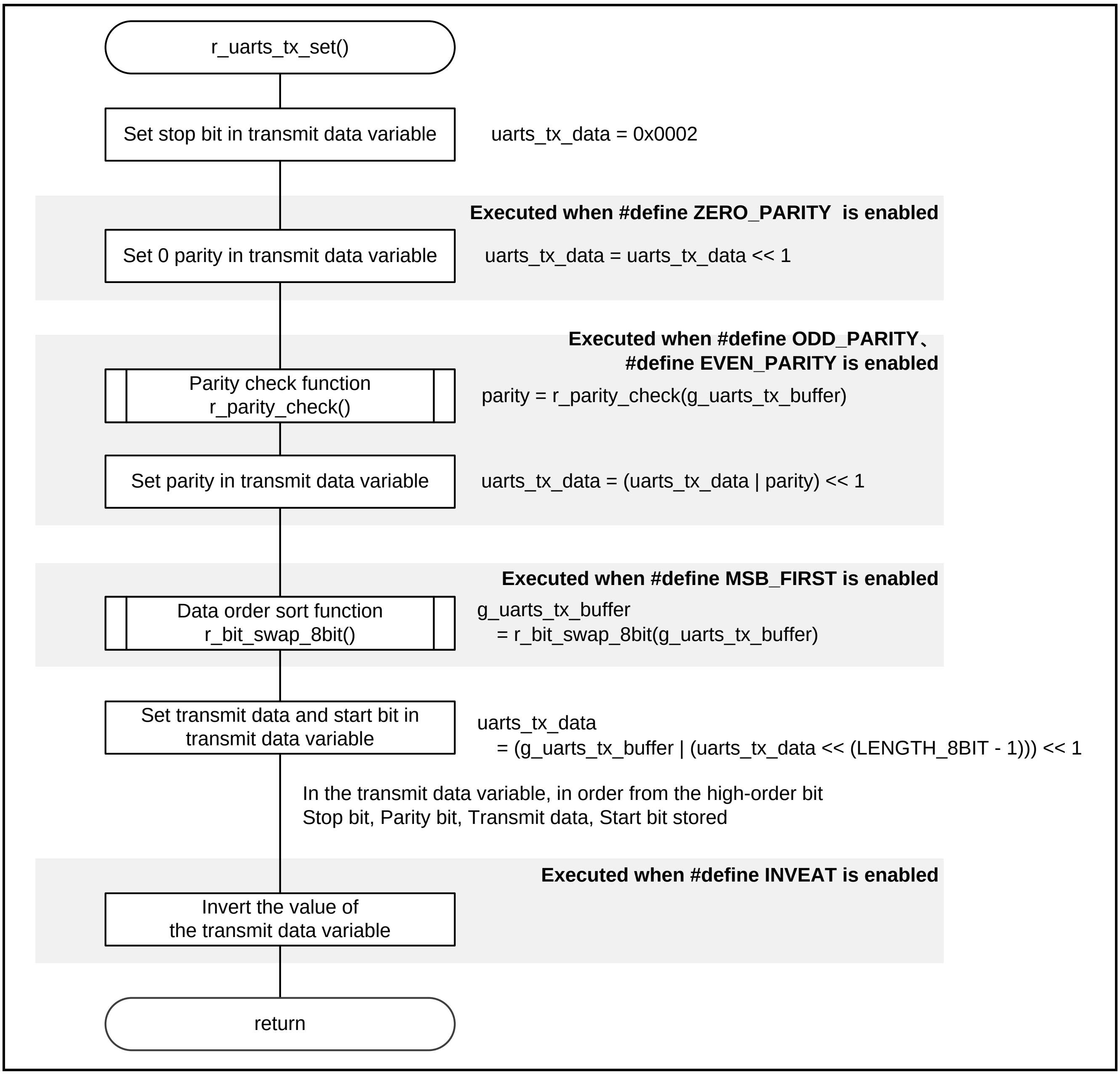

Figure 5.12 Software UART Communication Operation Setting Function

5.7.11 Software UART Reception Status Initialization Function

Figure 5.13 show the flowchart for the software UART reception status initialization function.

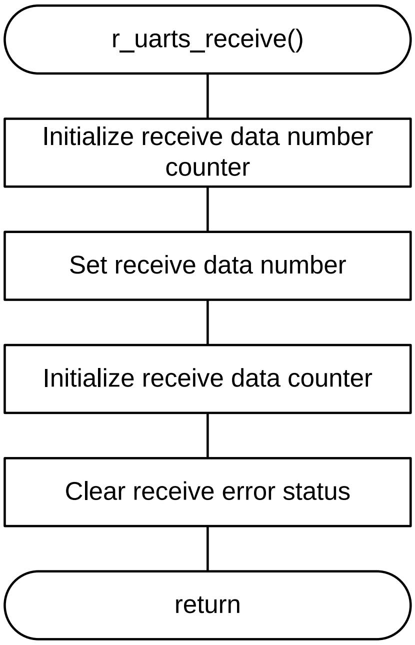

Figure 5.13 Software UART Reception Status Initialization Function

5.7.12 External Interrupt Enable Function

Figure 5.14 show the flowchart for the external interrupt enable function.

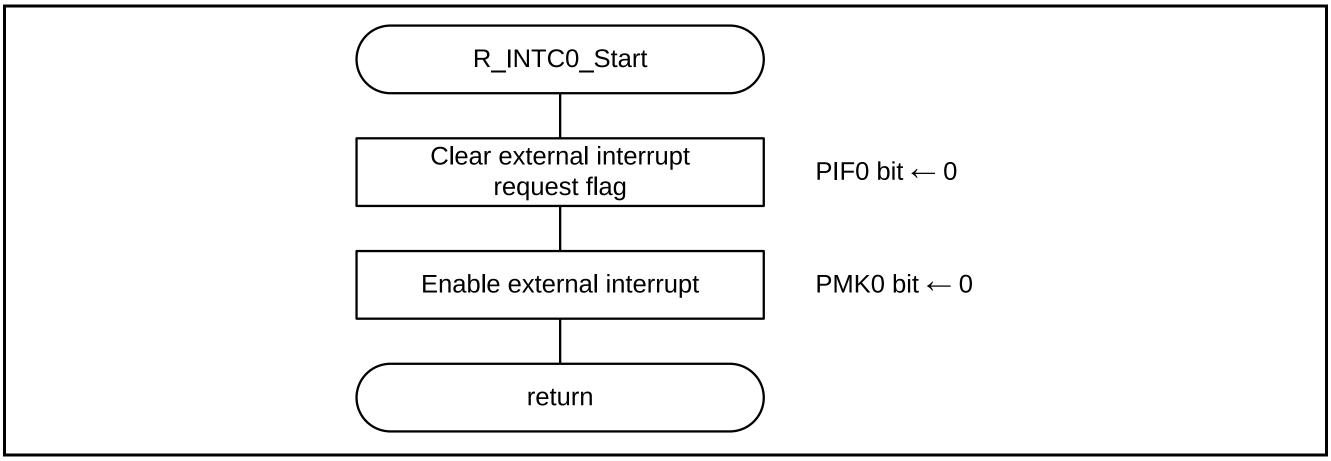

Figure 5.14 External Interrupt Enable Function

5.7.13 External Interrupt Disable Function

Figure 5.15 show the flowchart for the external interrupt disable function.

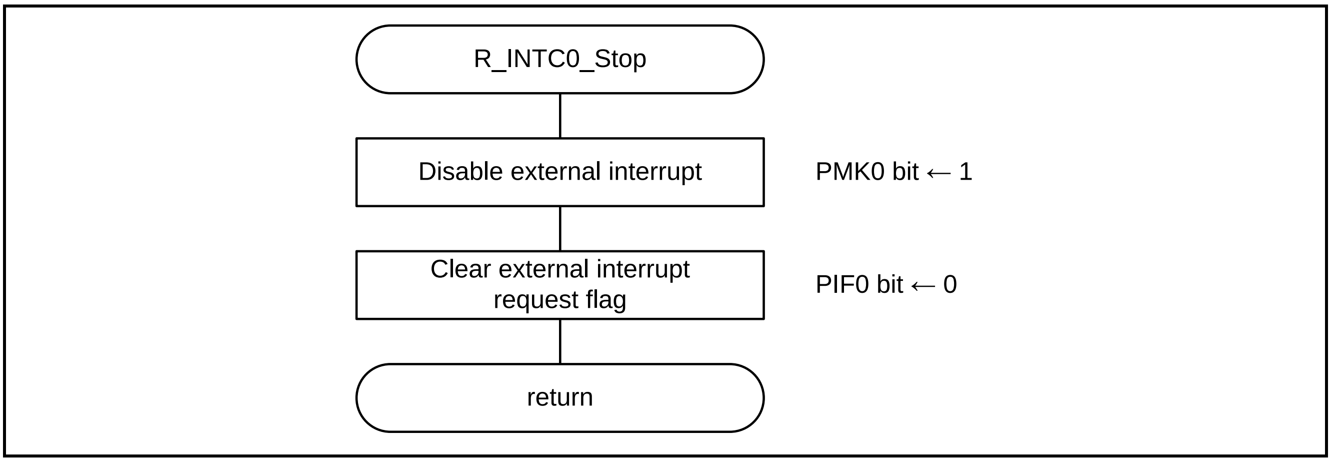

Figure 5.15 External Interrupt Disable Function

5.7.14 Parity Check Function

Figure 5.16 show the flowchart for the parity check function.

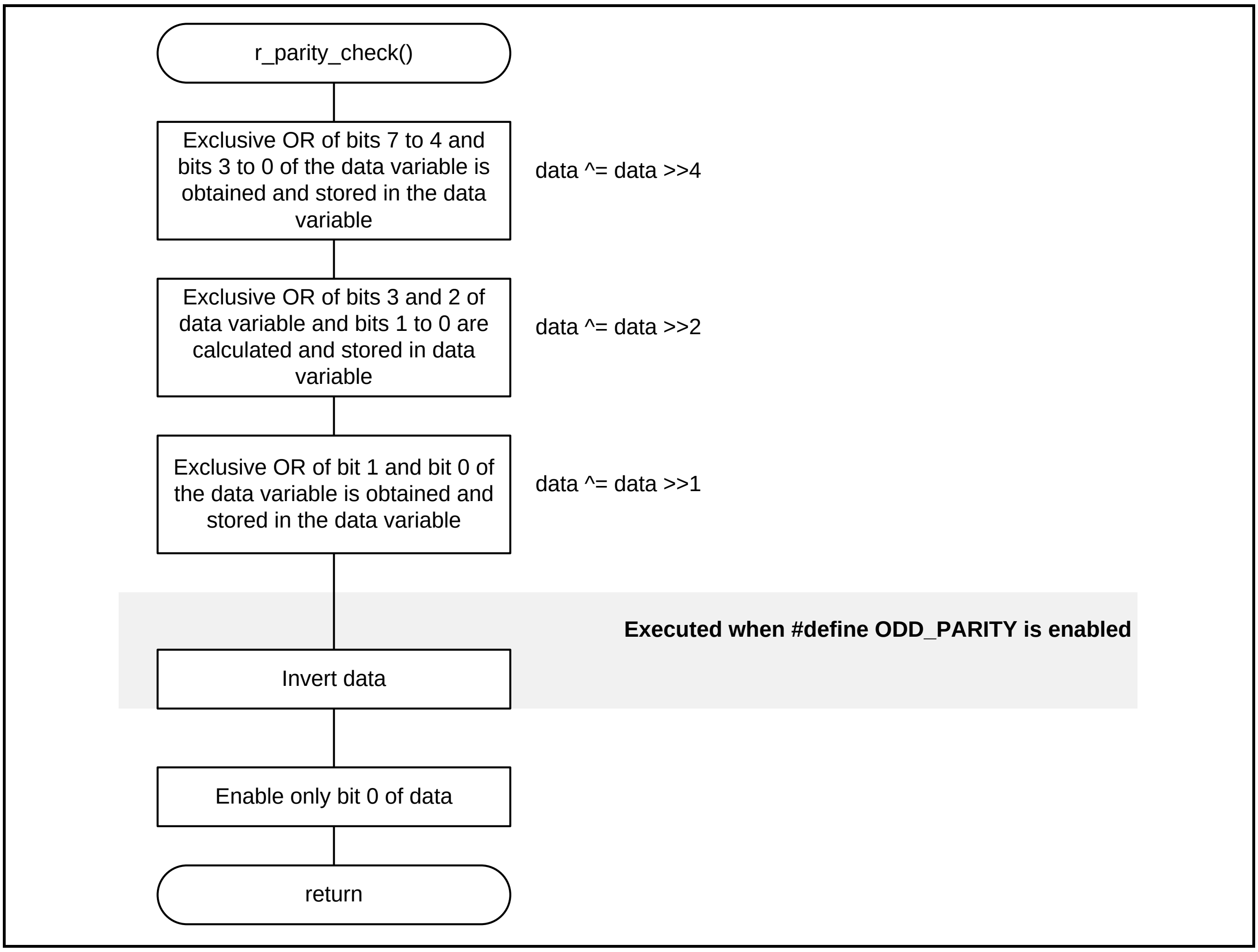

Figure 5.16 Parity Check Function

5.7.15 Data Order Sort Function

Figure 5.17 show the flowchart for the data order sort function.

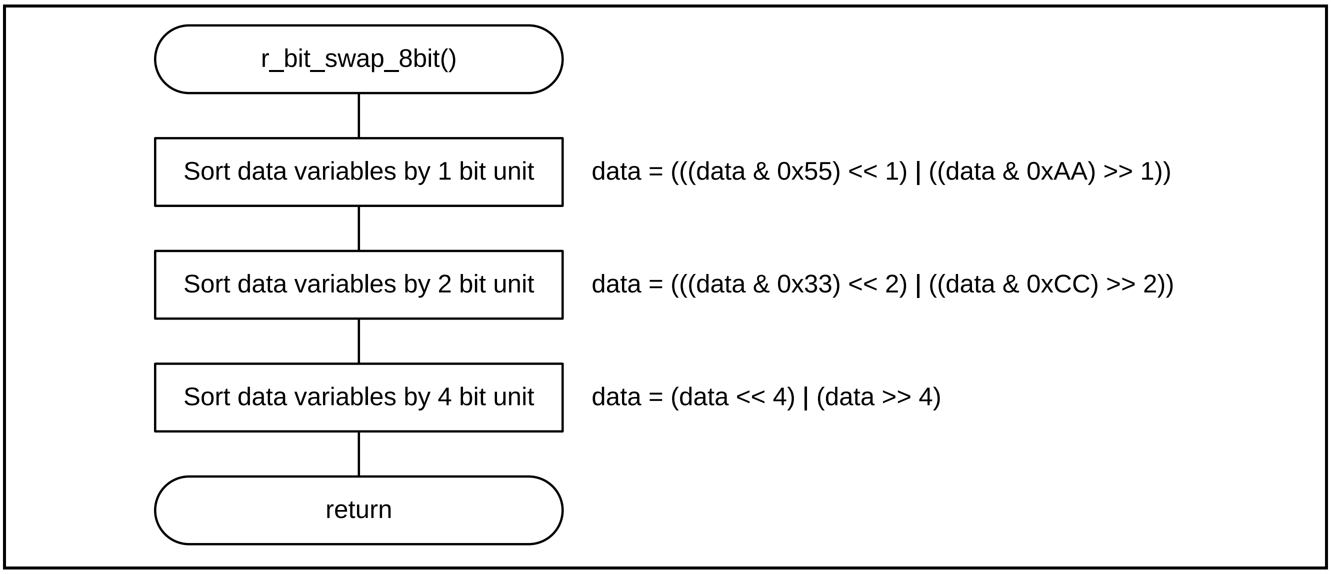

Figure 5.17 Data Order Sort Function

5.7.16 External Interrupt Processing

Figure 5.18 show the flowchart for the external interrupt processing.

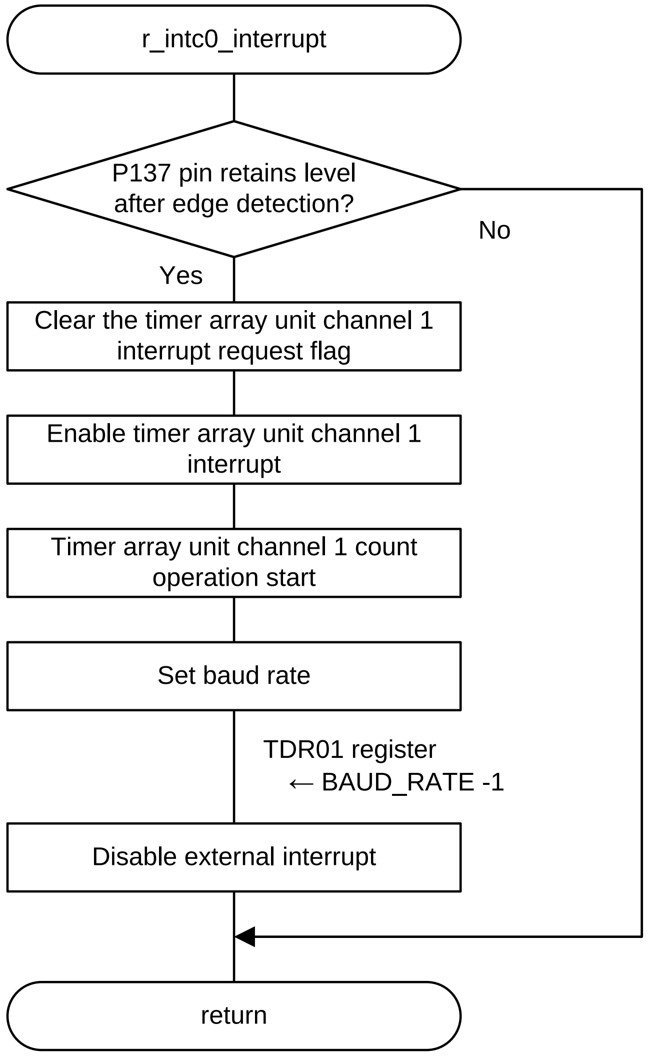

Figure 5.18External Interrupt Processing

5.7.17 Timer Array Unit Channel 0 Interrupt Processing

Figure 5.19 show the flowchart for the timer array unit channel 0 interrupt processing.

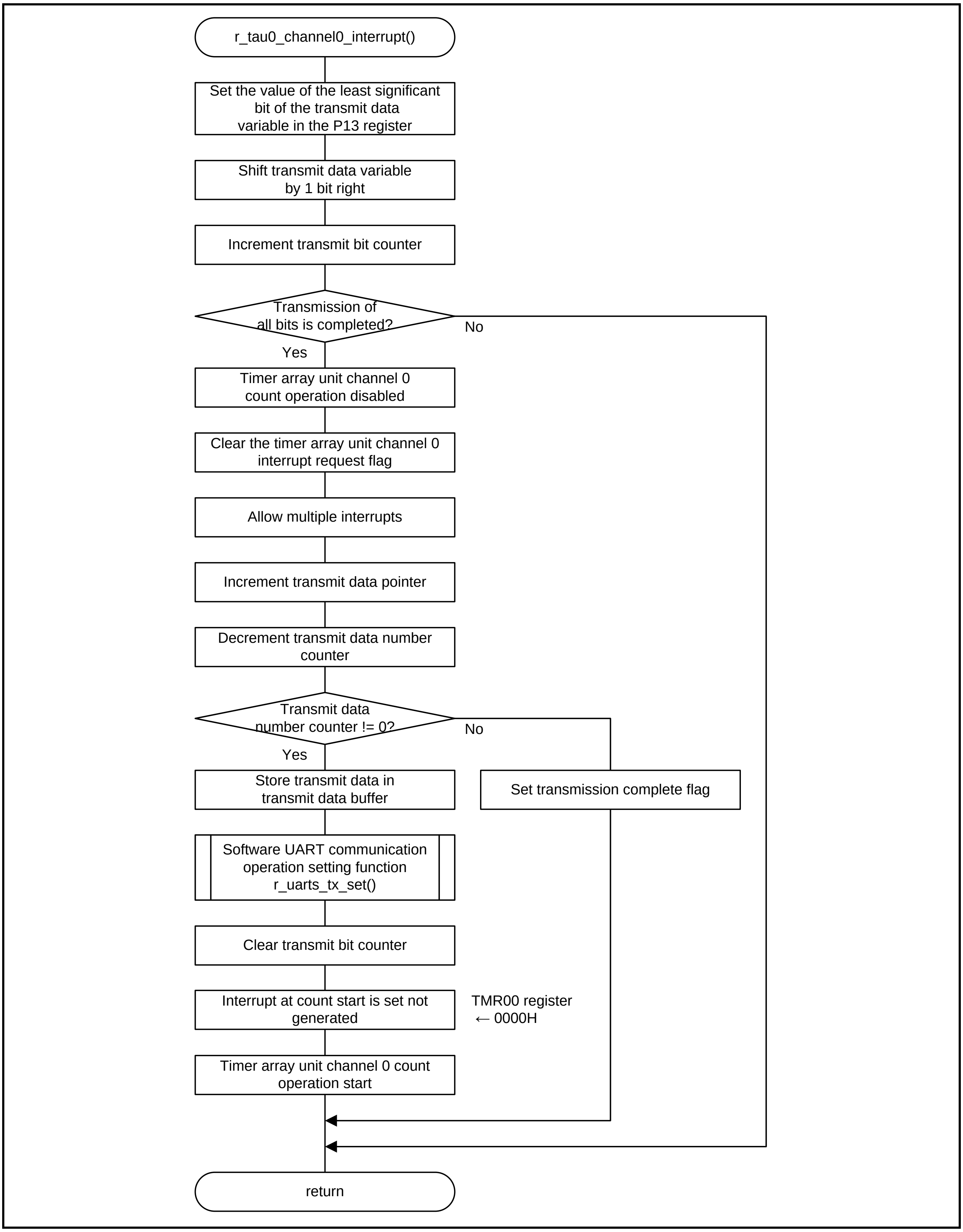

Figure 5.19 Timer Array Unit Channel 0 Interrupt Processing

5.7.18 Timer Array Unit Channel 1 Interrupt Processing

Figure 5.20 to Figure 5.22 show the flowchart for the timer array unit channel 1 interrupt processing.

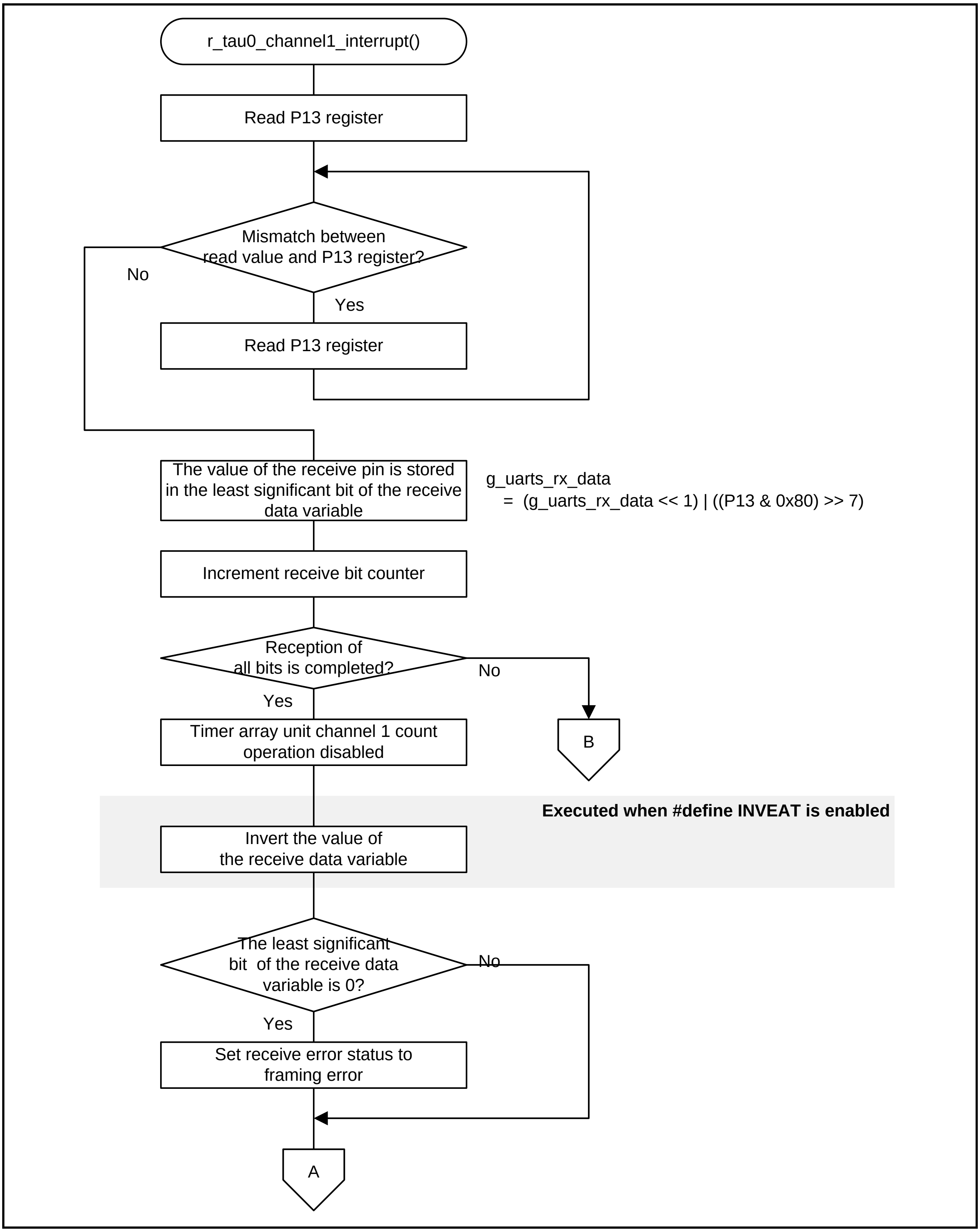

Figure 5.20 Timer Array Unit Channel 0 Interrupt Processing (1/3)

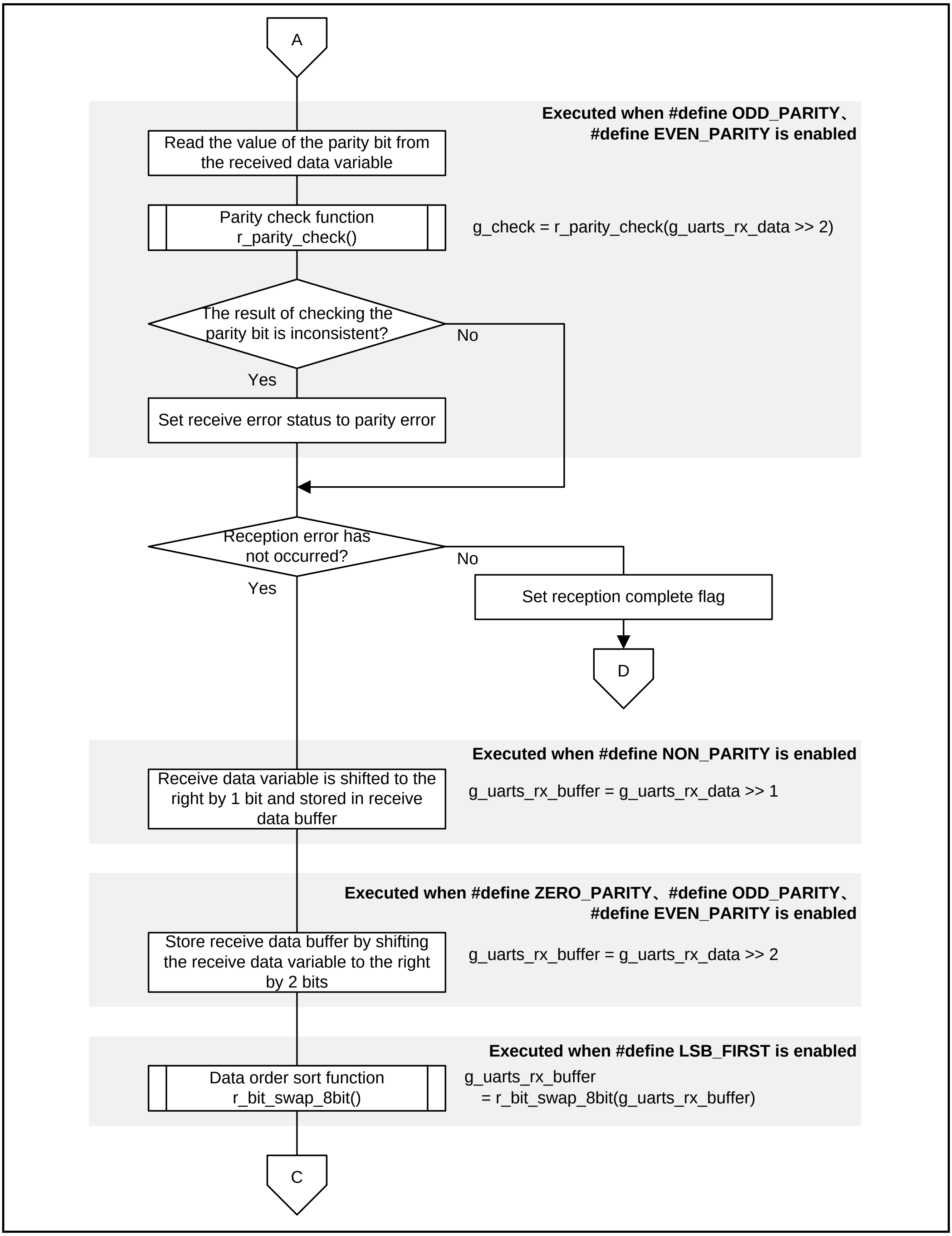

Figure 5.21 Timer Array Unit Channel 0 Interrupt Processing (2/3)

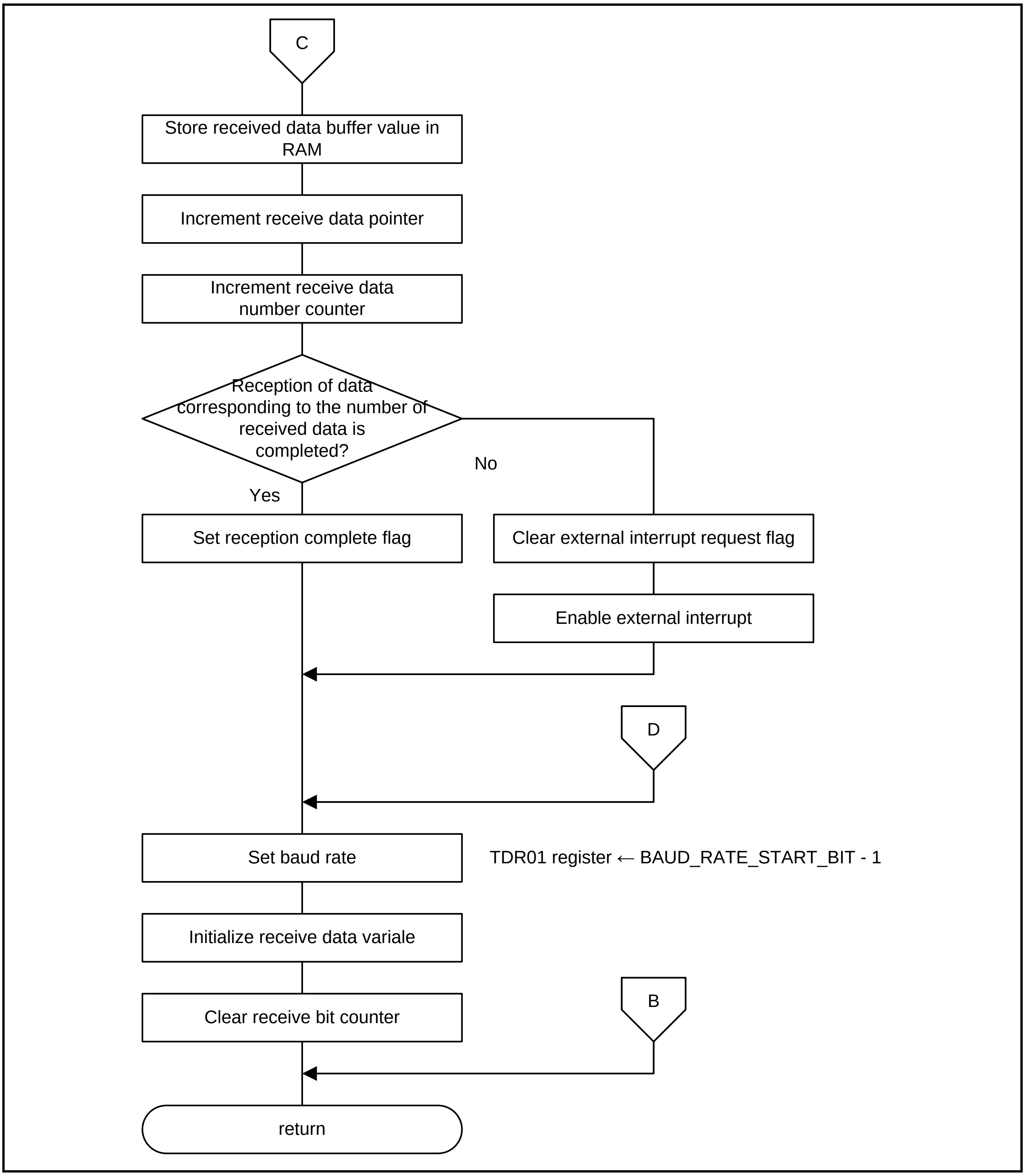

Figure 5.22 Timer Array Unit Channel 0 Interrupt Processing (3/3)

6. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

7. Reference Documents

RL78/G13 User's Manual: Hardware (R01UH0146E)
RL78 Family User's Manual: Software (R01US0015E)
The latest versions can be downloaded from the Renesas Electronics website.

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

Website and Support

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

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

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	May. 7, 2019	—	First edition issued

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

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

1. 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

Trademarks

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/.