R32C/100 Series

Accessing an EEPROM Using Synchronous Serial Interface Mode

R01AN1198EJ0100
Rev. 1.00
Dec. 14, 2012

Abstract

This document describes accessing a Renesas Electronics R1EX25032ASA00A serial EEPROM using synchronous serial interface mode.
The R32C/118 Group MCU has nine channels (UART0 to UART8) that can be used while in synchronous serial interface mode. This application note uses UART2. When using a channel other than UART2, refer to the User's Manual: Hardware and rewrite the registers associated with the corresponding channel.

Products

R32C/116 Group
R32C/117 Group
R32C/118 Group

When using this application note with other Renesas MCUs, careful evaluation is recommended after making modifications to comply with the alternate MCU.

Contents

1. Specifications

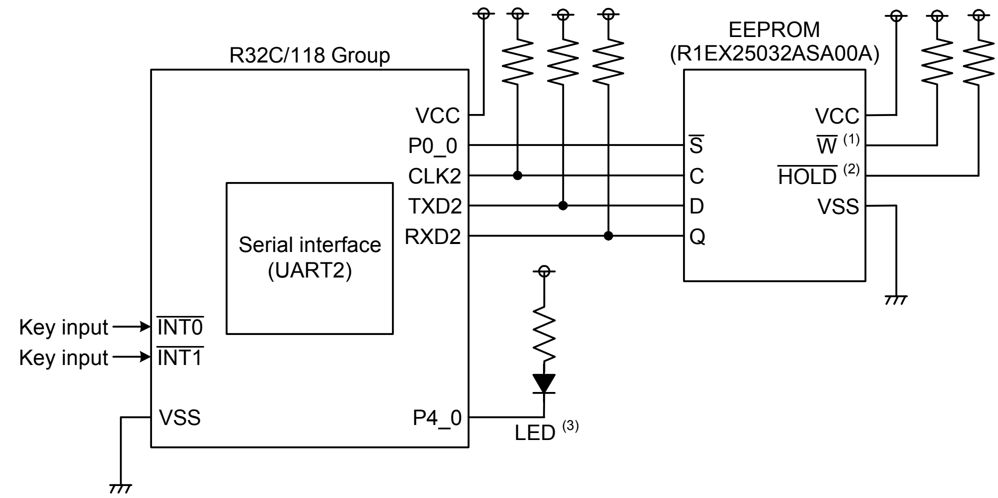
This document describes how to access an EEPROM using synchronous serial interface mode through UART2. After initializing the EEPROM status register (EEPROM initialization), data is written to the EEPROM (EEPROM write processing) and then data is read from the EEPROM (EEPROM read processing).
This application note uses a Renesas Electronics R1EX25xxx Series EEPROM. For details on the EEPROM, refer to the product datasheet.

Conditions for accessing the EEPROM:
- Bit rate: 1 Mbps
- Transfer data length: 1 to 32 bytes (not including the instruction codes or memory address)

Table 1.1 lists the Peripheral Functions and Their Applications. Figure 1.1 shows the Connection Diagram.

Table 1.1 Peripheral Functions and Their Applications

Peripheral Function	Application
Serial interface (UART2)	Communication with the EEPROM
INT0 interrupt	Executes EEPROM write processing
INT1 interrupt	Executes EEPROM read processing
Timer A0	5 ms timer for timeout detection (5 ms is the maximum amount of time needed to write to the EEPROM or write to the EEPROM status register)



Notes:
1. In this application note, EEPROM W pin control is not performed.
2. In this application note, EEPROM HOLD pin control is not performed.
3. The LED is ON when key input can be accepted. The LED is OFF when the EEPROM is being accessed or when there is an EEPROM hardware error.

Figure 1.1 Connection Diagram

2. Operation Confirmation Conditions

The sample code accompanying this application note has been run and confirmed under the conditions below.

Table 2.1 Operation Confirmation Conditions

Item	Contents
MCU used	R5F64189DFD (R32C/118 Group)
Device used	R1EX25032ASA00A
Operating frequencies	• XIN clock: 16 MHz • PLL clock: 100 MHz • Base clock: 50 MHz • CPU clock: 50 MHz • Peripheral bus clock: 25 MHz • Peripheral clock: 25 MHz
Operating voltage	5 V
Integrated development environment	Renesas Electronics High-performance Embedded Workshop Version 4.08
C compiler	Renesas Electronics R32C/100 Series C Compiler V.1.02 Release 01
	Compile options -D__STACKSIZE__=0X300 -D__ISTACKSIZE__=0X300 -DVECTOR_ADR=0x0FFFFFBDC -c -finfo -dir "$(CONFIGDIR)" Default setting is used in the integrated development environment.
Operating mode	Single-chip mode
Sample code version	1.00
Board used	Renesas Starter Kit for R32C/118 (product name: R0K564189S000BE)

3. Reference Application Notes

Application notes associated with this application note are listed below. Refer to these application notes for additional information.

- R32C/100 Series Configuring PLL Mode (REJ05B1221)
- R32C/100 Series Serial Interface Operation in Special Mode 2 Using Master Transmission/Reception (R01AN0493EJ)
- R32C/100 Series Serial Interface Operation (Transmission in Clock Synchronized Serial Interface Mode) (REJ05B1233)
- R32C/100 Series Serial Interface Operation When Receiving Data in Synchronous Serial Interface Mode (R01AN0178EJ)

4. Hardware

4.1 Pins Used

Table 4.1 lists the Pins Used and Their Functions.

Table 4.1 Pins Used and Their Functions

Pin Name	I/O	Function
P0_0	Output	Outputs to control the EEPROM $\overline{\text{S}}$ pin (chip-select pin)
P7_0/TXD2	Output	Outputs serial data to the EEPROM
P7_1/RXD2	Input	Inputs serial data from the EEPROM
P7_2/CLK2	Output	Outputs a clock to set the timing for serial data I/O with the EEPROM
P4_0	Output	Outputs for the LED that indicates the EEPROM is being accessed
P8_2/$\overline{\text{INT0}}$	Input	Executes EEPROM write processing
P8_3/$\overline{\text{INT1}}$	Input	Executes EEPROM read processing

5. Software

In the sample code for this application note, EEPROM initialization is performed when the MCU is reset, 32 bytes of data are written to the EEPROM (EEPROM write processing) (1) when a falling edge is detected on the $\overline{\text{INT0}}$ signal, and 32 bytes of data are read from the EEPROM (EEPROM read processing) (2) when a falling edge is detected on the $\overline{\text{INT1}}$ signal.

Notes:

1. When writing to the EEPROM with user data based on the sample code provided with this application note, designate the EEPROM address and size of the write data so the EEPROM address written does not straddle the page boundaries.
2. When reading the EEPROM with user data based on the sample code provided with this application note, designate the EEPROM address and size of the read data so the EEPROM address read is not larger than the last address.

Table 5.1 lists the instructions for accessing the EEPROM. For details on the instructions, refer to the R1EX25xxx Series datasheet.

Table 5.1 Instructions for Accessing the EEPROM

Instruction	Instruction Format
WREN: Write enable	0000 0110
WRDI: Write disable (This instruction is not used in this application note.)	0000 0100
RDSR: Read the EEPROM status register	0000 0101
WRSR: Write to the EEPROM status register	0000 0001
READ: Read the data	0000 0011
WRITE: Write the data	0000 0010

Figure 5.1 shows the format of the EEPROM status register. Use the RDSR instruction to read the register and use the WRSR instruction to write to the register. For details on the instructions, refer to the R1EX25xxx Series datasheet.

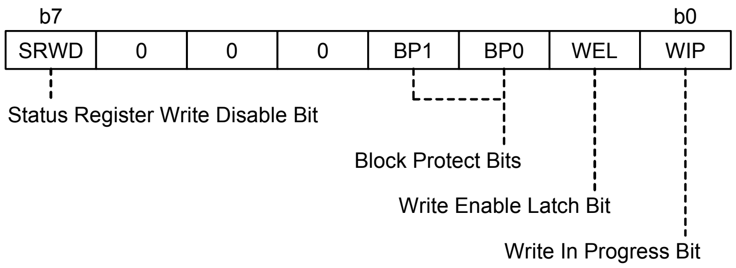


Figure 5.1 EEPROM Status Register Format

Table 5.2 lists the UART2 channel settings used when accessing the EEPROM.

Table 5.2 UART2 Channel Settings Used When Accessing the EEPROM

Mode	Synchronous serial interface mode
Transmit/receive clock	Internal clock
U2BRG count source	f1
TXD2 pin and CLK2 pin output method	Push-pull output
Bit order	MSB first
CTS function	Disabled
CLK polarity	Output transmit data on the falling edge of the transmit/receive clock and input receive data on the rising edge.
Data logic	Not inverted
UART2 transmit interrupt	Used (interrupt source: U2TB register empty)
UART2 receive interrupt	Used
Bit rate	1 Mbps

The formula for calculating the transfer rate is as follows:

$$\text{Transfer rate} = \frac{\text{U2BRG count source}}{(2 \times (\text{U2BRG register setting value} + 1))} = \frac{25\ \text{MHz (f1)}}{(2 \times (11 + 1))} \approx 1\ \text{Mbps}$$

5.1 Operation Overview

5.1.1 Commands for Serial Transfer

In this application note, EEPROM instructions (WREN, WRSR, RDSR, WRITE, and READ) for port operation and serial transfer are performed in minimum units called “commands”. A command performs the following:

(1) Drive port P0_0, which is connected to the $\overline{S}$ pin, low.
(2) Transmit EEPROM instruction.
(3) When necessary, transmit and receive EEPROM instruction parameters.
(4) Set port P0_0 to high.

5.1.2 Command Sequence

In EEPROM initialization, EEPROM write processing, and EEPROM read processing, commands are executed in a predetermined order. In this application note, the order for executing these commands is called the “command sequence”. The command sequence for each process is shown below.

- EEPROM initialization sequence:
  WREN command → RDSR command → WRSR command → RDSR command
- EEPROM write sequence:
  WREN command → RDSR command → WRITE command → RDSR command
- EEPROM read sequence:
  READ command

5.1.3 Procedures for Executing EEPROM Initialization, EEPROM Write Processing, and EEPROM Read Processing

In this application note, synchronous serial interface mode of the serial interface is used to perform half-duplex communication with the EEPROM.

(1) Command sequence start
An MCU reset calls the EEPROM initialization sequence start, an INT0 interrupt calls the function to start the EEPROM write sequence, and an INT1 interrupt calls the function to start the EEPROM read sequence.
The following processes are executed by functions:
- The variable that determines whether or not to proceed to the next command is cleared.
- The command sequence is set.
- The command execution status flag (proc_state) is set to command execution started (STATE_BEGIN).
- The transmit data is stored in the transmit data array (txdata_rw[]) for EEPROM write processing and EEPROM read processing.

(2) Command execution start processing
When the command execution status flag becomes command execution started (STATE_BEGIN), the following processes are executed:
- The variables used in the transmit interrupt and receive interrupt are set.
- Low is output from port P0_0.
- The timer used for timeout detection starts when the RDSR (WIP bit confirmation) command is executed.
- The command execution status flag is set to command executed (STATE_EXECUTE).
- UART2 transmit/receive is enabled.
- EEPROM instructions are transmitted.

(3) Transmitting transmit data
When the data in the U2TB register is transferred to the transmit shift register, a transmit interrupt is generated. In the transmit interrupt handler, 1 byte of transmit data stored in the transmit data array (depending on the command, stored in txdata_rw[], txdata_wren[], txdata_rdsr[], or txdata_wrsr[]) is written to the U2TB register. This process is repeated each time a transmit interrupt is generated until the transmit data has been completely transmitted.

(4) Storing received data
A receive interrupt is generated when the MCU receives data. In the receive interrupt handler, the U2RB register is read and the data is stored in the receive data storage buffer (rxdata_buf[]). After all data has been received, transmission and reception stop, and the command execution status flag is set to command execution complete (STATE_END).

(5) Command execution complete processing
When the command execution status flag becomes command execution complete (STATE_END), the following processes are executed:
- High is output from port P0_0
- When the RDSR command is executed
  The value in the EEPROM status register is stored in the EEPROM status variable (rdsr_val)
- When the READ command is executed
  Data read from the EEPROM is stored in the user designated area.

(6) Processing to proceed to next command
After step (5) is performed, the following processing is executed. Whether or not to proceed to the next command is determined, p_cmd_type is incremented, and the command sequence continues. Also, the command execution status flag is set to command execution started (STATE_BEGIN) in order to start the next command.

Subsequently, steps (2) to (6) are repeated. When all commands are completed, the command execution status flag is set to command not executed (STATE_NON), and the command sequence is completed.

Figure 5.2 shows the Timing Diagram.

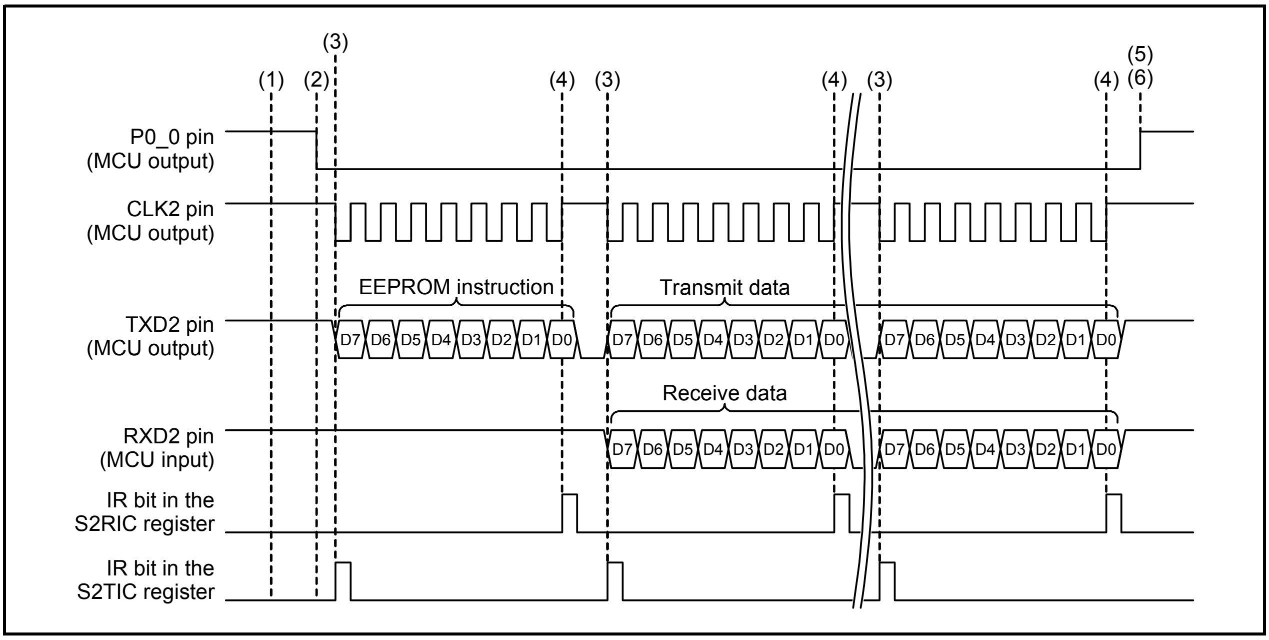


Figure 5.2 Timing Diagram

5.2 Constants

Table 5.3 lists the Constants Used in the Sample Code.

Table 5.3 Constants Used in the Sample Code

Constant Name	Setting Value	Contents
EEP_T_S_HOLD	0x000A	$\overline{S}$ pin set-up and hold wait time
EEP_REG_INI	0x00	Initial setting value for the EEPROM status register
EEP_REG_WEL	0x02	WEL bit in the EEPROM status register
EEP_REG_WIP	0x01	WIP bit in the EEPROM status register
EEPROM_MEM_ADDR	0x0000	Address on the EEPROM when executing the READ command or WRITE command
BUFSIZE	32	Size of the body of the transmit/receive data
TXSIZE	(BUFSIZE+3)	Maximum transmit data size (+3 indicates the EEPROM instruction plus the EEPROM address)
DUMMY	0xff	Dummy data transmitted during data receive processing
UART_BRG	(12 - 1)	Setting value for the UART baud rate

5.3 Structure/Union List

Figure 5.3 shows the Structure/Union Used in the Sample Code.

```
/* **** Port definition **** */
#define EEP_P_S       p0_0           /* Port register for EEPROM S pin control */
#define EEP_D_S       pd0_0          /* Port direction register for EEPROM S pin control */
#define EEP_P_LED     p4_0           /* Port register for the LED */
#define EEP_D_LED     pd4_0          /* Port direction register for the LED */

/* **** Command definition **** */
typedef enum
{
    CMD_WREN,                        /* WREN */
    CMD_RDSR_WEL,                    /* RDSR[Read the WEL bit] */
    CMD_RDSR_WIP,                    /* RDSR[Read the WIP bit] */
    CMD_WRSR,                        /* WRSR */
    CMD_READ,                        /* READ */
    CMD_WRITE,                       /* WRITE */
    CMD_TERMINATE,                   /* All commands complete */
} eeprom_command_t;

/* **** Command execution status definition **** */
typedef enum
{
    STATE_NON,                       /* Command not executed */
    STATE_BEGIN,                     /* Command execution started */
    STATE_EXCECUTE,                  /* Command mid-execution */
    STATE_END,                       /* Command execution completed */
    STATE_ERROR,                     /* Error occurred during command execution */
} command_process_state_t;

/* **** Interrupt request level definition **** */
typedef enum
{
    INT_LEVEL_DISABLE = 0,           /* Interrupts disabled */
    INT_LEVEL_1       = 1,           /* Level 1 */
    INT_LEVEL_2       = 2,           /* Level 2 */
    INT_LEVEL_3       = 3,           /* Level 3 */
    INT_LEVEL_4       = 4,           /* Level 4 */
    INT_LEVEL_5       = 5,           /* Level 5 */
    INT_LEVEL_6       = 6,           /* Level 6 */
    INT_LEVEL_7       = 7,           /* Level 7 */
} interrupt_level_t;

/* **** Port status definition **** */
typedef enum
{
    LEVEL_LOW,                       /* Low level */
    LEVEL_HIGH                       /* High level */
} logic_level_t;
```

Figure 5.3 Structure/Union Used in the Sample Code

5.4 Variables

Table 5.4 lists the Global Variables and Table 5.5 lists the const Variables.

Table 5.4 Global Variables

Type	Variable Name	Contents	Function Used
uint8_t	rdsr_val	EEPROM status variable	variable_clear, cmd_proc_end, cmd_proc_change
uint8_t	txdata_rw[]	Transmit data array when executing a READ or WRITE command	R_EEP_StartWriteSeq, R_EEP_StartReadSeq (1)
uint8_t	rxdata_buf[]	Receive data storage buffer	cmd_proc_begin, cmd_proc_end
uint8_t	comm_data_size_array[]	Transmit data size for each command	R_EEP_StartWriteSeq, R_EEP_StartReadSeq, cmd_proc_begin, cmd_proc_end
uint8_t const *	p_cmd_type	Command pointer	R_EEP_StartInitSeq, R_EEP_StartWriteSeq, R_EEP_StartReadSeq, cmd_proc_begin, cmd_proc_end, cmd_proc_change
uint8_t	proc_state	Command execution status flag	main, R_EEP_StartInitSeq, R_EEP_StartWriteSeq, R_EEP_StartReadSeq, cmd_proc_begin, cmd_proc_change, uart2_rx_interrupt, int0_interrupt, int1_interrupt
bool	f_timeout	Timeout flag	variable_clear, cmd_proc_begin, cmd_proc_change, timer_a0_interrupt
uint8_t *	p_txdata	Transmit data pointer	cmd_proc_begin, uart2_tx_interrupt
uint8_t	remaining_tx_size	Untransmitted data size	cmd_proc_begin, uart2_tx_interrupt
uint8_t *	p_rxdata_buf	Pointer for the receive data storage buffer	cmd_proc_begin, uart2_rx_interrupt
uint8_t	remaining_rx_size	Unreceived data size	cmd_proc_begin, uart2_rx_interrupt
uint8_t *	p_eep_read_data	Pointer for the area where the data read from the EEPROM is stored	R_EEP_StartReadSeq, cmd_proc_end
uint8_t	write_buffer[BUFSIZE]	Area for EEPROM write data	int0_interrupt, main
uint8_t	read_buffer[BUFSIZE]	Area for storing data read from the EEPROM	int1_interrupt

Note:

1. This function is used as an element in the composition of p_tx_data_array[].

Table 5.5 const Variables

Type	Variable Name	Contents	Function Used
const uint8_t	cmd_seq_init[]	Command sequence storage array for EEPROM initialization	R_EEP_StartInitSeq
const uint8_t	cmd_seq_write[]	Command sequence storage array for EEPROM write processing	R_EEP_StartWriteSeq
const uint8_t	cmd_seq_read[]	Command sequence storage array for EEPROM read processing	R_EEP_StartReadSeq
const uint8_t	txdata_wren[]	Transmit data array when using WREN	See Note 1.
const uint8_t	txdata_rdsr[]	Transmit data array when using RDSR	See Note 1.
const uint8_t	txdata_wrsr[]	Transmit data array when using WRSR	See Note 1.
const uint8_t * const	p_tx_data_array[]	Pointer for the transmit data arrays corresponding to each command	cmd_proc_begin

Note:

1. This function is used as an element in the composition of p_tx_data_array[].

5.5 Functions

Table 5.6 lists the Functions.

Table 5.6 Functions

Function Name	Outline
main	Main processing
R_EEP_StartIntSeq	EEPROM initialization sequence start
R_EEP_StartWriteSeq	EEPROM write sequence start
R_EEP_StartReadSeq	EEPROM read sequence start
variable_clear	Clear variable of proceed to next command determination
cmd_proc_begin	Command execution start processing
cmd_proc_end	Command execution complete processing
cmd_proc_change	Processing to proceed to next command
eep_s_control	Controlling the EEPROM $\overline{\mathrm{S}}$ pin
uart2_tx_interrupt	UART2 serial transmission (UART2 transmit interrupt)
uart2_rx_interrupt	UART2 serial reception (UART2 receive interrupt)
timer_a0_interrupt	5 ms timeout processing (timer A0 interrupt)
int0_interrupt	Accepting an EEPROM write request (INT0 interrupt)
int1_interrupt	Accepting an EEPROM read request (INT1 interrupt)
timer_a0_init	Timer A0 initialization
uart2_init	UART2 initialization

5.6 Function Specifications

The following tables list the sample code function specifications.

main	
Outline	Main processing
Header	None
Declaration	void main(void)
Description	This function performs the initial settings for the system clock, UART2, and timer A0, and initializes the variables. Then, EEPROM initialization sequence is started and the MCU enters the main loop. In the main loop, the command execution start processing, command execution complete processing, and processing to proceed to the next command are performed according to the command execution status flag.
Argument	None
Returned value	None

R_EEP_StartInitSeq	
Outline	EEPROM initialization sequence start
Header	None
Declaration	void R_EEP_StartInitSeq(void)
Description	The command sequence is performed in the EEPROM initialization sequence, and the command execution status flag is set to command execution started (STATE_BEGIN).
Argument	None
Returned value	None

R_EEP_StartWriteSeq	
Outline	EEPROM write sequence start
Header	None
Declaration	void R_EEP_StartWriteSeq(uint8_t *p_write_data, uint8_t size, uint16_t eeprom_adrs)
Description	This function sets the WRITE command transmit data size and generates serial data. Then, the command sequence is set in the EEPROM write processing, and the command execution status flag is set to command execution start (STATE_BEGIN).
Argument	uint8_t *p_write_data: Pointer for the area where the write data is stored uint8_tsize: Size of the write data (in bytes) uint16_t eeprom_adrs: EEPROM address
Returned value	None

R_EEP_StartReadSeq	
Outline	EEPROM read sequence start
Header	None
Declaration	void R_EEP_StartReadSeq(uint8_t *p_read_data, uint8_t size, uint16_t eeprom_adrs)
Description	This function sets the READ command transmit data size and generates serial data. It also sets the pointer (p_eep_read_data) for the area where the data read from the EEPROM is stored. Then, the command sequence is set in the EEPROM read processing, and the command execution status flag is set to command execution start (STATE_BEGIN).
Argument	uint8_t *p_read_data: Pointer for the area where the read data is stored uint8_tsize: Size of the read data (in bytes) uint16_t eeprom_adrs: EEPROM address
Returned value	None

variable_clear	
Outline	Clear variable of proceed to next command determination
Header	None
Declaration	void variable_clear(void)
Description	This function clears the variable used in determining command transition.
Argument	None
Returned value	None

cmd_proc_begin	
Outline	Command execution start processing
Header	None
Declaration	void cmd_proc_begin(void)
Description	This function sets the variable for the serial transmit/receive processing, and after the $\overline{S}$ pin, which is connected to the P0_0 port, is driven low. Also, when executing the RDSR command (read the WIP bit), the timer for the timeout detection starts. Then, the command execution status flag is set to command execution in progress (STATE_EXECUTE), UART2 is transmit/receive enabled, and the first byte of data is transmitted.
Argument	None
Returned value	None

cmd_proc_end	
Outline	Command execution complete processing
Header	None
Declaration	void cmd_proc_end(void)
Description	This function drives $\overline{S}$ pin, which is connected to the P0_0 port, high. When executing the RDSR command, the value received from the EEPROM status register is set to the EEPROM status variable. When executing the READ command, the EEPROM data received is stored in the user designated area.
Argument	None
Returned value	None

cmd_proc_change	
Outline	Processing to proceed to next command
Header	None
Declaration	void cmd_proc_change(void)
Description	This function performs proceed to next command determination and proceed to next command. • Proceed to next command determination: This function reads the EEPROM status register and confirms that a timeout has occurred. This function also determines whether or not to proceed to the next command, and determines if an error occurred during command execution. • Proceed to next command: <u>No error occurred during command execution and the command sequence proceeds to the next command</u> Increment p_cmd_type and continue the command sequence. When the command becomes all commands complete (CMD_TERMINATE), the command execution status flag becomes command not executed (STATE_NON). In all other cases, the command execution status flag becomes command execution started (STATE_BEGIN). <u>No error occurred during command execution and the command sequence does not proceed to the next command</u> The command execution status flag is set to command execution started (STATE_BEGIN). <u>An error occurred during command execution</u> The command execution status flag is set to error occurred during command execution (STATE_ERROR). Also, in the sample code accompanying this application note, there is no processing to transition out of the STATE_ERROR state.
Argument	None
Returned value	None

eep_s_control	
Outline	Controlling the EEPROM $\overline{S}$ pin
Header	None
Declaration	void eep_s_control(logic_level_t level)
Description	High or low is output from port P0_0 to the EEPROM $\overline{S}$ pin.
Argument	logic_level_t level: Output level is set LEVEL_LOW: Low level LEVEL_HIGH: High level
Returned value	None

uart2_tx_interrupt	
Outline	UART2 serial transmission (UART2 transmit interrupt)
Header	None
Declaration	void uart2_tx_interrupt(void)
Description	This function is called by the UART2 transmit interrupt that occurs when the U2TB register becomes empty. If there is untransmitted data, 1 byte of the untransmitted data is transmitted.
Argument	None
Returned value	None

uart2_rx_interrupt	
Outline	UART2 serial reception (UART2 receive interrupt)
Header	None
Declaration	void uart2_rx_interrupt(void)
Description	This function is called by the UART2 receive interrupt that occurs when reception is completed. After reading the UART2 receive buffer (U2RB), this function stores the data in the receive data storage buffer (rxdata_buf[]). When all data is received, UART2 transmission/reception is disabled, and the command execution status flag is set to command execution completed (STATE_END).
Argument	None
Returned value	None

timer_a0_interrupt	
Outline	5 ms timeout processing (timer A0 interrupt)
Header	None
Declaration	void timer_a0_interrupt(void)
Description	After starting timer A0 and 5 ms elapse, this function is called by the timer A0 interrupt that occurs. This function sets the timeout flag (f_timeout) and stops the timer for timeout detection.
Argument	None
Returned value	None

int0_interrupt	
Outline	Accepting an EEPROM write request (INT0 interrupt)
Header	None
Declaration	void int0_interrupt(void)
Description	This function is called by the INT0 interrupt that occurs when a low is input to port P8_2. This function executes the EEPROM write sequence start (R_EEP_StartWriteSeq).
Argument	None
Returned value	None

int1_interrupt	
Outline	Accepting an EEPROM read request (INT1 interrupt)
Header	None
Declaration	void int1_interrupt(void)
Description	This function is called by the INT1 interrupt that occurs when a low is input to port P8_3. This function executes the EEPROM read sequence start (R_EEP_StartReadSeq).
Argument	None
Returned value	None

timer_a0_init	
Outline	Timer A0 initialization
Header	None
Declaration	void timer_a0_init(void)
Description	This function sets timer A0 to timer mode.
Argument	None
Returned value	None

uart2_init	
Outline	UART2 initialization
Header	None
Declaration	void uart2_init(void)
Description	This functions sets the UART2 channel to synchronous serial interface mode.
Argument	None
Returned value	None

5.7 Flowcharts

5.7.1 Main Processing

Figure 5.4 shows the Main Processing.

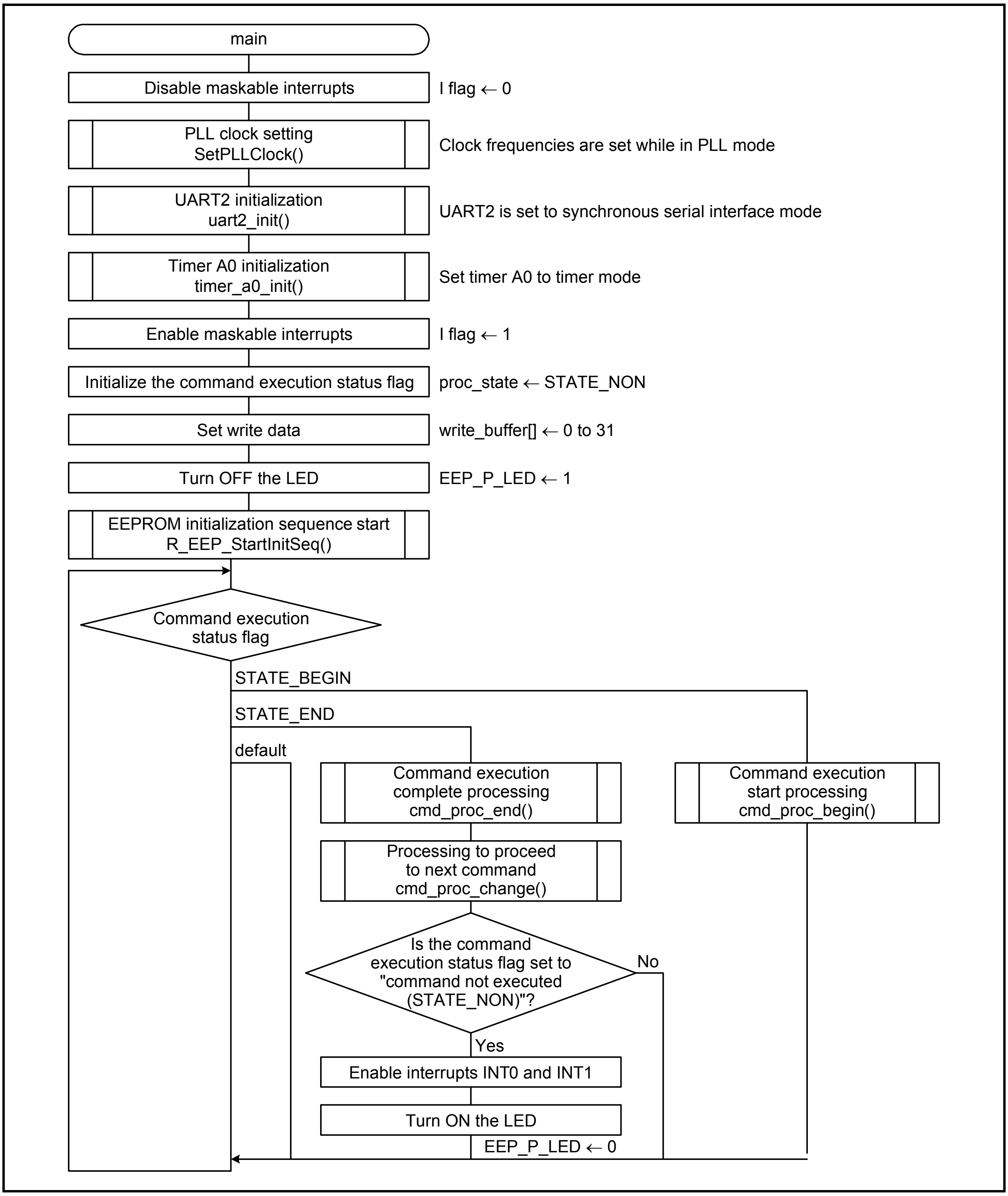


Figure 5.4 Main Processing

5.7.2 EEPROM Initialization Sequence Start

Figure 5.5 shows the start of the EEPROM initialization sequence.

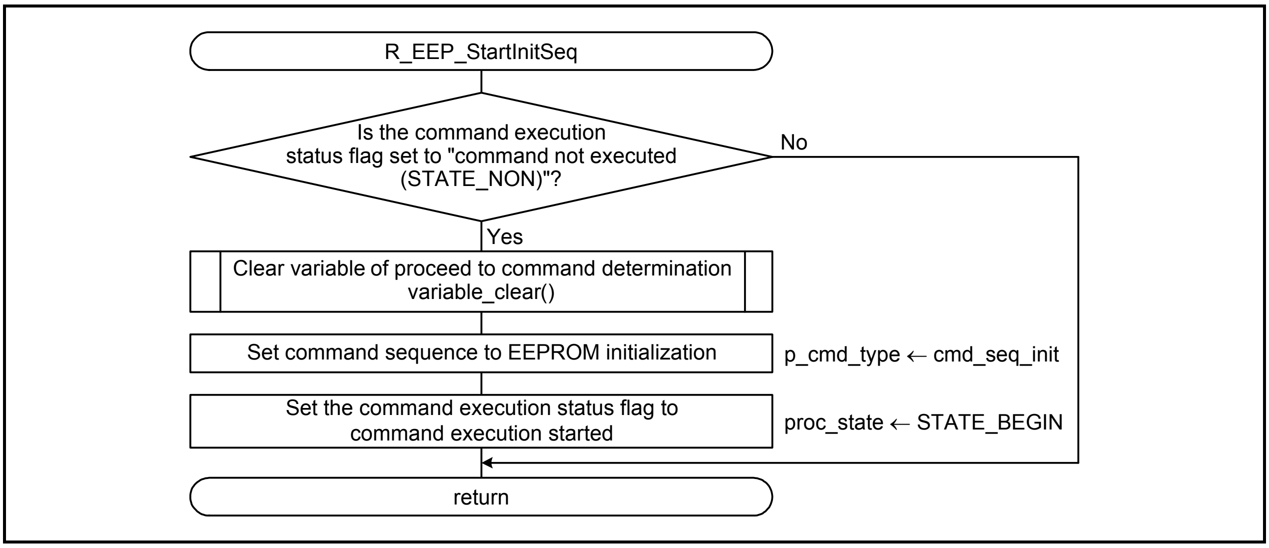


Figure 5.5 EEPROM Initialization Sequence Start

5.7.3 EEPROM Write Sequence Start

Figure 5.6 shows the start of the EEPROM write sequence.

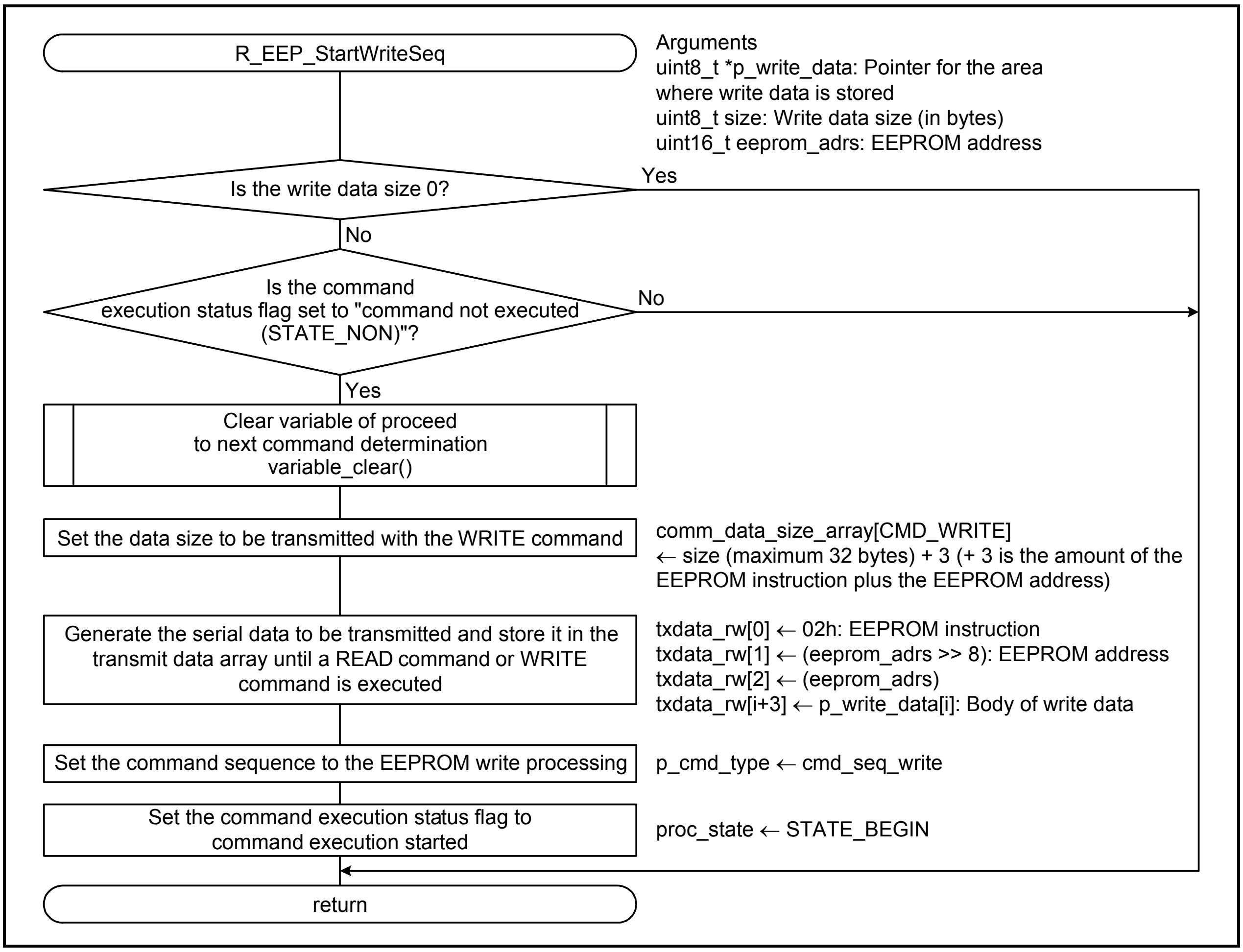


Figure 5.6 EEPROM Write Sequence Start

5.7.4 EEPROM Read Sequence Start

Figure 5.7 shows the start of the EEPROM read sequence.

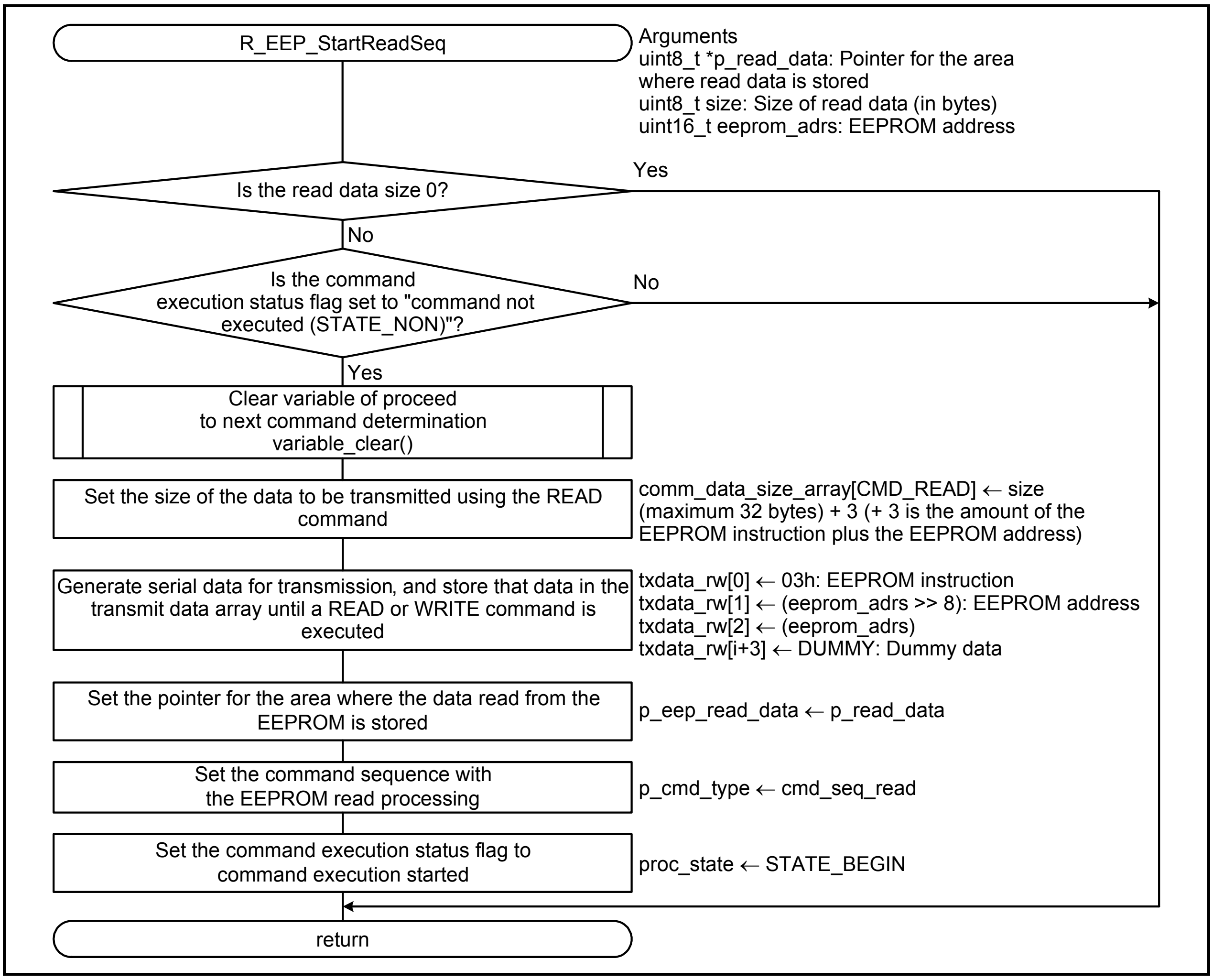


Figure 5.7 EEPROM Read Sequence Start

5.7.5 Clear Variable of Proceed to Next Command Determination

Figure 5.8 shows how to clear the variable of the proceed to the next command determination.

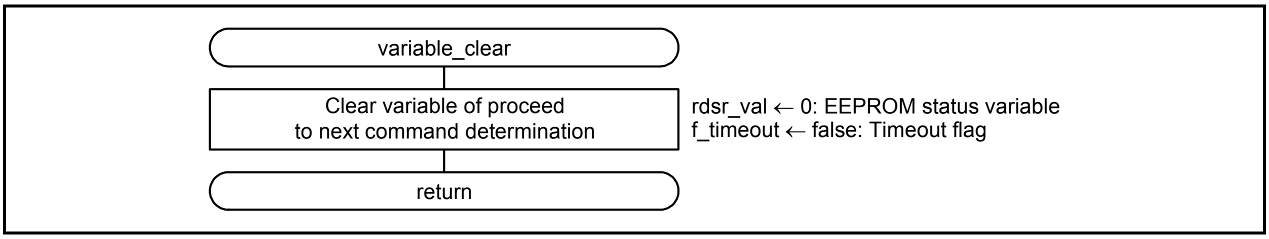
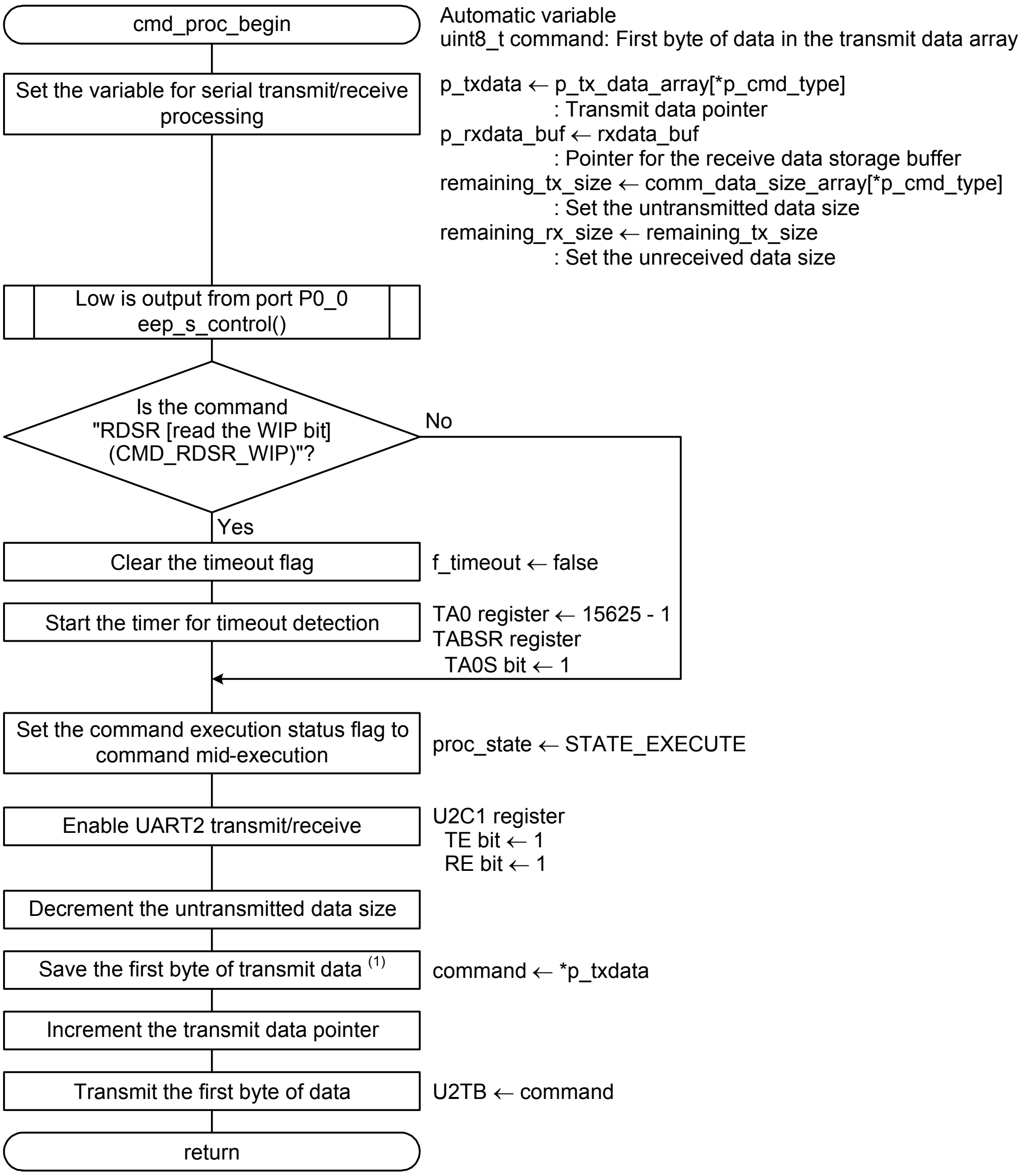


Figure 5.8 Clear Variable of Proceed to Next Command Determination

5.7.6 Command Execution Start Processing

Figure 5.9 shows the command execution start processing.



Note:

1. If the first byte of data is written directly to the UART transmit buffer, a transmit interrupt is generated before the pointer indicated by the transmit buffer is updated, causing the pointer to become misaligned. To prevent this from happening, the first byte of transmit data should be saved to the automatic variable first.

Figure 5.9 Command Execution Start Processing

5.7.7 Command Execution Complete Processing

Figure 5.10 shows the command execution complete processing.

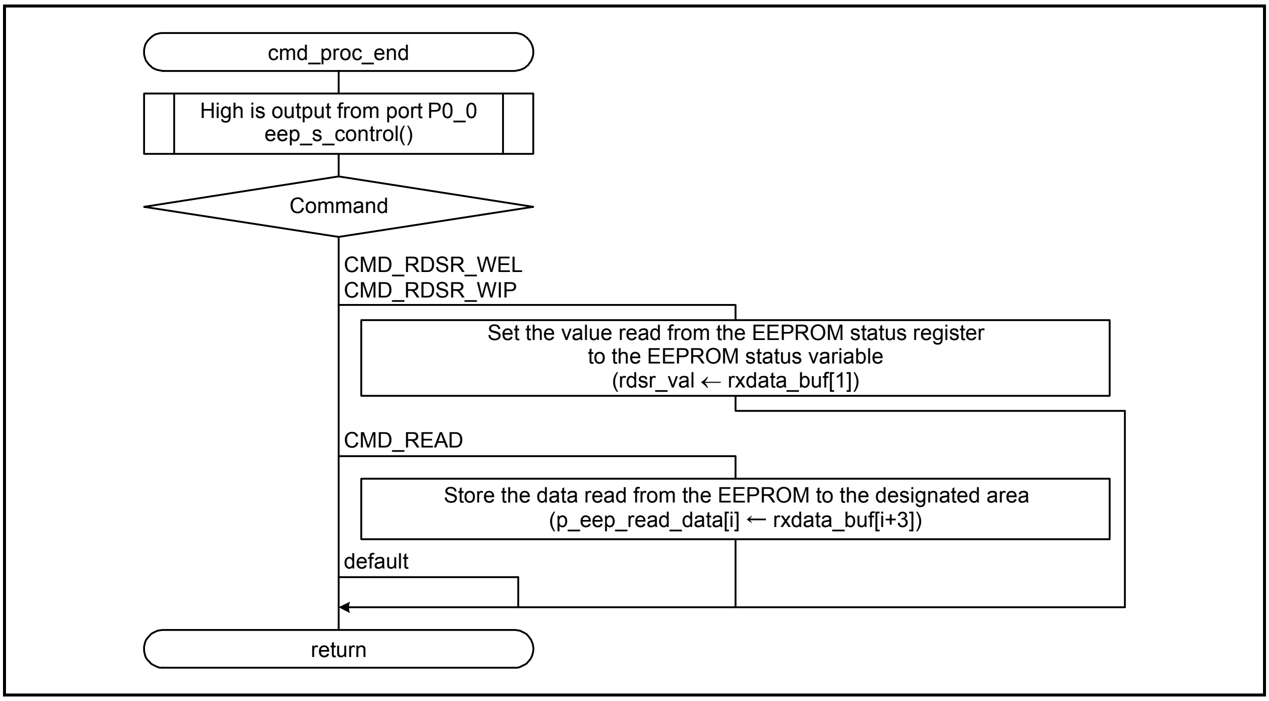


Figure 5.10 Command Execution Complete Processing

5.7.8 Processing to Proceed to the Next Command

Figure 5.11 and Figure 5.12 show the processing to proceed to the next command.

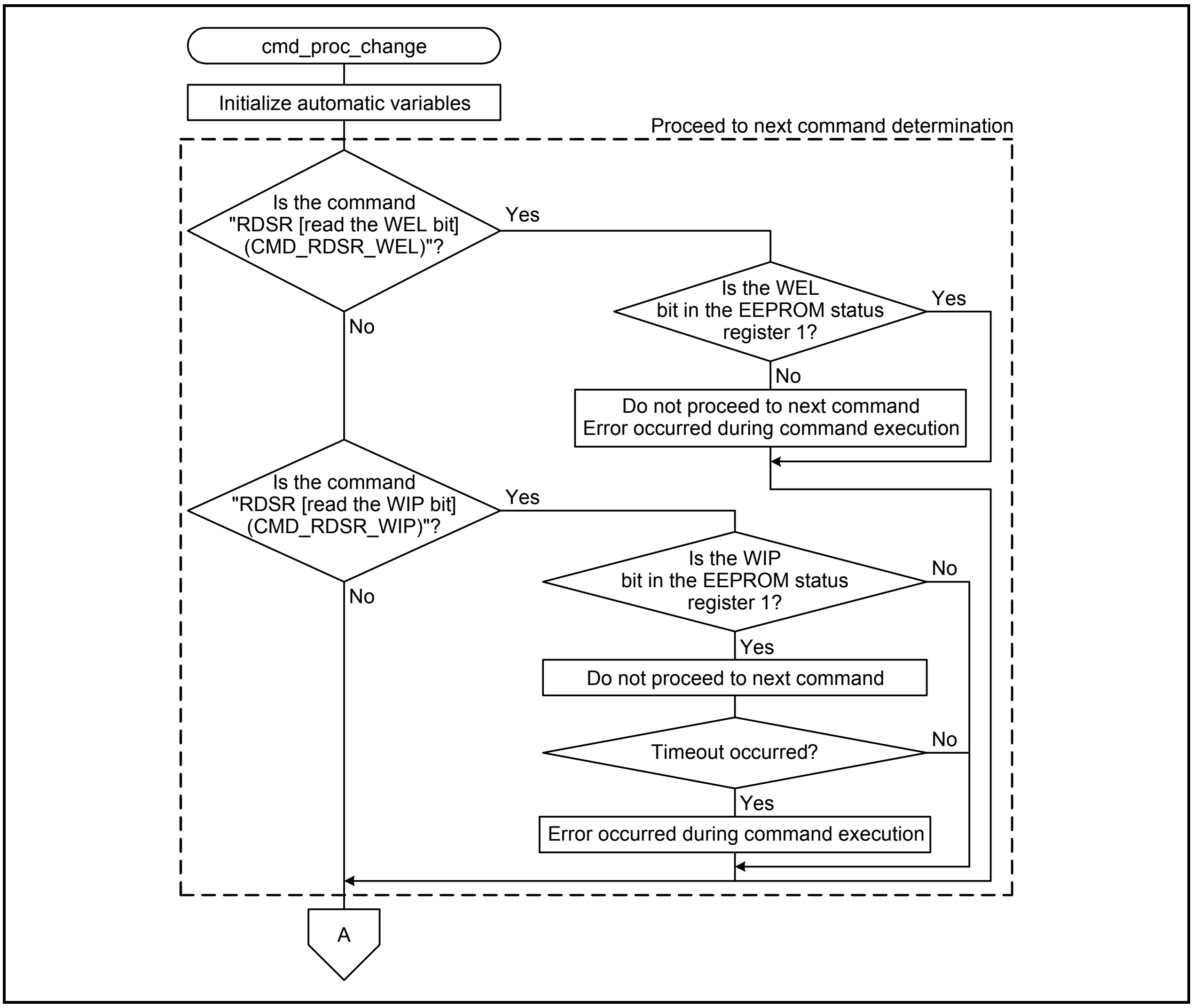
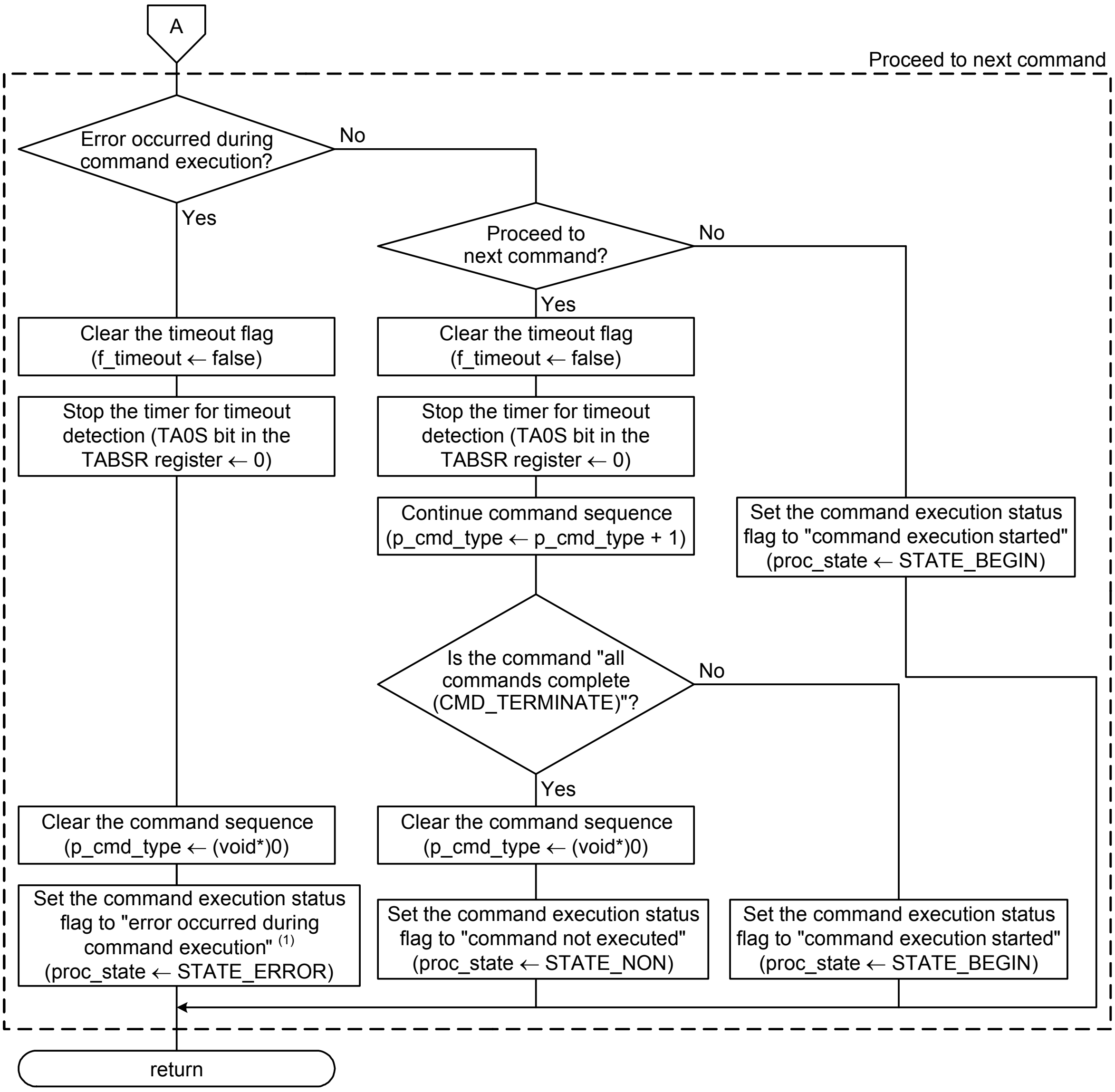


Figure 5.11 Processing to Proceed to the Next Command (1/2)



Note:
1. The sample code does not include processing of the command execution status flag (proc_state) transiting from "error occurred during command execution (STATE_ERROR)" to the original processing.

Figure 5.12 Processing to Proceed to the Next Command (2/2)

5.7.9 Controlling the EEPROM $\overline{S}$ Pin

Figure 5.13 shows controlling the EEPROM $\overline{S}$ pin.

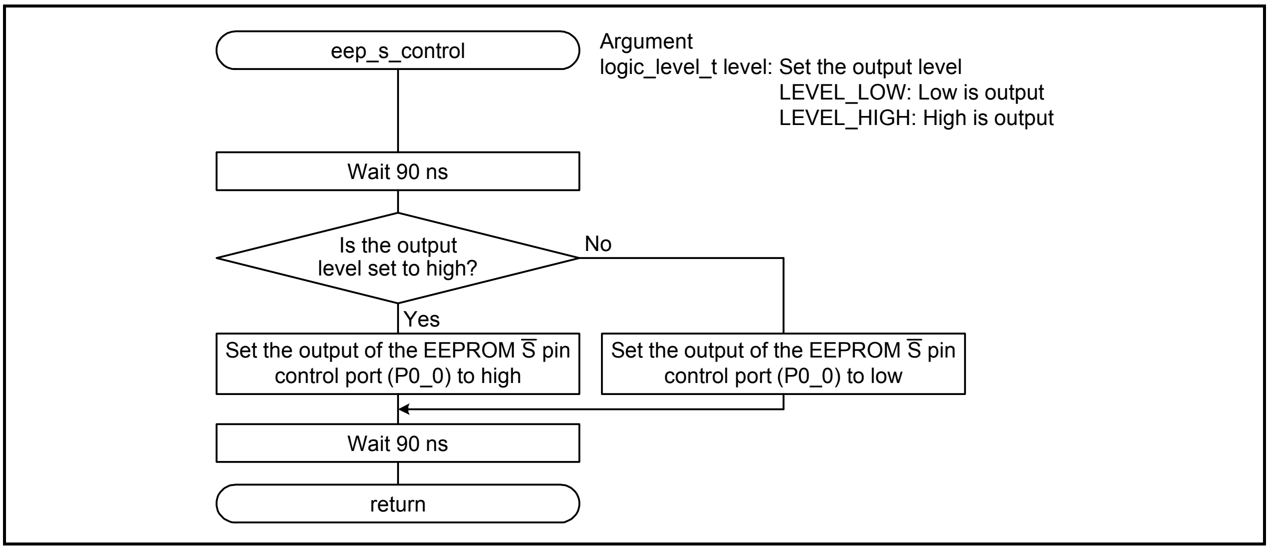


Figure 5.13 Controlling the EEPROM $\overline{S}$ Pin

5.7.10 UART2 Serial Transmission (UART2 Transmit Interrupt)

Figure 5.14 shows UART2 serial transmission (UART2 transmit interrupt).

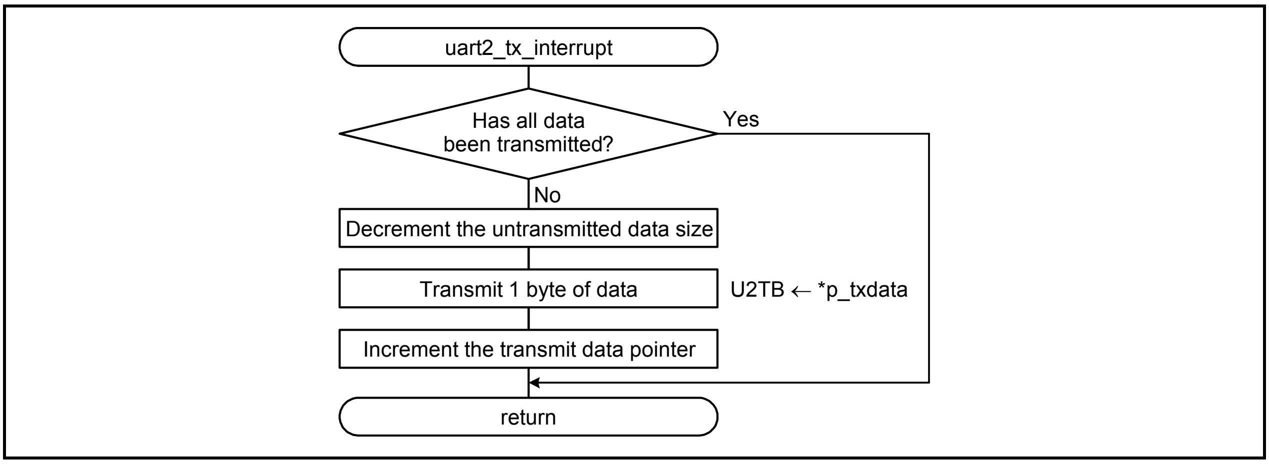


Figure 5.14 UART2 Serial Transmission (UART2 Transmit Interrupt)

5.7.11 UART2 Serial Reception (UART2 Receive Interrupt)

Figure 5.15 shows UART2 serial reception (UART2 receive interrupt).

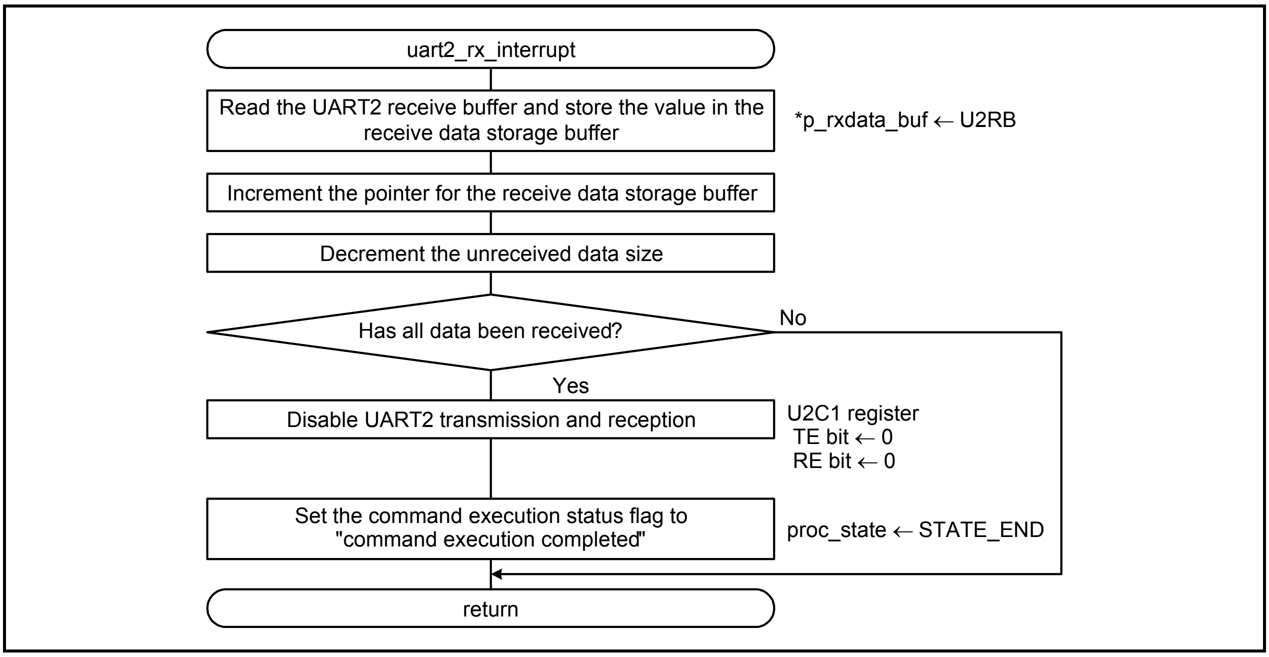


Figure 5.15 UART2 Serial Reception (UART2 Receive Interrupt)

5.7.12 5 ms Timeout Processing (Timer A0 Interrupt)

Figure 5.16 shows 5 ms timeout processing (timer A0 interrupt).

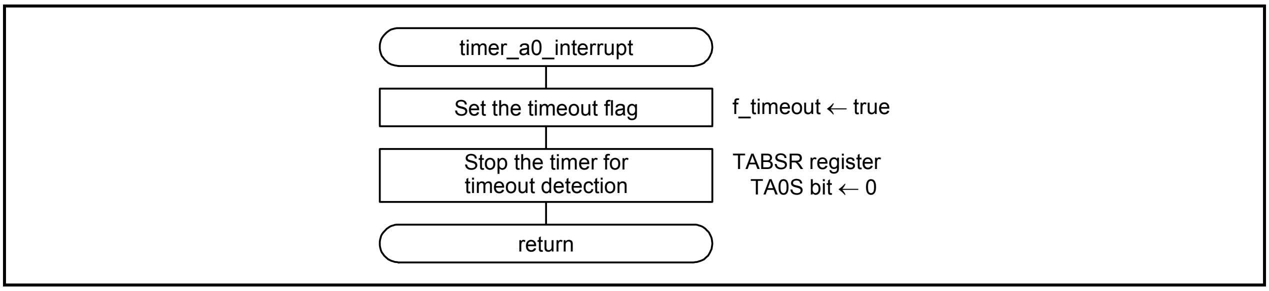


Figure 5.16 5 ms Timeout Processing (Timer A0 Interrupt)

5.7.13 Accepting an EEPROM Write Request (INT0 Interrupt)

Figure 5.17 shows accepting an EEPROM write request (INT0 interrupt).

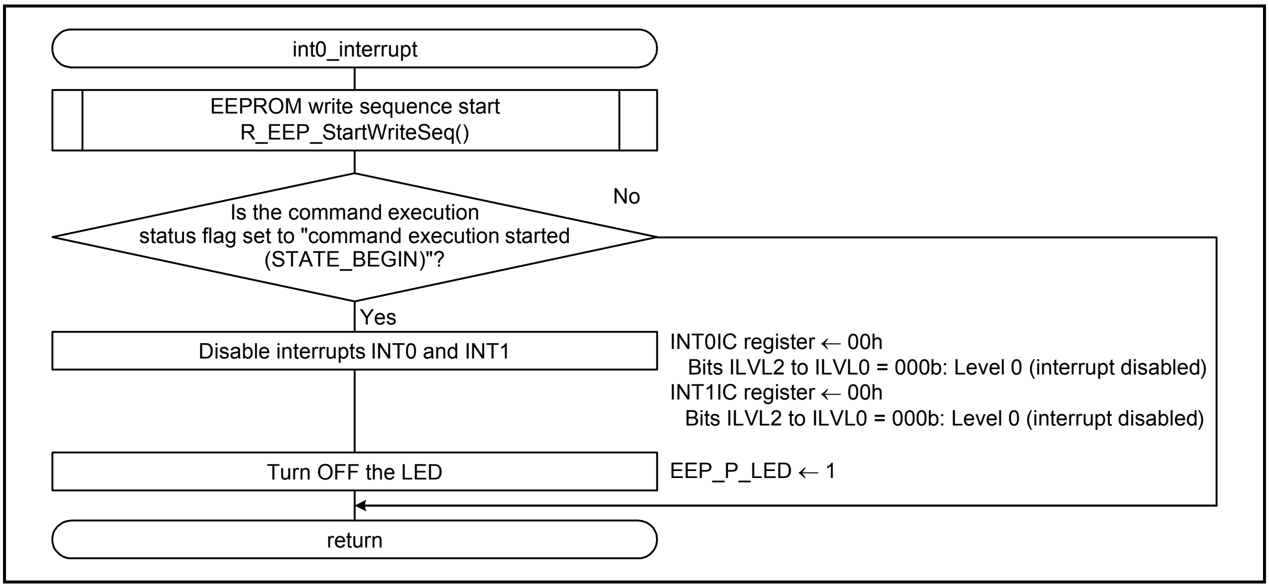


Figure 5.17 Accepting an EEPROM Write Request (INT0 Interrupt)

5.7.14 Accepting an EEPROM Read Request (INT1 Interrupt)

Figure 5.18 shows accepting an EEPROM read request (INT1 interrupt).

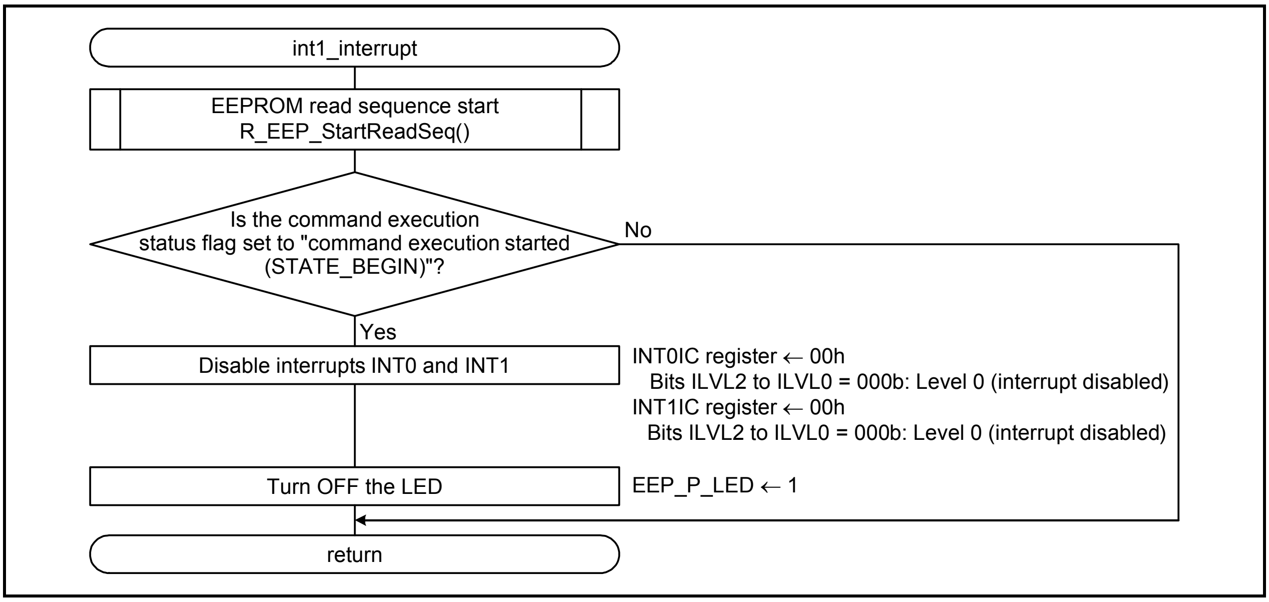


Figure 5.18 Accepting an EEPROM Read Request (INT1 Interrupt)

5.7.15 Timer A0 Initialization

Figure 5.19 shows timer A0 initialization.

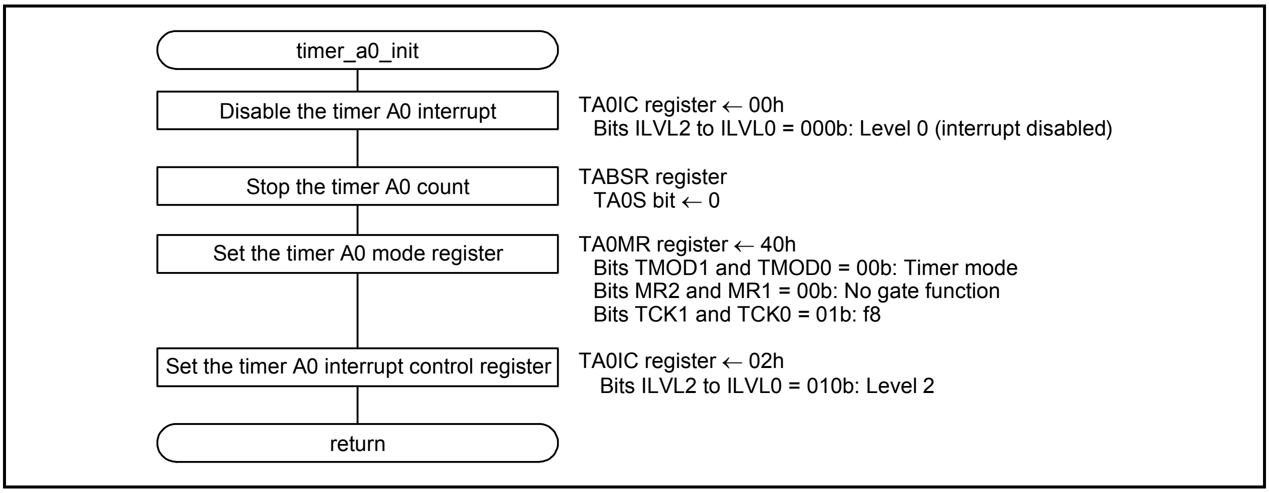


Figure 5.19 Timer A0 Initialization

5.7.16 UART2 Initialization

Figure 5.20 shows UART2 initialization.

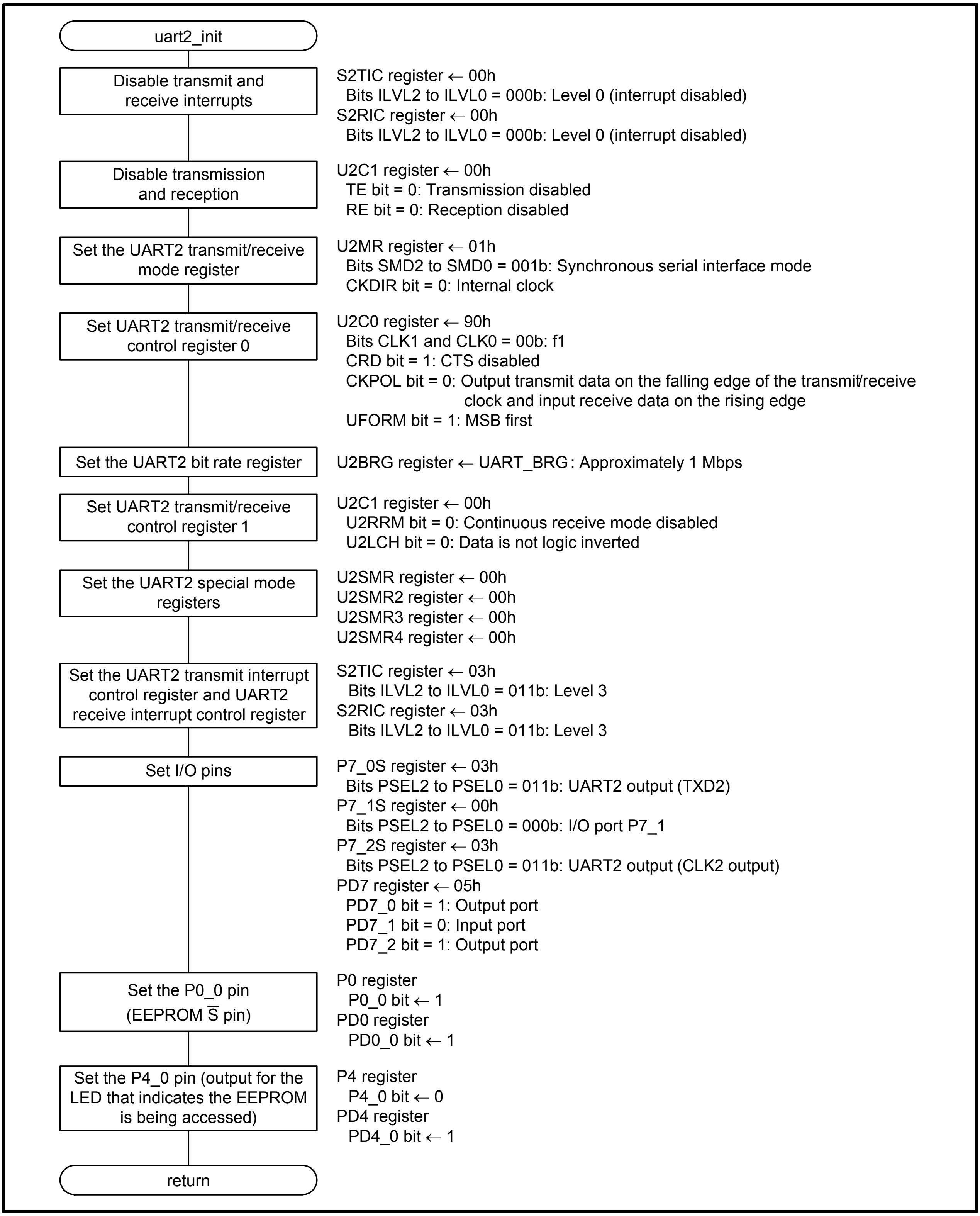


Figure 5.20 UART2 Initialization

6. Sample Code

Sample code can be downloaded from the Renesas Electronics website.

7. Reference Documents

R32C/116 Group User's Manual: Hardware Rev.1.10
R32C/117 Group User's Manual: Hardware Rev.1.10
R32C/118 Group User's Manual: Hardware Rev.1.10
The latest versions can be downloaded from the Renesas Electronics website.

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

C Compiler Manual
R32C/100 Series C Compiler Package V.1.02
C Compiler User's Manual Rev.2.00
The latest version can be downloaded from the Renesas Electronics website.

R1EX25xxx Series EEPROM Datasheet Rev. 0.01
The latest version can be downloaded from the Renesas Electronics website.

Website and Support

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

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

Revision History	R32C/100 Series Accessing an EEPROM Using Synchronous Serial Interface Mode

Rev.	Date	Description	
		Page	Summary
1.00	Dec. 14, 2012	—	First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this manual, refer to the relevant sections of the manual. If the descriptions under General Precautions in the Handling of MPU/MCU Products and in the body of the manual differ from each other, the description in the body of the manual takes precedence.

1. Handling of Unused Pins
   Handle unused pins in accord with the directions given under Handling of Unused Pins in the manual.
   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.
2. Processing at Power-on
   The state of the product is undefined at the moment when power is supplied.
   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.
3. Prohibition of Access to Reserved Addresses
   Access to reserved addresses is prohibited.
   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.
4. Clock Signals
   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.
   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.
5. Differences between Products
   Before changing from one product to another, i.e. to one with a different part number, confirm that the change will not lead to problems.
   — The characteristics of MPU/MCU in the same group but having different part numbers may differ because of the differences in internal memory capacity and layout pattern. When changing to products of different part numbers, implement a system-evaluation test for each of the products.

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-651-700, Fax: +44-1628-651-804

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 2.2