SQ003116E007

ML610Q400 Series

Sample Program AP Notes

For
EEPROM Application

2nd Edition
Issue Date: April 16, 2010

NOTICE

No copying or reproduction of this document, in part or in whole, is permitted without the consent of LAPIS Semiconductor Co., Ltd.

The content specified herein is subject to change for improvement without notice.

The content specified herein is for the purpose of introducing LAPIS Semiconductor's products (hereinafter "Products"). If you wish to use any such Product, please be sure to refer to the specifications, which can be obtained from LAPIS Semiconductor upon request.

Examples of application circuits, circuit constants and any other information contained herein illustrate the standard usage and operations of the Products. The peripheral conditions must be taken into account when designing circuits for mass production.

Great care was taken in ensuring the accuracy of the information specified in this document. However, should you incur any damage arising from any inaccuracy or misprint of such information, LAPIS Semiconductor shall bear no responsibility for such damage.

The technical information specified herein is intended only to show the typical functions of and examples of application circuits for the Products. LAPIS Semiconductor does not grant you, explicitly or implicitly, any license to use or exercise intellectual property or other rights held by LAPIS Semiconductor and other parties. LAPIS Semiconductor shall bear no responsibility whatsoever for any dispute arising from the use of such technical information.

The Products specified in this document are intended to be used with general-use electronic equipment or devices (such as audio visual equipment, office-automation equipment, communication devices, electronic appliances and amusement devices).

The Products specified in this document are not designed to be radiation tolerant.

While LAPIS Semiconductor always makes efforts to enhance the quality and reliability of its Products, a Product may fail or malfunction for a variety of reasons.

Please be sure to implement in your equipment using the Products safety measures to guard against the possibility of physical injury, fire or any other damage caused in the event of the failure of any Product, such as derating, redundancy, fire control and fail-safe designs. LAPIS Semiconductor shall bear no responsibility whatsoever for your use of any Product outside of the prescribed scope or not in accordance with the instruction manual.

The Products are not designed or manufactured to be used with any equipment, device or system which requires an extremely high level of reliability the failure or malfunction of which may result in a direct threat to human life or create a risk of human injury (such as a medical instrument, transportation equipment, aerospace machinery, nuclear-reactor controller, fuel-controller or other safety device). LAPIS Semiconductor shall bear no responsibility in any way for use of any of the Products for the above special purposes. If a Product is intended to be used for any such special purpose, please contact a ROHM sales representative before purchasing.

If you intend to export or ship overseas any Product or technology specified herein that may be controlled under the Foreign Exchange and the Foreign Trade Law, you will be required to obtain a license or permit under the Law.

Table of Contents

1. Overview

This document describes the application programming notes (hereafter called the AP notes) arranged to help customers develop software that performs EEPROM control on the ML610Q400 Series MCU (hereafter called the MCU).

APIs are provided for each function module. The AP notes describe the functions and operating conditions of each API and samples of use of those APIs.

In connection with the AP notes, a sample program is provided that actually operates using APIs on ML610Q400 Series Demo Kit.

◆ Related Documents
The following are the related documents. Read them as required.

・ML610Q400 Series Sample Program AP Notes For Sensor/Mesurement Application
・ML610Q400 Series Sample Program API Manual
・ML610Q431/ML610Q432 User's Manual
・ML610Q411/ML610Q412/ML610Q415 User's Manual
・ML610Q421/ML610Q422 User's Manual
・ML610Q482 User's Manual
・ML610Q435/ML610Q436 User's Manual
・ML610Q400 Series Demo kit Hardware User's Manual
・nX-U8/100 Core Instruction Manual
・MACU8 Assembler Package User's Manual
・CCU8 User's Manual
・CCU8 Programming Guide
・CCU8 Language Reference
・DTU8 User's Manual
・IDEU8 User's Manual
・uEASE User's Manual
・uEASE Connection Manual ML610Qxxx
・FWuEASE Flash Writer Host Program User's Manual
・LCD Image Tool User's Manual

2. System Configuration

2.1. Hardware Configuration

The following figure shows the hardware configuration on which the sample software runs.

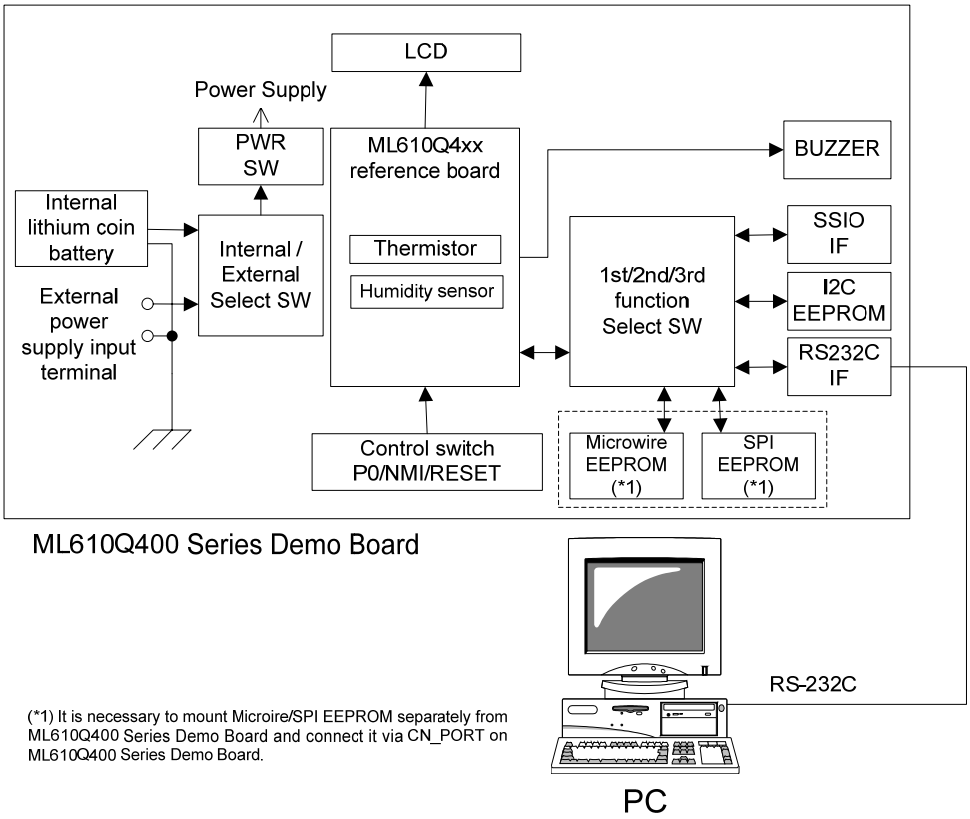


Figure 2-1 Hardware Configuration

In the above hardware configuration, the peripheral parts which are necessary for running the sample software are shown below.

Peripheral parts	The number of peripheral parts	Descriptions
Control switch	4	The switch S1, S2, S3 and S4 are used to change mode or control the application.
EEPROM	1	It saves the data, received from PC via UART. Select one EEPROM interface out of I2C/SPI/Microwire by changing option setting when compiling the sample software.
LCD panel	1	It displays the mode and result of operation.
RS-232C interface	1	It is used for data communication with PC. The communication condition is as follows. Baud rate : 9600bps, Data : 8bit, Parity bit : none, Stop bit : 1 bit When receiving data from PC, MCU writes the data into EEPROM. Then, MCU reads the saved data in EEPROM and send the data to PC.

2.2. Peripheral Circuit Diagram

The sample program sets the port P44 - P46 as 3rd function, because the sample program uses both UART and SSIO (for SPI/Microwire EEPROM control). Please use the reference board which is wired P44 - P46 as following diagram (Note that it is different from ML610Q431 reference board which is mounted on ML610Q400 Series Demo Kit in default). For details of the port 4, please refer to the chapter "Port 4" of the User's Manual for your target MCU.

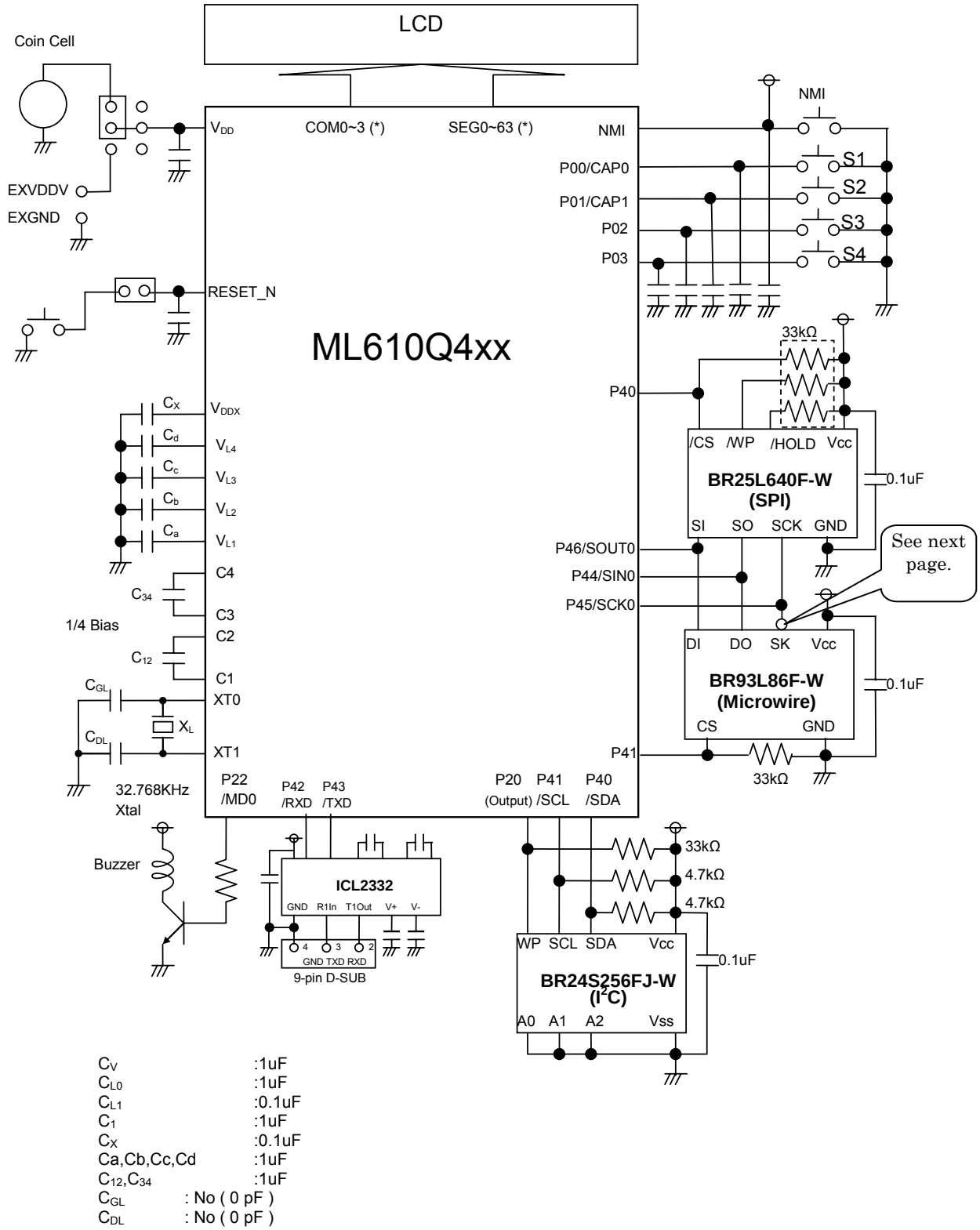
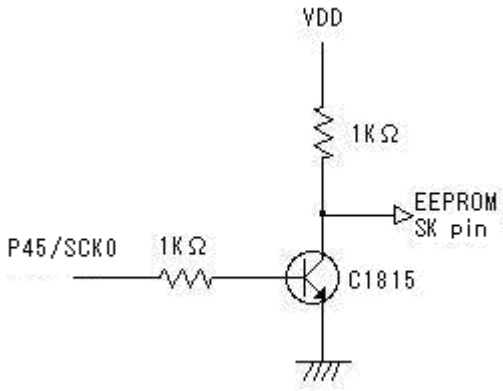


Figure 2-2 Peripheral Circuit Diagram

(*) The number of COM/SEG pin that can be connected to LCD panel depends on the type of the LCD driver built into the MCU. Please see the chapter "LCD Driver" of the User's Manual for your target MCU.

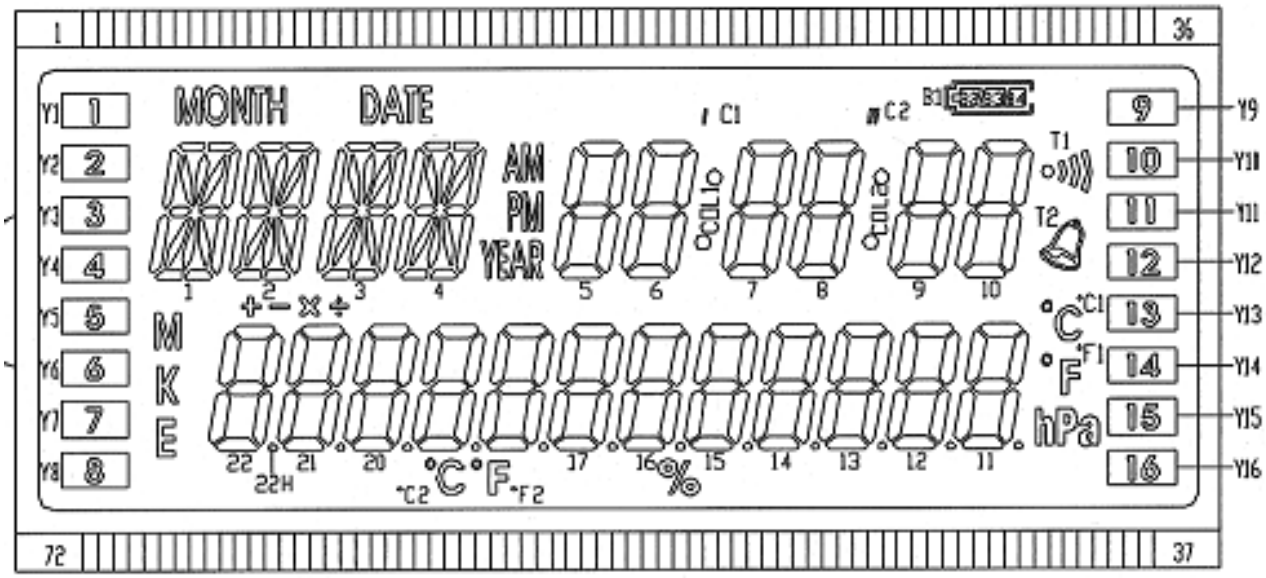
For more detail about the peripheral curcuit, please see the “ML610Q400 Series Demo Kit Hardware User’s Manual”.

Note:
When transmitting or receiving data by SSIO, MCU outputs or captures data on the rising clock edge. But Microwire EEPROM (BR93L86F-W) captures or outputs data on the falling clock edge. Therefore the following inversion curcuit is necessary at SK pin of Microwire EEPROM.



SCK inversion circuit

2.3. LCD Panel Specifications



16-segment characters:	The 4 digits on the upper part of the panel
7-segment characters:	The 6 digits on the upper part of the panel
8-segment characters:	The 12 digits on the lower part of the panel
Marks for hand-held calculator:	7
Other marks:	32

Figure 2-3 Layout of the LCD Panel

Table 2-1 Pin Assignments (COM/SEG)

PIN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
COM1	/	/	/	COM1	Y1	1H	1A	1B	1C	MONTH	2H	2A	2B	2C	3H	3A	3B	3C	4H	4A	4B	4C	AM	5F
COM2	/	/	COM2	/	Y2	1I	1J	1K	1L	DATE	2I	2J	2K	2L	3I	3J	3K	3L	4I	4J	4K	4L	PM	5G
COM3	/	COM3	/	/	Y3	1P	1Q	1N	1M	/	2P	2Q	2N	2M	3P	3Q	3N	3M	4P	4Q	4N	4M	YEAR	5E
COM4	COM4	/	/	/	Y4	1G	1F	1E	1D	/	2G	2F	2E	2D	3G	3F	3E	3D	4G	4F	4E	4D	/	5D
PIN	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
COM1	5A	6F	6A	7F	7A	8F	8A	9F	9A	10F	10A	B3	/	Y9	Y13	T2	11A	11F	12A	12F	13A	13F	14A	14F
COM2	5B	6G	6B	7G	7B	8G	8B	9G	9B	10G	10B	B4	/	Y10	Y14	*C1	11B	11G	12B	12G	13B	13G	14B	14G
COM3	5C	6E	6C	7E	7C	8E	8C	9E	9C	10E	10C	B2	/	Y11	Y15	*F1	11C	11E	12C	12E	13C	13E	14C	14E
COM4	/	6D	COL1	7D	C1	8D	COL2	9D	C2	10D	T1	B1	/	Y12	Y16	hpa	11H	11D	12H	12D	13H	13D	14H	14D
PIN	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72
COM1	15A	15F	16A	16F	17A	17F	18A	18F	/	19A	19F	20A	20F	21A	21F	22A	22F	÷	M	Y5	/	/	/	COM1
COM2	15B	15G	16B	16G	17B	17G	18B	18G	%	19B	19G	20B	20G	21B	21G	22B	22G	×	K	Y6	/	/	COM2	/
COM3	15C	15E	16C	16E	17C	17E	18C	18E	*F2	19C	19E	20C	20E	21C	21E	22C	22E	−	E	Y7	/	COM3	/	/
COM4	15H	15D	16H	16D	17H	17D	18H	18D	*C2	19H	19D	20H	20D	21H	21D	22H	22D	+	/	Y8	COM4	/	/	/

Specifications of Operation

Clock for bias generation circuit voltage multiplication:	1/16 LSCLK (2 kHz)
Bias of the bias generation circuit:	1/4
Duty:	1/4 duty
Frame frequency:	73 Hz

2.4. Software Configuration

Figure 2-4 shows the software configuration.

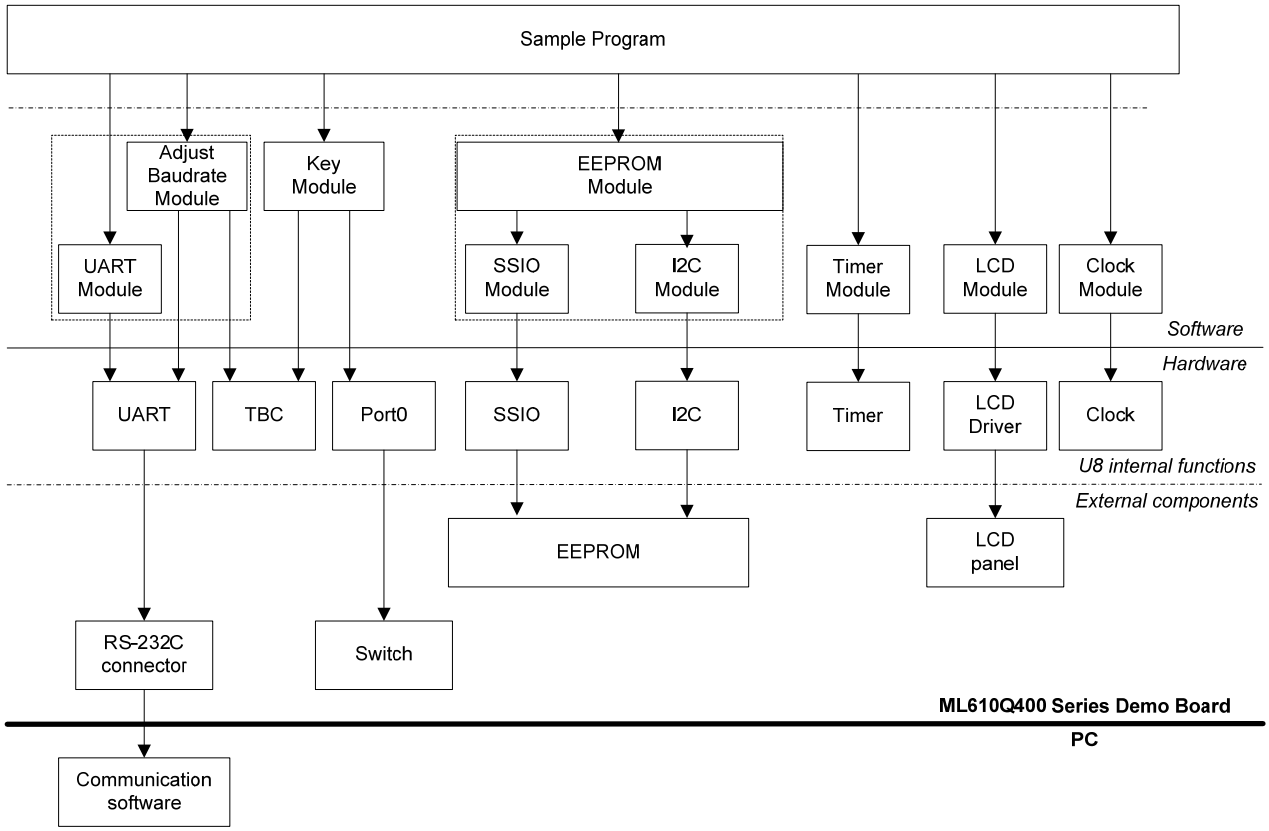


Figure 2-4 Software Configuration

2.5. List of Folders and Files

The folders and the files are as listed below.

```
[eeprom]
├ [_output]                  … Build result output folder
│ ├ _hex
│ ├ _lst
│ ├ _obj
│ └ _prn
├ [adjustBaudrate]           … UART baud rate correction module folder
│ ├ adjustBaudrate.c
│ └ adjustBaudrate.h
├ [clock]                    … Clock control module folder
│ ├ clock.c
│ ├ clock.h
│ ├ clock_sysFunc.c
│ └ clock_sysFunc.h
├ [common]                   … General-purpose function module folder
│ ├ common.c
│ └ common.h
├ [eeprom]                   … EEPROM control module folder
│ ├ eeprom.c
│ ├ eeprom.h
│ ├ eeprom_i2c.c
│ ├ eeprom_microwire.c
│ └ eeprom_spi.c
├ [i2c]                      … I2C communication control module folder
│ ├ i2c.c
│ └ i2c.h
├ [irq]                      … Interrupt control module folder
│ ├ irq.c
│ └ irq.h
├ [key]                      … Key input control module folder
│ ├ key.c
│ └ key.h
├ [lcd]                      … LCD display control module folder
│ ├ LCD.c
│ ├ LCD.h
│ ├ U8_Sample.tac
│ └ U8_Sample.tbc
├ [main]                     … Sample program main folder
│ ├ [mcu_large]
│ │ └ mcu.h
│ ├ [mcu_small]
│ │ └ mcu.h
│ ├ eepromMap.h
│ ├ main.c
│ ├ main.h
│ ├ S610431SW.asm
│ ├ S610435LW.asm
│ ├ uartMode.c
│ └ uartMode.h
├ [ssio]                     … SSIO module folder
│ ├ ssio.c
│ └ ssio.h
├ [tbc]                      … Time base counter control module folder
│ ├ tbc.c
│ └ tbc.h
```

[continued from the previous page]

```
├ [timer]                    … Timer control module folder
│ ├ timer.c
│ └ timer.h
├ [uart]                     … UART communication control module folder
│ ├ uart.c
│ └ uart.h
├ readme.txt                 … Description of compile options
├ U8_EEPROM_Sample_Large.PID       … Project file for large model MCU
└ U8_EEPROM_Sample_Small.PID       … Project file for small model MCU
```

2.6. Build Procedure

① Start IDEU8, select the menu “Open” and open the project file (PID file). In the case that MCU memory model is small model, the project file is “U8_EEPROM_Sample_Small.PID”. In the case of large model, the project file is “U8_EEPROM_Sample_Large.PID”. Correspondence of MCU and PID file is shown below.

Table 2-2 Correspondence of MCU and PID file

	U8_EEPROM_Sample_Small.PID	U8_EEPROM_Sample_Large.PID
Supported MCU	ML610Q431/432 ML610Q421/422 ML610Q411/412/415 ML610Q482	ML610Q435/436

② In the default setting, ML610Q431 is set as the target MCU.
If your target MCU is different, follow the procedure below to change the setting.

(1) Select the menu “Project” -> “Options” -> “Compiler/assembler”.
(2) In the displayed window, select the target MCU from the “Target microcontroller” list in the “General” tab.
(3) Remove the startup file “S610431SW.asm“ registered in the file tree of IDEU8. Instead of that, register your target MCU’s startup file. (In the case of ML610Q432, it is S610432SW.asm.)
(4) Define the macro that represents the target MCU.
Select the menu “Project” -> “Options” -> “Compiler/assembler” -> ”Macro”tab. In the displayed window, modify the macro like following name.
_ML610Q4XX
About the “XX” part, replace with the type number of MCU
For example, if ML610Q432 is used, define the following macro.
_ML610Q432
In the case that the macro other than the type number in the above Table 2-2 is defined, the case that macro such as above is not defined, or the case that the memory model that is supported by PID file is different from the memory model of MCU that is defined by the above macro, the compiler issues the following error at the beginning of the output messages.
Error : E2000 : #error : “Unknown target MCU”
(5) If necessary, modify other macro definitions.
About the available macro definitions, see the “readme.txt” in the sample program folder.
- For ML610Q43X series MCU
  LCD_TYPE = 1
  FREQ_TIMER_MODE = 0
  _EEPROM_IF_TYPE = 1 or 2 or 3
  _SSIO_P46_P45_P44
- For ML610Q42X series MCU
  LCD_TYPE = 1
  FREQ_TIMER_MODE = 0 or 1
  _EEPROM_IF_TYPE = 1 or 2 or 3
  _SSIO_P46_P45_P44
- For ML610Q41X series MCU
  LCD_TYPE = 0
  FREQ_TIMER_MODE = 0 or 1
  (For ML610Q415, frequency measurement mode by hardware is not available on ML610Q415 because it does not have low-speed crystal oscillation clock. Please define FREQ_TIMER_MODE macro as 0.)
  _EEPROM_IF_TYPE = 1 or 2 or 3
  _SSIO_P46_P45_P44
- For ML610Q41X series MCU
  FREQ_TIMER_MODE = 0 or 1
  _EEPROM_IF_TYPE = 1 or 2 or 3
  _SSIO_P46_P45_P44

③ Select the menu “Project” -> “Rebuild”. Then the build procssing for the sample program starts.

④ When the build processing is completed, .abs file is generated in the project folder and .hex file is generated in _output¥_hex folder.

2.7. Restrictions

2.7.1. About Available Functional Modules

In the functional modules that compose this sample program, the available functional modules are different by target MCU, due to the difference of MCU peripherals. In the case that these functional modules are applied to user application, available functional modules on each MCU are shown below.

Table 2-3 List of available functional modules

		Supported MCU			
		ML610Q43X	ML610Q42X	ML610Q41X	ML610Q48X
Functional modules	UART Communication Control Module *3	○	○	○	○
	UART Baud Rate Correction Module	○	○	○	○
	Frequency measurement mode *3	×	○	○ *1	○
	Key Input Control Module *3	○	○	○	○
	EEPROM Control Module	○	○	○	○
	SSIO Module *3	○	○	○	○
	I2C Communication Control Module *3	○	○	○	○
	Timer Control Modul *3	○	○	○	○
	LCD Display Control Module *3	○	○ *2	○ *2	×
	Clock Control Module *3	○	○	○	○
	Time Base Counter Control Module *3	○	○	○	○

○ : Available
× : Not available

*1: Frequency measurement mode by hardware is not available on ML610Q415 because it does not have low-speed crystal oscillation clock.
*2: All display area of LCD panel can not be available, because the number of SEG pin that is connected to LCD panel is not enough.
*3: For the details of these modules, please see the “ML610Q400 Series Sample Program AP Notes For Sensor/Mesurement Application”.

2.7.2. About Display Area of LCD panel

The display area of LCD panel is different by each MCU as follows, because of the specification difference of LCD driver.
* It is requred for displaying all areas of LCD panel that LCD driver supports 64seg×4com pins at least. The number of COM/SEG pin that LCD driver in each MCU supports is listed in parenthesis.

ML610Q43X: All area can be displayed.
(ML610Q431: 64seg×16com, ML610Q432: 64seg×24com)

ML610Q42X: Only the area of ①, ② and ④ can be displayed.
(ML610Q421: 50seg×8com, ML610Q422: 50seg×16com)

ML610Q41X: Only the area of ① and ② can be displayed.
(ML610Q411: 36seg×4com, ML610Q412: 44seg×4com, ML610Q415: 36seg×4com)

ML610Q48X: All area can not be displayed, because ML610Q48X does not have LCD driver.

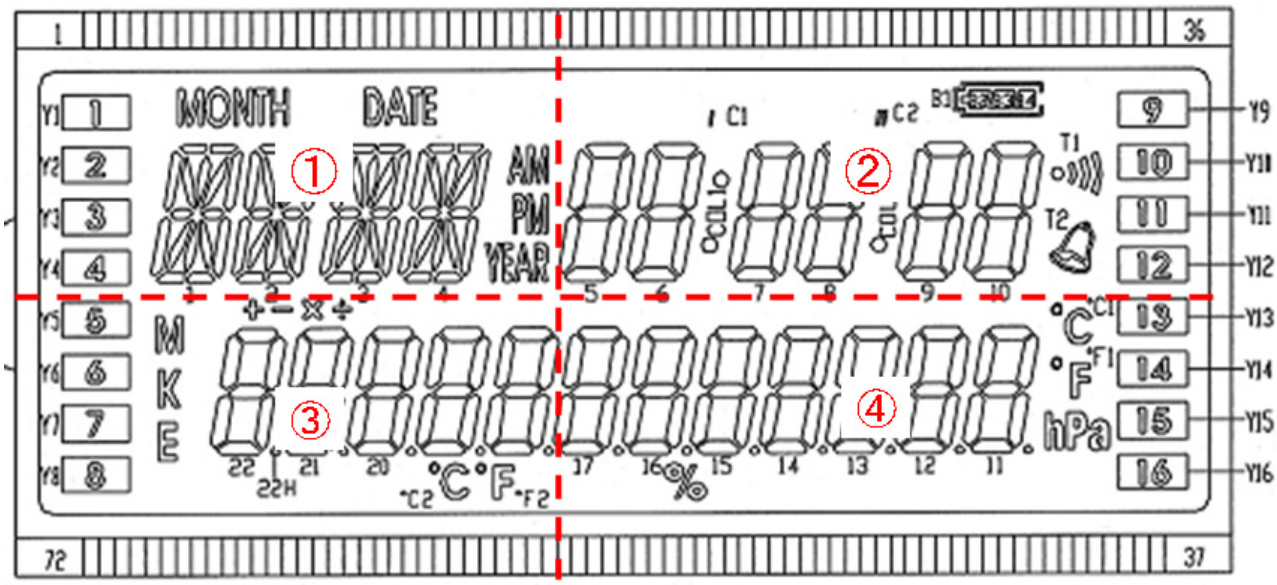


Figure 2-5 Layout of the LCD Panel

2.8. About SPI bus and Microwire bus interface

The MCU has one channel of the 8/16-bit synchronous serial port (SSIO) and can also be used to control the EEPROM device. This AP note describes how to control EEPROM which has the following bus interface.

* For details about SSIO, refer to the chapter "Synchronous Serial Port" of the User's Manual for your target MCU.

Table 2-4 Bus interface

Bus interface	Description
SPI	SPI is proposed by Motorola (Freescale). It is a three-wire serial interface. The three signals are input, output and clock. Also, it requires a separated chip select input for each connected slave device. In the sample program, the data size in each transmission is 8bit.
Microwire	Microwire is proposed by National Semiconductor. Similar to SPI, it controls three serial signals (output, input and clock), and chip select a separated chip select input for each connected slave device. In the sample program, the data size in each transmission is 16bit.

* About the description for I2C bus interface EEPROM, please refer "ML610Q400 Series Sample Program AP Notes for Sensor/Mesurement Application".

The sample program provides three ways to control EEPROM, which are by using I2C, SPI and Microwire bus interface. In default setting, I2C bus interface is selected. If the following macro definition is changed, SPI or Microwire interfade also can be selected.

Macro	Value	Enabled Bus interface
_EEPROM_IF_TYPE	1	I2C
	2	SPI
	3	Microwire

* If the value other than 1, 2 and 3 is defined as the above macro, the compiler issues the following error message.

```
Error : E2000 : #error :    "The compilation switch "_EEPROM_IF_TYPE" is illegal setting in EEPROM module."
```

2.8.1. SPI bus interface

This section describes about SPI bus interface EEPROM which the sample program uses.

2.8.1.1. EEPROM specification

The following shows the specification of EEPROM hich the sample program uses.

- Manufacturer — Rohm
- Part Number — BR25L640F-W
- Capacity — 8Kbyte
- Operating voltage — 1.8V - 5.5V
- Interface — SPI
- Maximum clock frequency — 2MHz (1.8V - 2.5V)
  5MHz (2.5V - 5V)

  * In the sample program, 125KHz (a quarter of 500KHz HSCLK) is used as serial clock.
- Data bit order — MSB first
- Data bit length — 8 bit
- Clock output phase — Low level in default
- CS signal — Low level in enable, High level in disable.

2.8.1.2. Command

Communication with SPI interface EEPROM is done per command.
The following table shows the commands for this EEPROM.

Command	Opcode (Binary format)	Comments
WRITE	0000 0110	Write data
READ	0000 0100	Read data
WREN	0000 0011	Enable write operations
WRDI	0000 0010	Disable write operations
RDSR	0000 0101	Read status register
WRSR	0000 0001	Write status register

2.8.2. Microwire bus interface

This section describes about Microwire bus interface EEPROM which the sample program uses.

2.8.2.1. EEPROM specification

The following shows the specification of EEPROM hich the sample program uses.

- Manufacturer — Rohm
- Part Number — BR93L86F-W
- Capacity — 2Kbyte
- Operating voltage — 1.8V - 5.5V
- Bus interface — Microwire
- Maximum clock frequency — 500KHz (lower than 2.5V)
  2MHz (2.5V - 5V)

  * In the sample program, 125KHz (a quarter of 500KHz HSCLK) is used as serial clock.
- Data bit order — MSB first
- Clock output phase — High level in default
- Data bit length — 16 bit
- CS signal — High level in enable, Low level in disable.

2.8.2.2. Command

Communication with Microwire bus interface EEPROM is done per command.
The folloing table shows the commands for this EEPROM.

Command	Start bit	Opcode (Binary format)	Address (Binary format)	Comments
WRITE	1	01	A9,A8,A7,A6,A5,A4,A3,A2,A1,A0	Write data
READ	1	10	A9,A8,A7,A6,A5,A4,A3,A2,A1,A0	Read data
WEN	1	00	1 1 * * * * * * * *	Enable write operations
WED	1	00	0 0 * * * * * * * *	Disable write operations

"*" means "don't care", so the sample program assigns 0 for this bit.

SSIO in the MCU can transmit data only per 8bit or 16bit. But about Microwire bus interface, the total bit length of opcode and address is not a multiple of 16bit. (In the case of above command, the total bit length from start bit to the end of address is 13bit.)
In the EEPROM used in the sample program, the start condition of command is defined as "the first input of 1 after CS signal is rised". So it is allowed to control the bit length by inserting 0 data before the start bit. The sample program inserts 3bit of 0 before the start bit so that the total length of the command becomes 16bit.

2.9. Changing EEPROM parameter

In the case that EEPROM and its capacity is changed, please change the following macro definition in the sample program.

- Change capacity

File	Macro	Value	Comments
eeprom_spi.c	EEPROM_SIZE	8192	Capacity of EEPROM [byte].
eeprom_microwire.c		2048	Capacity of EEPROM [byte].

- Change page size

File	Macro	Value	Comments
eeprom_spi.c	EEPROM_PAGE_SIZE	32	Page size [byte], used only for SPI.

3. Description of Functional Modules

3.1. EEPROM Control Module (SPI)

The sample program provides the following functional module to control SPI bus interface EEPROM.

3.1.1. Function Overview

This EEPROM control module writes/reads data to/from SPI bus interface EEPROM using the SSIO of the MCU. Using the module, N-byte write and N-byte read can be controlled.

Table 3-1 lists the EEPROM control module APIs for SPI bus interface. It is compatible with APIs for Microwire bus interface.

Table 3-1 List of APIs

Function name	Description
eeprom_init function	Initializes EEPROM control module.
eeprom_read function	Starts reading data from EEPROM.
eeprom_write function	Starts writing data to EEPROM.
eeprom_continue function	Continues data write or read processing.
eeprom_stop function	Stops data write or read processing.
eeprom_getStatus function	Acquires the EEPROM control module status.

3.1.2. Operating condition

This section describes the operating conditions and valid range of this module. It also describes the restrictions on this module.

- This module performs read/write operations on EEPROM via SSIO of the MCU. For this reason, it is necessary to set the settings for HSCLK (will be used by the clock of SSIO) and SSIO interrupt processing in advance.
- The following EEPROM is supported by this module: BR25L640F-W manufactured by Rohm. One EEPROM device can be connected.
- This module does not handle the write protect pin (WP), because this module performs the write protect function by WREN/WRDI command.
- At initialization, set the Port40 settings as shown in the table below.

Table 3-2 Port 40 Settings

Port	Input/output mode	Input/output status	Description
P40	Output mode	CMOS output	EEPROM chip select (CS) pin

3.1.3. Sample of Use

The subsection describes the procedure to read/write data from/to EEPROM.

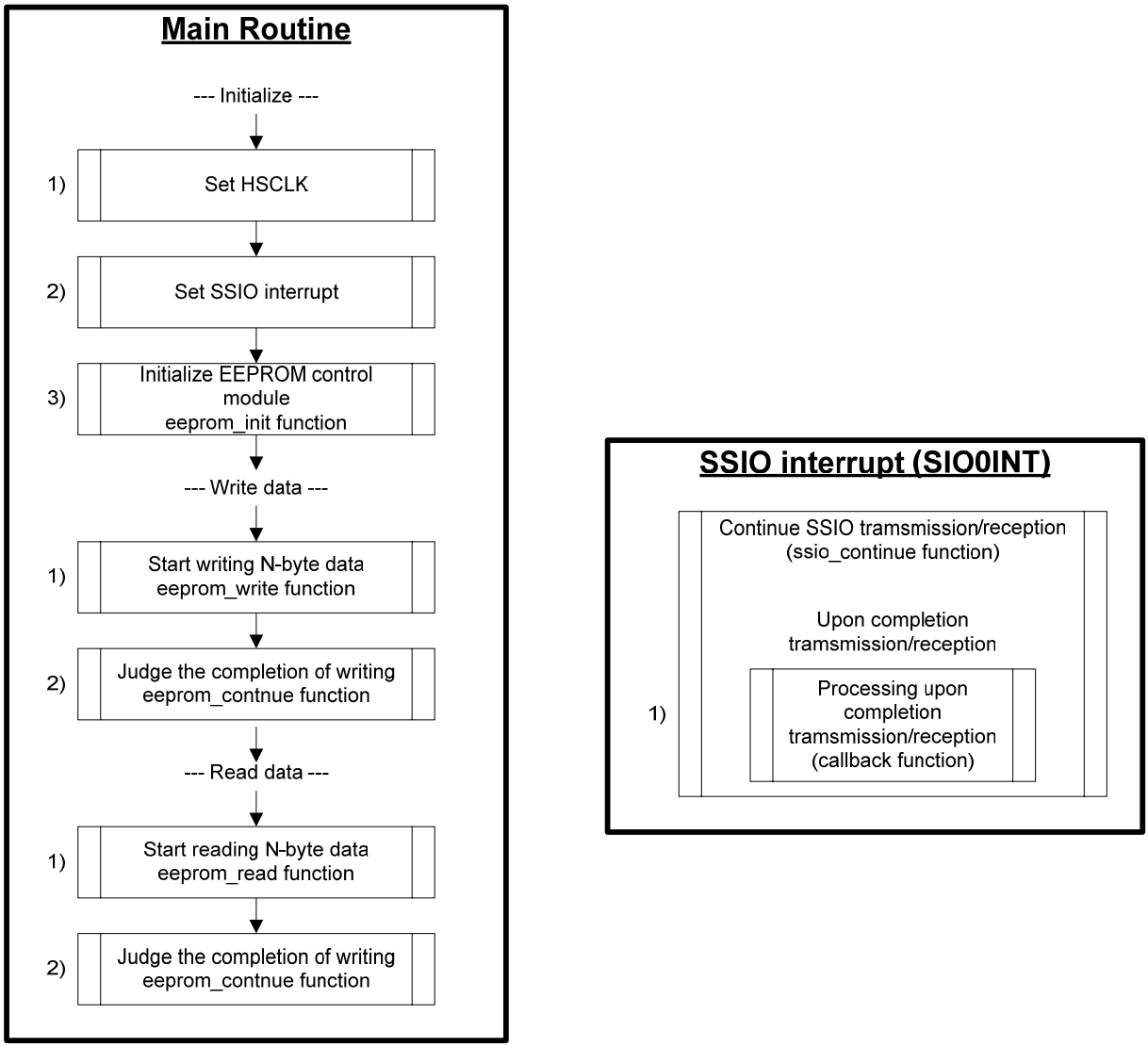


Figure 3-1 EEPROM Write/Read Procedure

3.1.3.1. Initialize

The following shows the initialization procedure for EEPROM control module..

[Main Routine]

1) Set HSCLK
   - Since SSIO uses HSCLK as a clock, it is necessary to set the settings for HSCLK before initializing the EEPROM control module. The sample program sets HSCLK as 500KHz.

2) Set SSIO interrupt
   - Set SSIO interrupt processing in the interrupt vector of SSIO. The ssio_continue function is called from there.

3) Initialize EEPROM control module

Perform the following initialization using the **eeprom_init function**:

- Set the settings for the P40 pin as the chip select (CS) pin.
- Initialize the control parameters of the EEPROM control module (put the module into an EEPROM read/write stop state).
- Set the following communication conditions and initialize the SSIO communication control module:
  ①Transfer clock : 125KHz (1/4 HSCLK)
  ②Transfer bit order : MSB first
  ③Bit length : 8 bit
  ④Clock output phase : Low level in default

3.1.3.2. Write data

The EEPROM has a page write function, so the sample program writes data to EEPROM per page.
The sample program sends WREN command before WRITE command, because the EEPROM status transits to "disable write" after each WRITE command is executed.
The following shows the EEPROM write procedure.

1) Start writing N-byte data
   - Call **eeprom_write function** with the following transmit data information. This function tramsmits WREN command and starts the command tramsmission processing for writing data to EEPROM.
     ①Address at which EEPROM write is started
     ②Initial address of the area that contains transmit data
     ③Transmit data size (in bytes)

2) Judge the completion of writing
   - Judge the completion of writing in the **eeprom_continue function**.

[Callback function upon SSIO tramsmission/reseption completion]

1) Continue the write processing
   - The callback function is called when each command or data transmission/reception is completed. This function performs the following processing. In detail, please see "3.1.3.4 Read/Write Processing Detail".
     ①If there is the remained data for writing, start tramsmission of the remained data. Otherwise, completes the write processing.
     ②While writing to EEPROM, transmit RDSR command in order to judge the completion of EEPROM writing. This function completes the write processing when no data remains or an error is occurred.

3.1.3.3. Read data

The following shows the EEPROM read procedure.

1) Start reading N-byte data
   - Call **eeprom_read function** with the following transmit data information. This function starts the command tramsmission processing for reading data from EEPROM.
     ①Address at which EEPROM read is started
     ②Initial address of the area that contains the receive data
     ③Receive data size (in bytes)
     * Actual processing for reading data from EEPROM is performed in the callback function upon completion SSIO tramsmission/reception.

2) Judge the completion of reading
   - Judge the completion of reading in the **eeprom_continue function**.

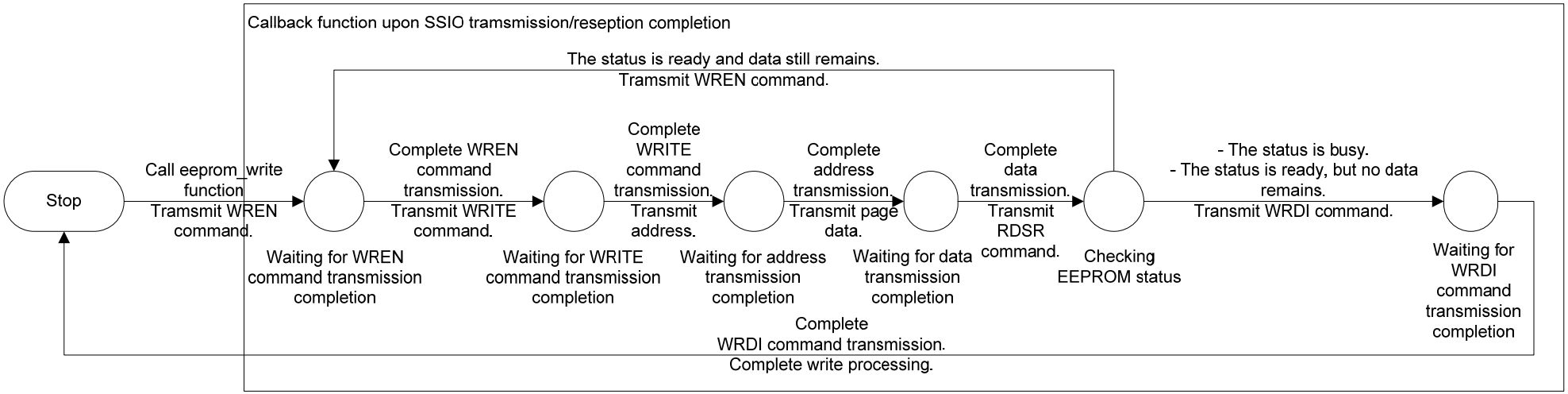
[Callback function upon SSIO tramsmission/reseption completion]

1) Continue the read processing
   - The callback function is called when each command or data transmission/reception is completed. This function performs the processing for reading data and its completion. In detail, please see "3.1.3.4 Read/Write Processing Detail".

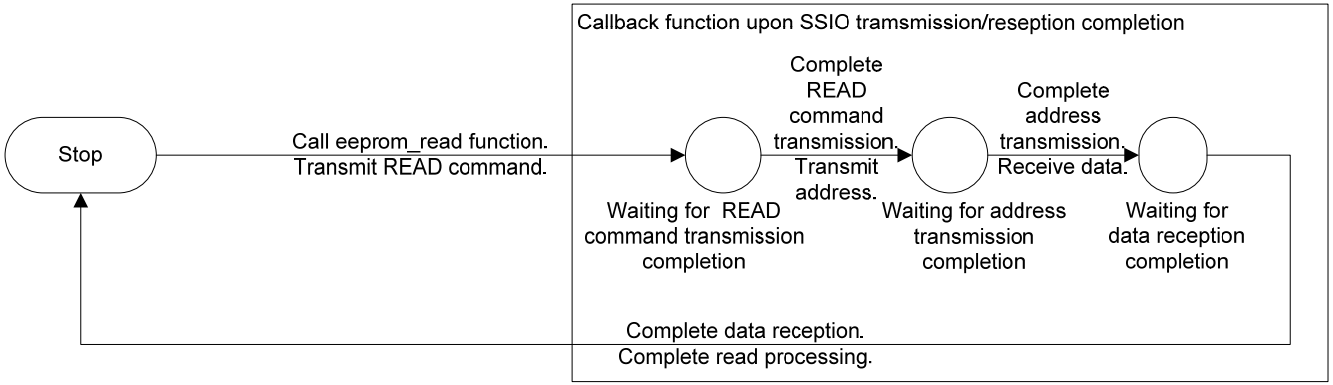
3.1.3.4. Read/Write Processing Detail

The following shows the state transition diagrams of the EEPROM control module in read and write processing.
Comments at upper part and lower part of each transition mean the condition of the transition and the action on the transition, respectively.

In EEPROM Write Processing



In EEPROM Read Processing



3.2. EEPROM Control Module (Microwire)

The sample program provides the following functional module to control Microwire bus interface EEPROM.

3.2.1. Function Overview

This EEPROM control module writes/reads data to/from Microwire bus interface EEPROM using the SSIO of the MCU. Using the module, N-byte write and N-byte read can be controlled.

Table 3-3 lists the EEPROM control module APIs for Microwire bus interface. It is compatible with APIs for SPI bus interface.

Table 3-3 List of APIs

Function name	Description
eeprom_init function	Initializes EEPROM control module.
eeprom_read function	Starts reading data from EEPROM.
eeprom_write function	Starts writing data to EEPROM.
eeprom_continue function	Continues data write or read processing.
eeprom_stop function	Stops data write or read processing.
eeprom_getStatus function	Acquires the EEPROM control module status.

3.2.2. Operating condition

This section describes the operating conditions and valid range of this module. It also describes the restrictions on this module.

- This module performs read/write operations on EEPROM via SSIO of the MCU. For this reason, it is necessary to set the settings for HSCLK (will be used by the clock of SSIO) and SSIO interrupt processing in advance.
- The following EEPROM is supported by this module: BR93L86F-W manufactured by Rohm. One EEPROM device can be connected.
- At initialization, set the Port41 settings as shown in the table below.

Table 3-4 Port 41 Settings

Port	Input/output mode	Input/output status	Description
P41	Output mode	CMOS output	EEPROM chip select (CS) pin

3.2.3. Sample of Use

The subsection describes the procedure to read/write data from/to EEPROM.

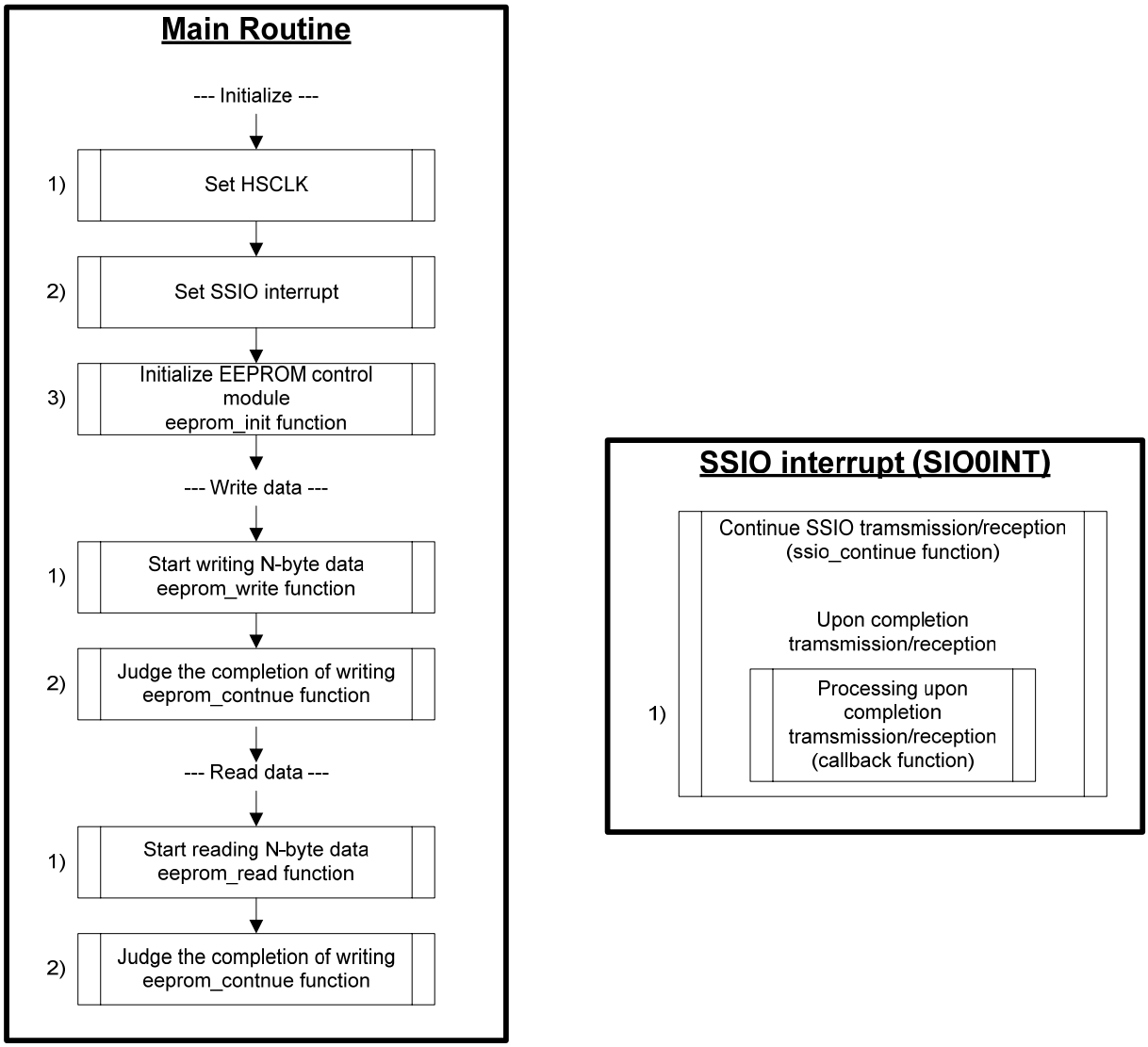


Figure 3-2 EEPROM Write/Read Procedure

3.2.3.1. Initialize

The following shows the initialization procedure for EEPROM control module..

[Main Routine]

1) Set HSCLK
   - Since SSIO uses HSCLK as a clock, it is necessary to set the settings for HSCLK before initializing the EEPROM control module. The sample program sets HSCLK as 500KHz.

2) Set SSIO interrupt
   - Set SSIO interrupt processing in the interrupt vector of SSIO. The ssio_continue function is called from there.

3) Initialize EEPROM control module

Perform the following initialization using the **eeprom_init function**:

- Set the settings for the P41 pin as the chip select (CS) pin.
- Initialize the control parameters of the EEPROM control module (put the module into an EEPROM read/write stop state).
- Set the following communication conditions and initialize the SSIO communication control module:
  ①Transfer clock : 125KHz (1/4 HSCLK)
  ②Transfer bit order : MSB first
  ③Bit length : 16 bit
  ④Clock output phase : High level in default

3.2.3.2. Write data

The sample program sends WEN command before WRITE command, because the EEPROM status transits to "disable write" after each WRITE command is executed.
The following shows the EEPROM write procedure.

1) Start writing N-word data
   - Call **eeprom_write function** with the following transmit data information. This function tramsmits WEN command and starts the command tramsmission processing for writing data to EEPROM.
     ①Address at which EEPROM write is started
     ②Initial address of the area that contains transmit data (an even address)
     ③Transmit data size (in bytes) * If an odd number is specified, the function returns en error.
     * Because the transmit data buffer is read or written by word accessing, the address of the transmit data must be an even address, and the transmit data size must be an even number in bytes.

2) Judge the completion of writing
   - Judge the completion of writing in the **eeprom_continue function**.

[Callback function upon SSIO tramsmission/reseption completion]

1) Continue the write processing
   - The callback function is called when each command or data transmission/reception is completed. This function performs the following processing. In detail, please see "3.2.3.4 Read/Write Processing Detail".
     ①If there is the remained data for writing, start tramsmission of the remained data. Otherwise, completes the write processing.
     ②While writing to EEPROM, stop SSIO and watch the busy status by setting the SSIO port as primary function (input), in order to judge the completion of EEPROM writing.
     This function completes the write processing when no data remains or an error is occurred.

3.2.3.3. Read data

The following shows the EEPROM read procedure.

1) Start reading N-word data
   - Call **eeprom_read function** with the following transmit data information. This function starts the command tramsmission processing for reading data from EEPROM.
     ①Address at which EEPROM read is started
     ②Initial address of the area that contains the receive data (an even address)
     ③Receive data size (in bytes) * If an odd number is specified, the function returns en error.
     * Because the transmit data buffer is read or written by word accessing, the address of the transmit data must be an even address, and the transmit data size must be an even number in bytes.

2) Judge the completion of reading
   - Judge the completion of reading in the **eeprom_continue function**.

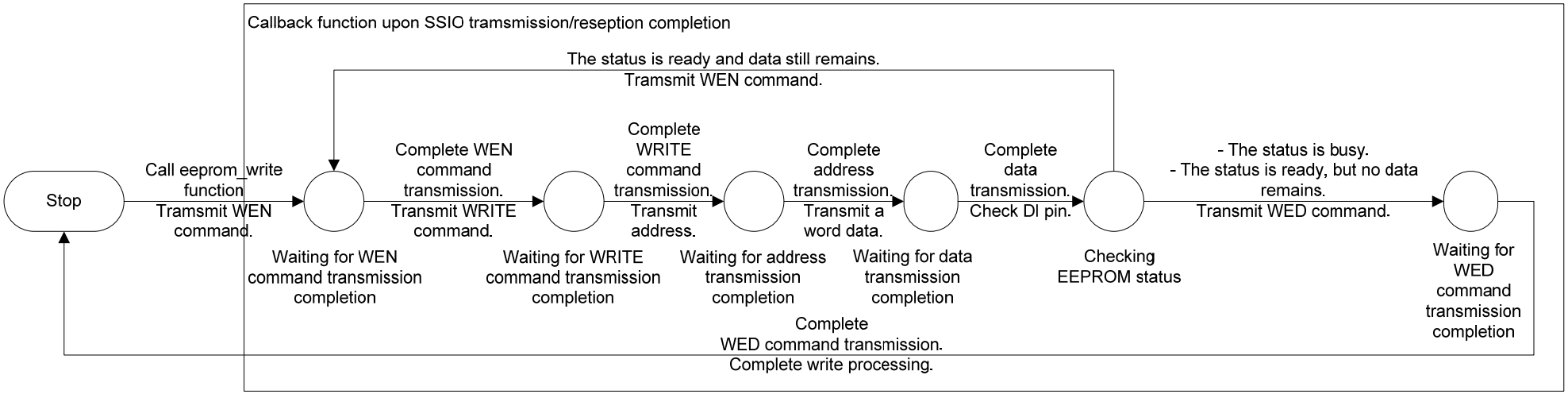
[Callback function upon SSIO tramsmission/reseption completion]

1) Continue the read processing
   - The callback function is called when each command or data transmission/reception is completed. This function performs the processing for reading data and its completion. In detail, please see "3.2.3.4 Read/Write Processing Detail".

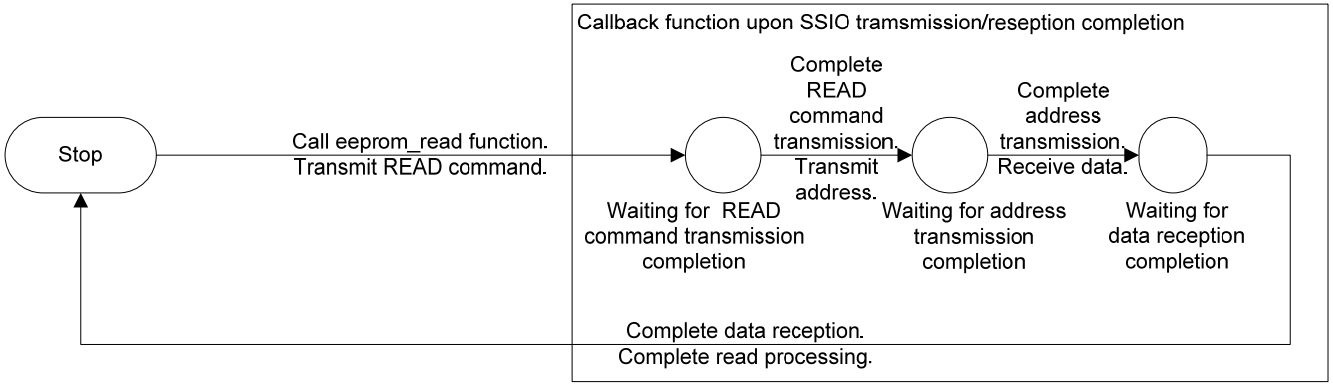
3.2.3.4. Read/Write Processing Detail

The following shows the state transition diagrams of the EEPROM control module in read and write processing.
Comments at upper part and lower part of each transition mean the condition of the transition and the action on the transition, respectively.

In EEPROM Write Processing



In EEPROM Read Processing



4. Description of the Sample Program

The following shows the functional specification of the sample program.

4.1. Function Overview

The sample program receives character data by UART until a termination character (carriage return: ¥r) is received. When a termination character is received, it writes the received data, except a termination character, to EEPROM. And then, it reads the data that was just written to EEPROM and tramsmits the read data by UART. But, When it receives 128 byte character before a termination character, it starts the above operation (write, read and tramsmit data) immediately.

* When only one termination character is received, it does not operate EEPROM.
* In the case that Microwire bus interface is enabled (_EEPROM_IF_TYPE is defined as 3), if the received data size except a termination character is odd size, the last data is not written to EEPROM because the data is written per 16 bit.
* The interface of EEPROM is selected by the following macro definition.

Macro	Value	Enabled Bus interface
_EEPROM_IF_TYPE	1	I2C
	2	SPI
	3	Microwire

4.2. Operation conditions

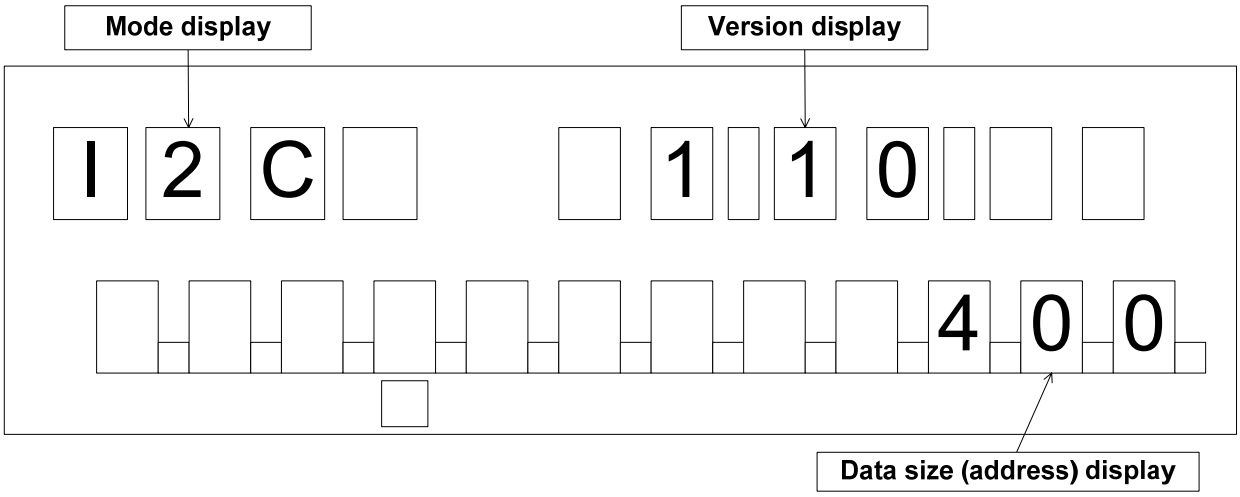
1) System clock
   - SYSCLK=HSCLK (RC oscillation mode 500 kHz)
2) UART
   - 9600 bps, 8-bit, no parity, 1 Stop bit, positive logic, LSB first
     * To use RS232C interface mounted on ML610Q400 Series Demo Kit, it is necessary to set P42 and P43 as a secondary function by selection of a port function jumper switch (short-circuit between 2-1 pins) on ML610Q400 Series Demo Kit.
3) LCD driver
   - Bias voltage multiplying clock: 2 kHz, Bias: 1/4 bias, Duty: 1/4 duty
   - Frame frequency : 73 Hz

4.3. Configuration of the LCD Panel

The following subsections describe the LCD panel configuration and types of display.
The LCD panel has two types of display patterns depending on the type of the LCD driver built into the MCU: one with the display allocation function and the other without it. The section from here onward assumes that the LCD panel is equipped with the display allocation function.

4.3.1. LCD Display Image with Display Allocation Function

The display allocation function is available if DSPMOD1's DASN (bit 2) can be set to "1".
The display image in this case is shown below.



Name	Content to be displayed
Mode display	Displays the name of the selected EEPROM bus interface. I2C interface: "I2C", SPI interface: "SPI", Microwire interface: "MCWR"
Version display	Displays the version information of the firmware.
Data size (address) display	Displays the data size which is already written. It is the same value as the offset address of the data area which the next data is written to.

4.4. Key Event

The key event that the sample program handles is the short-press release.

Short-press release: Polling is performed at 128-Hz intervals from the time a key was pressed, and a short-press release is confirmed if a match occurs four times but the key is released in less than 2 seconds.
Priority: Key S1 > Key S2 > Key S3 > Key S4

4.5. Function Details

4.5.1. State transition

The following shows the state transision diagram of the sample program.

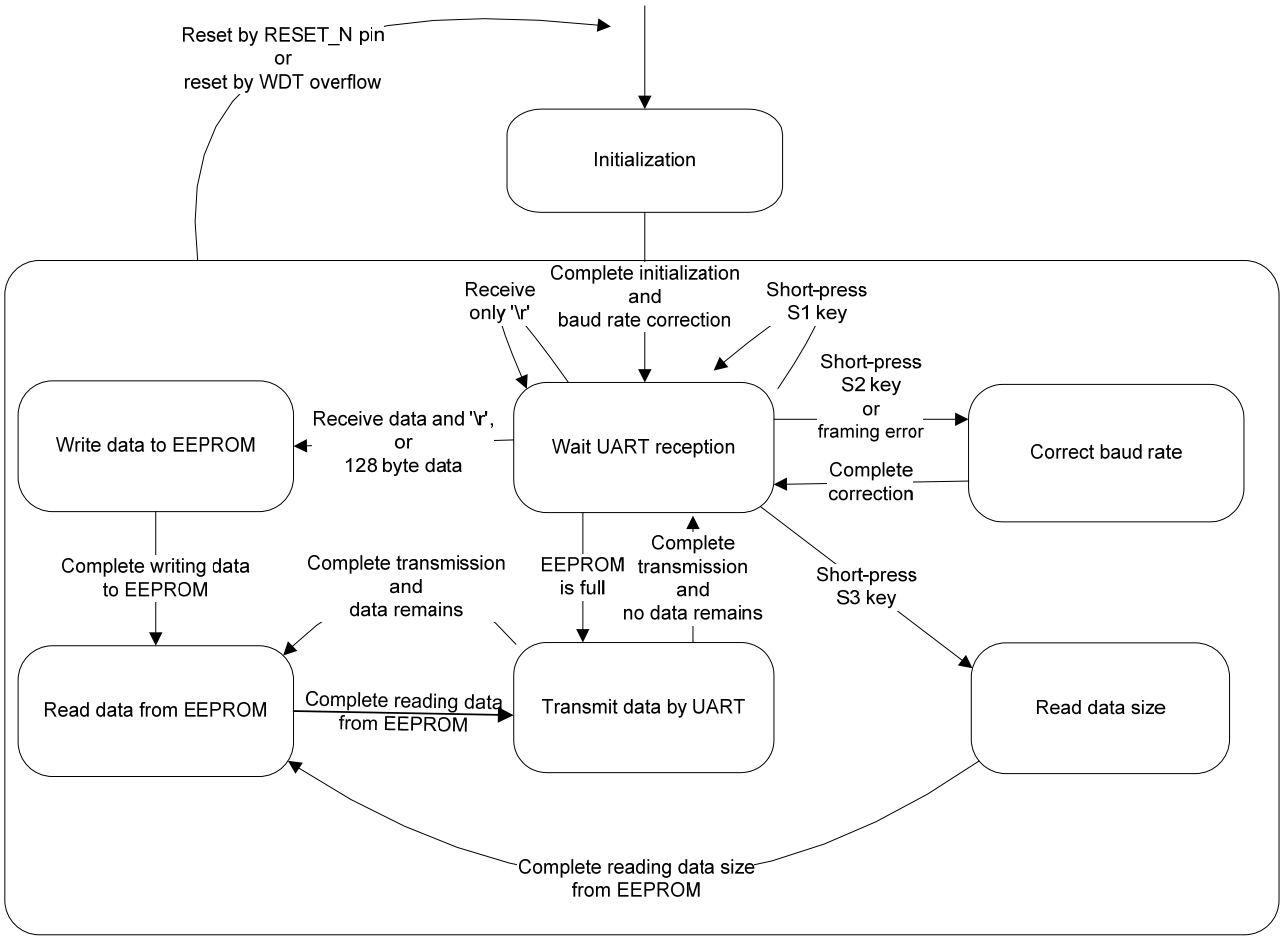


Figure 4-1 State Transition Diagram

State	Description
Initialization	The sample program transits to this state just after power-on. • Corrects baud rate. • Reads signature that is stored in EEPROM. If the signature is "U8EEPROMSample", transits "Wait UART reception" state.
Wait UART reception	The sample program normally stays this state. • Displays the current value of the data size (that is, the target address which the next data will be written to EEPROM), to the data size (address) area in LCD panel. • When '¥r' or 128 bytes data is received, writes the received data (Max. 1Kbytes) and the target address to EEPROM. • If the received data size is over the rest of EEPROM data area, transmits an error message by UART. • When S1 key is short-pressed, clears the data size (the target address). • When S2 key is short-pressed, corrects baud rate. • When S3 key is short-pressed, transmits all data which is written from the address 0000h in EEPROM data area.
Transmit data by UART	• When the data transmission is completed, transits "Wait UART reception" state. The data is transmitted per 128 bytes. In the case that the transmission data is remained, transits "Read data from EEPROM" state. • While data transmission, the key operation and data reception from UART can not be accepted.
Write data to EEPROM	• When writing all received data is completed, transits "Read data from EEPROM" state. • While writing data to EEPROM, the key operation and data reception from UART can not be accepted.
Read data from EEPROM	• When reading data from EEPROM is completed, transits "Transmit data by UART" state. • While reading data from EEPROM, the key operation and data reception from UART can not be accepted.
Correct baud rate	• When baud rate correction is completed, transits "Wait UART reception" state. • While baud rate correction, UART is stopped and data reception from UART can not be accepted. • The key operation can not be accepted.
Read data size	• Reads data size which has already been written to EEPROM, and then transits "Read data from EEPROM" state. • While reading data size from EEPROM, the key operation and data reception from UART can not be accepted.

4.5.2. Description of UART Display

```
[Read data]abcdefg
…
[Read All data]12345678901234567890
…
[Error]EEPROM is full.
…
```

Content to be displayed	Display Data
Read data from EEPROM	Displays data which is read from EEPROM, after [Read data] display. (When data is received by UART)
All read data from EEPROM	Displays all data which is read from EEPROM, after [Read All data] display. (When S3 key is short-pressed.)
Error output	When EEPROM is full, displays "[Error]EEPROM is full.".

4.5.3. UART Data Formats

The following shows the UART transmission data formats.

Table 4-1 UART transmission data format in normal case

Offset	Size (byte)	Content of data	Value of data	Remarks
0	11	Start message	“[Read data]”	
11	Max. 128	Read data from EEPROM	(ASCII characters)	
over 12	1	Carriage return	“¥r”	
Total	Max. 140			

* Use text data for the value of data.

Table 4-2 UART transmission data format in transmitting all data in EEPROM

Offset	Size (byte)	Content of data	Value of data	Remarks
0	15	Start message	“[Read All data]”	
15	Max. 1024	Read data from EEPROM	(ASCII characters)	
over 16	1	Carriage return	“¥r”	
Total	Max. 1040			

* Use text data for the value of data.

Table 4-3 UART transmission data format in error output

Offset	Size (byte)	Content of data	Value of data	Remarks
0	7	Start message	“[Error]”	
7	15	Error message	“EEPROM is full”	
22	1	Carriage return	“¥r”	
Total	23			

* Use text data for the value of data.

4.5.4. EEPROM Memory Map

The memory map of EEPROM in the sample program is shown below.

4.5.4.1. SPI bus interface EEPROM

Address	Area name	Size
0x0000 to 0x000F	Management information storage area	16 bytes
0x0010 to 0x040F	UART received data storage area	1024 bytes
0x0410 to 0x1FFF	Unused area	7152 bytes

BR25L640F-W (Capacity 64Kbit : 8K × 8bit)

4.5.4.2. Microwire bus interface EEPROM

Address	Area name	Size
0x0000 to 0x0007	Management information storage area	16 bytes
0x0008 to 0x0207	UART received data storage area	1024 bytes
0x0208 to 0x03FF	Unused area	1008 bytes

BR93L86F-W (Capacity 16Kbit : 1K × 16bit)

4.5.4.3. I2C bus interface EEPROM

Address	Area name	Size
0x0000 to 0x000F	Management information storage area	16 bytes
0x0010 to 0x040F	UART received data storage area	1024 bytes
0x0410 to 0x7FFF	Unused area	31728 bytes

BR24S256FJ-W (Capacity 256Kbit : 32K × 8bit)

4.5.4.4. Storage Data Formats

Data is sotred in the following data format.

Table 4-4 Storage Data Format (SPI bus interface EEPROM)

Offset	Size (byte)	Content of data	Value of data	Remarks
0	14	Signature	“U8EEPROMSample”	ASCII data
14	2	Data size stored in EEPROM (target address to write next data)	Number	Unit : Byte
16	From 0 to 1024	Received data by UART	ASCII data	
Total	1040			

Table 4-5 Storage Data Format (Microwire bus interface EEPROM)

Offset	Size (byte)	Content of data	Value of data	Remarks
0	14	Signature	“U8EEPROMSample”	ASCII data
7	2	Data size stored in EEPROM (target address to write next data)	Number	Unit : Byte
8	From 0 to 1024	Received data by UART	ASCII data	
Total	1040			

Table 4-6 Storage Data Format (I2C bus interface EEPROM)

Offset	Size (byte)	Content of data	Value of data	Remarks
0	14	Signature	“U8EEPROMSample”	ASCII data
14	2	Data size stored in EEPROM (target address to write next data)	Number	Unit : Byte
16	From 0 to 1024	Received data by UART	ASCII data	
Total	1040			

Revision History

Revision History

Edition	Date	Page		Description
		Previous Edition	Current Edition	
1	January 27, 2010	–	–	Initial Edition
2	April 16, 2010	7-8	7-8	List of Folders and Files is updated.
		9	9	Build procedure is updated.
		–	10-11	Description of Restrictions is added.