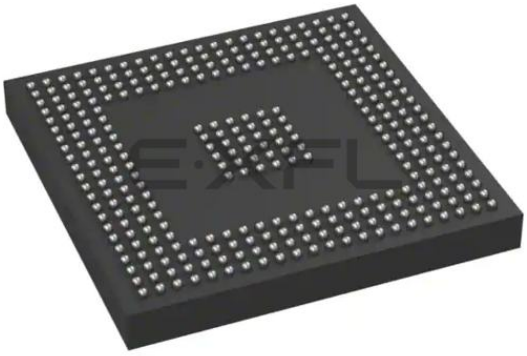

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	ARM® Cortex®-M3
Core Size	32-Bit Single-Core
Speed	100MHz
Connectivity	CANbus, CSI, Ethernet, I²C, UART/USART
Peripherals	DMA, WDT
Number of I/O	96
Program Memory Size	-
Program Memory Type	ROMless
EEPROM Size	-
RAM Size	512K x 8
Voltage - Supply (Vcc/Vdd)	0.9V ~ 3.6V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	324-FBGA
Supplier Device Package	324-FBGA (19x19)
Purchase URL	https://www.e-xfl.com/product-detail/renesas-electronics-america/upd60510f1-hn4-m1-a

How to Use This Manual

1. Purpose and Target Readers

This manual is intended for users who wish to understand the functions of industrial Ethernet network LSI “R-IN32M3-EC/CL” and design application systems using it.
Target users are expected to understand the fundamentals of electrical circuits, logic circuits, and microcomputers.

When designing an application system that includes this MCU, take all points to note into account. Points to note are given in their contexts and at the final part of each section, and in the section giving usage notes.

The list of revisions is a summary of major points of revision or addition for earlier versions. It does not cover all revised items. For details on the revised points, see the actual locations in the manual. The mark “<R>” in the text indicates the major revisions to this version. You can easily find these revisions by copying “<R>” and entering it in the search-string box for the PDF file.

Literature — Literature may be preliminary versions. Note, however, that the following descriptions do not indicate "Preliminary". Some documents on cores were created when they were planned or still under development. So, they may be directed to specific customers.

Documents related to the R-IN32M3 Series

Document Name	Document Number
R-IN32M3 Series Datasheet	R18DS0008EJ0200
R-IN32M3-EC User’s Manual	R18UZ0003EJ0101
R-IN32M3-CL User’s Manual	R18UZ0005EJ0101
R-IN32M3 Series User’s Manual: Peripheral Modules	R18UZ0007EJ0301
R-IN32M3 Series Programming Manual: Driver	This manual
R-IN32M3 Series Programming Manual: OS	R18UZ0011EJ0300
R-IN32M3 Series User's Manual: TCP/IP Stack	R18UZ0019EJ0200

List of Tables

2.4.2 ./ Device /Renesas/RIN32M3/Source/Middleware: Middleware

The configuration of source files for the middleware is listed below.

Table 2.6 Configuration of Files in Middleware Directories

Directory	File	Contents
eeprom/	eeprom.c	EEPROM middleware sample source
	eeprom.h	EEPROM middleware header file
flash/	flash.c	Parallel flash ROM middleware sample source
	flash.h	Parallel flash ROM middleware header file
sflash/	sflash.c	Serial flash ROM middleware sample source
	sflash.h	Serial flash ROM middleware header file

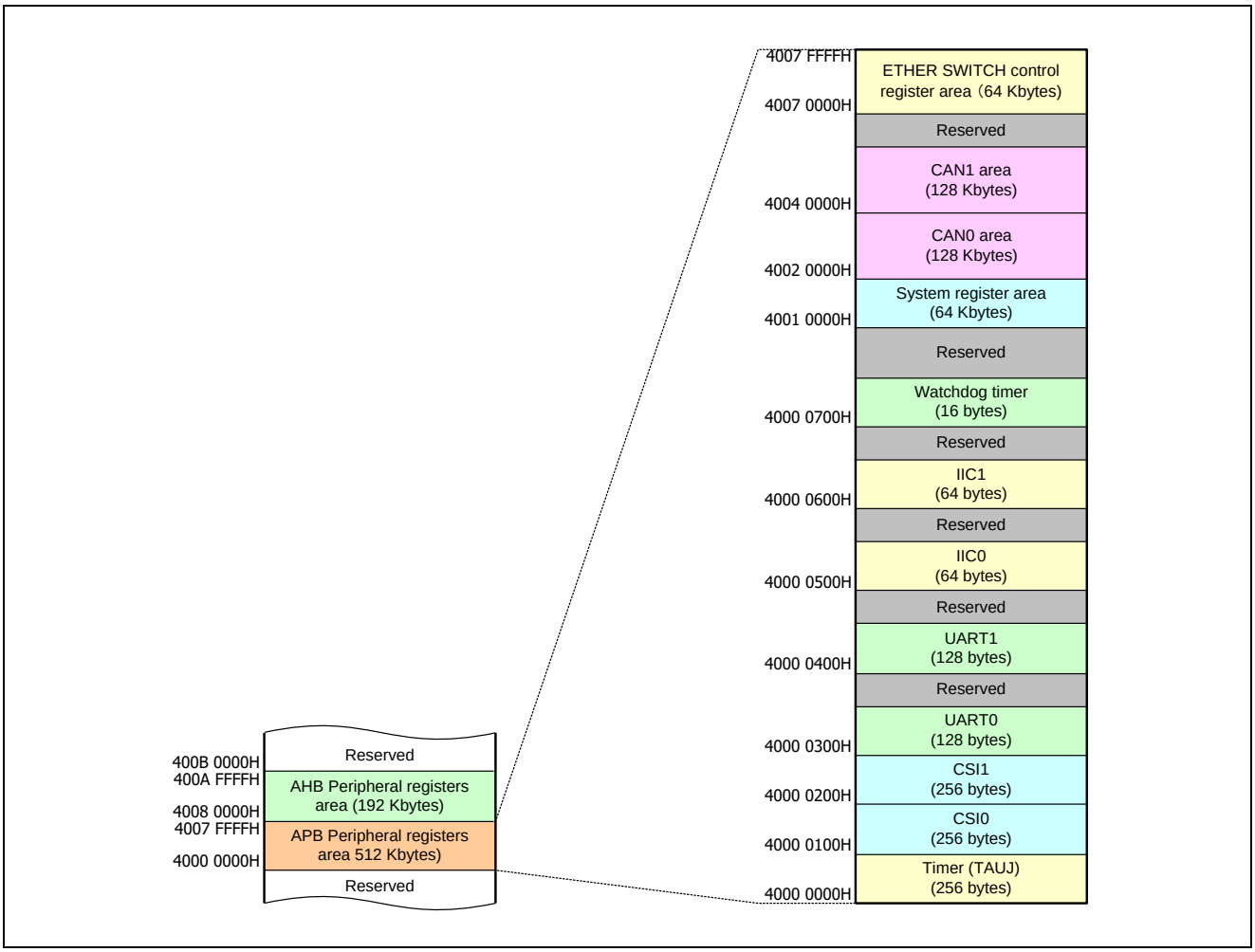

Figure 3.4 Memory Map (APB Peripheral Registers Area) (common to R-IN32M3-EC/CL)

Table 6.4 CSI Driver Functions

Function Name	Summary
csi_init	Initialization of CSI controller
csi_write	Transmit one byte of character data
csi_read	Receive one byte of character data
csi_check_tx	Check transmission data (for slave)
csi_check_rx	Check received data (for slave)
csi_change_mode	Tx/Rx mode change (for slave)

Table 6.5 DMAC Driver Function

Function Name	Summary
dmac_memcpy	Copy memory (DMA transfer)

Table 6.6 Serial Flash ROM Driver Functions

Function Name	Summary
sromc_init	Initialization of SPI bus controller
sromc_write	Write data to SPI bus
sromc_read	Read data form SPI bus

Table 6.7 Watchdog Timer Driver Functions

Function Name	Summary
wdt_init	Initialization of watchdog timer
wdt_start	Watchdog timer activation
wdt_clear	Counter clearing
wdt_wait_reset	Wait for reset

6.2.3 Initialization of One-Count Timer (triggered by hardware)

timer_onecount_hwtrg_init

(1) Description

Initialization of the one-count timer (triggered by hardware)

(2) C-Language Format

```
ER_RET timer_onecount_hwtrg_init( uint8_t ch,uint32_t o_time, uint32_t trg );
```

(3) Parameter

I/O	Parameter		Description
I	uint8_t	ch	Channel selection argument 0: channel- 0 1: channel- 1 2: channel- 2 3: channel- 3
I	uint32_t	o_time	Time for counting once (1 us to 42,949,672 ns)
I	uint32_t	trg	Trigger source selection argument (the number of IRQ in trigger sources +4)

(4) Function

This function sets the timer selected by the channel selection argument to one-count timer mode. This timer is triggered by the interrupt signal which is selected by the trigger source selection argument.
The timer stops counting after the specific time given by the “time for counting once” argument has elapsed.
The timer does not detect a trigger during counting.
A parameter error is returned if the value of the channel selection argument or the time for counting once is not available.
Set the frequency of the clock to drive counting by the timer to 100 MHz for this operation.

Caution: The interrupt will not be detectable if the counter clock period is longer than the interrupt pulse width.

(5) Return Value

Return Value	Meaning
ER_OK	Initialization succeeded
ER_PARAM	Parameter error - The selected channel is NOT 0 to 3 - The time for counting once is NOT 1 to 42,949,672 ns

Remark: This API uses “delay counting” described in “TAUJ2 Operations” in “R-IN32M3 Series User’s Manual: Peripheral Modules”.

6.3.4 Confirming Presence of Received Data

uart_check_receivedata

(1) Description

Checking the presence of received data

(2) C-Language Format

```
ER_RET uart_check_receivedata(uint8_t ch);
```

(3) Parameter

I/O	Parameter	Description
I	uint8_t ch	Channel selection argument 0: channel-0 1: channel-1

(4) Function

This function checks whether RX_FIFO of the selected channel is empty.
ER_PARAM is returned if the selected channel is not 0 or 1.
Selection of the channel is defined in system_RIN32M3.h.

(5) Return Value

Return Value	Meaning
1	The buffer holds received data
0	No received data
ER_PARAM	Parameter error - The selected channel is not 0 or 1

6.4 IIC Control

6.4.1 Initialization of IIC Controller<R>

iic_init

(1) Description

Initialization of the IIC controller

(2) C-Language Format

```
ER_RET iic_init(uint8_t ch);
```

(3) Parameter

I/O	Parameter		Description
I	uint8_t	ch	Channel selection argument 0: channel-0 1: channel-1

(4) Function

This function makes initial settings for the IIC controller of the selected channel.
ER_PARAM is returned if the selected channel is not 0 or 1.

- IIC clock setting
  - > Fast mode : 400 kHz
- IIC timing setting
  - > Stop and start interval : 80 × PCLK
  - > Setup cycles
    - Start condition : 80 × PCLK
    - Stop condition : 45 × PCLK
    - Write data : 2 × PCLK
  - > Hold cycles
    - Start condition : 45 × PCLK
    - Data : 80 × PCLK
    - Write data : 0 × PCLK
    - Read data : 2 × PCLK

Remark: The IIC clock setting “400 kHz” is based on the assumption that both the rise and fall times of SDAn and SCLn are 20 ns. Change the register settings appropriately according to your usage environment. For details, refer to the R-IN32M4 Series User’s Manual: Peripheral Modules.

6.4.4 Transmission of One Byte of Character Data

iic_write

(1) Description

Transmission of one byte of character data

(2) C-Language Format

```
ER_RET iic_write(uint8_t ch, uint8_t data);
```

(3) Parameter

I/O	Parameter	Description
I	uint8_t ch	Channel selection argument 0: channel-0 1: channel-1
I	uint8_t data	One byte of character data for transmission

(4) Function

This function transmits one byte of character data to the selected channel.
ER_PARAM is returned if the selected channel is not 0 or 1.
ER_NG (transmission failed) is returned in cases where ACK is not returned from the device each time 8-bit data is transmitted.

(5) Return Value

Return Value	Meaning
ER_OK	Transmission succeeded
ER_NG	Transmission failed - ACK is NOT returned from the device
ER_PARAM	Parameter error - The selected channel is not 0 or 1

6.5.6 Switching Tx/Rx Mode (for Slave)

csi_change_mode

(1) Description

Switching Tx/Rx mode

(2) C-Language Format

```
ER_RET csi_change_mode(uint32_t ch, uint32_t mode);
```

(3) Parameter

I/O	Parameter	Description
I	uint32_t ch	Channel selection argument 0: channel-0 1: channel-1
I	uint32_t mode	Transfer mode selection argument 0: Rx mode 1: Tx mode

(4) Function

This function sets CSI Tx/Rx mode for the selected channel.
ER_PARAM is returned if the channel selection argument or transfer mode selection argument is not 0 or 1.

- If the transfer mode selection argument is Rx mode, the setting is changed as below.
  - > Stopping Tx operation
  - > Enabling Rx operation
- If the transfer mode selection argument is Tx mode, the setting is changed as below.
  - > Tx operation is permit
  - > Rx operation is prohibit

(5) Return Value

Return Value	Meaning
ER_OK	Mode switching succeeded
ER_PARAM	Parameter error - The selected channel is not 0 or 1 - The selected mode is not 0 or 1

6.8.2 Starting Watchdog Timer

wdt_start

(1) Description

Starting the watchdog timer

(2) C-Language Format

```
ER_RET wdt_start(void);
```

(3) Parameter

None

(4) Function

This function starts the watchdog timer. The watchdog timer cannot be stopped once it is started.

(5) Return Value

Return Value	Meaning
ER_OK	Timer started

6.8.4 Waiting for Reset

wdt_wait_reset

(1) Description

Waiting for a reset

(2) C-Language Format

```
void wdt_wait_reset(void);
```

(3) Parameter

None

(4) Function

This is for waiting for the reset signal for the watchdog timer to be output in response to the overflow of the counter.

(5) Return Value

None

7.2.3 Reading EEPROM Data

eep_read

(1) Description

Reading EEPROM data

(2) C-Language Format

```
ER_RET eep_read(uint32_t eep_addr, uint8_t *data, uint32_t len);
```

(3) Parameter

I/O	Parameter		Description
I	uint32_t	eep_addr	Address where reading of data starts
O	uint8_t *	data	Pointer to the received data
I	uint32_t	len	Amount of data to be read

(4) Function

This function activates the IIC controller to receive the amount of data specified by len in the EEPROM from the eep_adr address. The read data in the EEPROM is written from the *data specified address to the len specified size of area. ER_PARAM is returned if the specified end address is over the device capacity (device-specific).

If the value returned from the IIC controller is not ER_OK (transmission succeeded), the operation is as follows.

- Transmission of device selection information : A stop condition is issued and the write failure status is returned.
- Transmission of byte address (high) : A stop condition is issued and the write failure status is returned.

  Transmission of byte address (low) : A stop condition is issued and the write failure status is returned.
- Transmission of device selection information to be read : The communication failure status is returned
- Transmission of the specified amount of data to be written : A stop condition is issued and the write failure status is returned.

(5) Return Value

Return Value	Meaning
ER_OK	Success in reading
ER_NG	Failure in reading - The result of IIC driver processing is an error.
ER_PARAM	Parameter error - The specified end address is over the device capacity.

7.3.2 Writing Data

flash_program

(1) Description

Writing data

(2) C-Language Format

```
ER_RET flash_program(uint16_t* buf, uint32_t addr, uint32_t size);
```

(3) Parameter

I/O	Parameter		Description
I	uint16_t*	buf	Address where the write data storage area starts
I	uint32_t	addr	Write address
I	uint32_t	size	Amount of data to be written (in bytes)

(4) Function

The function writes the amount of data specified by the size argument from the write address specified by the addr argument to the parallel flash ROM area.
The written data of the address area specified by the *buf argument is used.

(5) Return Value

Return Value	Meaning
ER_OK	Success
ER_PARAM	Parameter error - The addr argument is NOT a 16-bit address threshold value - The size argument is NOT a 16 bit address threshold value - The additional value of the addr argument and the size argument is over the maximum size of parallel flash ROM

7.3.3 Reading Data

flash_read_data

(1) Description

Reading data

(2) C-Language Format

```
ER_RET flash_read_data(uint16_t* buf, uint32_t addr, uint32_t size);
```

(3) Parameter

I/O	Parameter		Description
O	uint16_t*	buf	Address where writing of read data starts
I	uint32_t	addr	Read address
I	uint32_t	size	Amount of data to be read (in bytes)

(4) Function

The function reads the amount of data specified by the size argument in the parallel flash ROM area from the read address specified by the addr argument.
The data read is written to the address area specified by the *buf argument.

(5) Return Value

Return Value	Meaning
ER_OK	Success
ER_PARAM	Parameter error - The addr argument is NOT a 16-bit address threshold value - The size argument is NOT a 16 bit address threshold value - The additional value of the addr argument and the size argument is over the maximum size of parallel flash ROM

List 8.13 EEPROM Sample: Help Command Execution

```
I2C EEPROM Writer
> h                                    (← indicate command help)
  r [addr] [size]                      (← read command argument is start address and read size)
  w [addr] [data]                      (← write command argument is start address and 8-bit data)
  d [addr]                             (← download command argument is start address)
  h                                    (← help command)
```

List 8.14 EEPROM Sample: Other Command Execution (command error)

```
I2C EEPROM Writer
> x 0 800                              (← the commands except r,R/w,W/d,D/h,H are error))
  Command error !!
```

9.1.2 Replacing Library Functions

Library functions are redefined in syscalls.c. <R>

Table 9.1 Replacing ARM Library Functions

Function Name	Description
fputc	Handles transmission of one byte of data to the UART. The return value is Tx data.
fgetc	Handles reception of one byte of data from the UART and echoes back the result of reception. The return value is Rx data.
ferror	Handles processing for a file error. Processing is not implemented. The return value is EOF.
_sys_exit	Handles processing for exiting the system. The internal function enters an endless loop.
_ttywrch	Handles transmission of one byte of data to the UART. There is no return value.
_clock_init	No operation
clock	The elapsed time (clock_t type) is returned with the interval counter of the interval timer. The time precision is 80 ns.

9.3 IAR

9.3.1 Startup

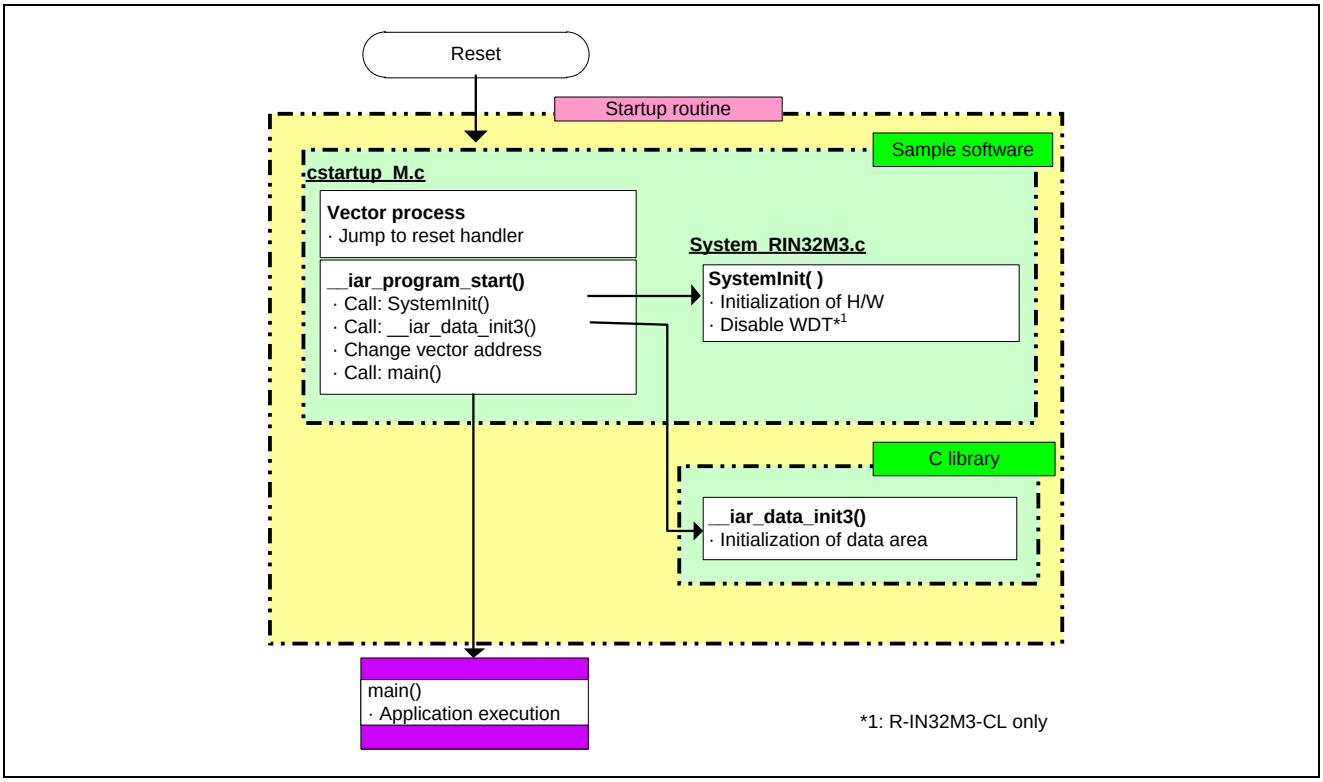

Figure 9.3 Startup Routine with IAR

[Memo]

R-IN32M3 Series
Programming Manual: Driver

Renesas Electronics Corporation