RX Family

R01AN3036EJ0111
Rev.1.11
Nov 11, 2016

Ethernet Simple Switch Function Using Firmware Integration Technology Modules

Introduction

This document explains the usage example of the simple switch function implemented in the Ethernet peripheral module of the RX64M/71M using EPTPC Light FIT (Firmware Integration Technology) module [1]. The simple switch function controls the inter ports frame transfer of the Ethernet MAC modules and the transfer method is selectable store & forward or cut-through one.

Target Device

This API supports the following device.

- RX64M Group
- RX71M 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. Overview

This document explains the usage example of the simple switch function implemented in the Ethernet peripheral module of the RX64M/71M using EPTPC Light FIT module (hereafter PTP light driver). The simple switch function controls the inter ports frame transfer of the Ethernet MAC modules and the transfer method is selectable store & forward or cut-through one.

This example is composed of a switch project and a test project. The switch project operates the simple switch function to the receiving test frame from one port. The test project creates the test frames whose data contents are read from the USB memory and issues those frames. If any relayed frames via RX64M/71M implemented to the switch project existed, the test project receives relayed frames, and measures the propagation and operation time of them.

1.1 Ethernet Simple Switch Function Using FIT Modules

This example is implemented in two projects and used as the operation example the simple switch function using PTP light driver.

1.2 Related documents

[1] RX Family EPTPC Light Module Using Firmware Integration Technology, Rev.1.11, Document No. R01AN3035EJ0111, Nov 11, 2016

[2] RX Family Ethernet Module Using Firmware Integration Technology, Rev.1.12, Document No. R01AN2009EJ0112, Nov 11, 2016

[3] RX Family Flash Module Using Firmware Integration Technology, Rev.1.20, Document No. R01AN2184EU0120, Dec 22, 2014

[4] Renesas USB MCU USB Basic Host and Peripheral firmware Using Firmware Integration Technology, Rev.1.10, Document No. R01AN2025EJ0110, Dec 26, 2015

[5] Renesas USB MCU USB Host Mass Storage Class Driver (HMSC) Using Firmware Integration Technology, Rev.1.10, Document No. R01AN2026EJ0110, Dec 26, 2015

[6] RX Family Open Source FAT File System [M3S-TFAT-Tiny] Module Firmware Integration Technology, Rev.3.00, Document No. R20AN0038EJ0300, Apr 01, 2014

[7] RX Family EPTPC Module Using Firmware Integration Technology, Rev.1.12, Document No. R01AN1943EJ0112, Nov 11, 2016

[8] RX64M Group Renesas Starter Kit+ User's Manual For e^2 studio, Rev. 1.10, Document No. R20UT2593EG0110, Jun 25, 2015

[9] RX71M Group Renesas Starter Kit+ User's Manual, Rev. 1.00, Document No. R20UT3217EG0100, Jan 23, 2015

1.3 Hardware Structure

This example uses the Ethernet peripheral modules of the RX64M/71M. The Ethernet peripheral modules are composed of the EPTPC, the PTP Host interface peripheral module (PTPEDMAC), dual channel Ethernet MAC ones (ETHERC (CH0), ETHERC (CH1)) and dual channel Ethernet Host interface ones (EDMAC (CH0), EDMAC (CH1)).

Furthermore, the test project use the USB module (USB), Multi-Function Timer Pulse Unit (MTU3) module and data flash of the RX64M/71M.

In detail, please refer to RX64M/71M Group User's Manual: Hardware.

1.4 Software Structure

This sample is operations example of the application layer applied to plural FIT modules. The application is different from the switch project and test project. The switch project manages the operation sequence and set the simple switch function. The test project manages the operation sequence, loads/writes the MAC address from/to the data flash, creates and verifies standard frame (hereafter frame), reads test data form the USB memory, writes transferred data and saves result data to the USB memory. The Ether driver [2] of each channel transmits and receives frames, MTU3 driver measures the propagation and operation time of the transferred frame, the data flash driver [3] erases and writes communication parameter[1] to the data flash. The USB Host driver [4], [5] accesses the USB memory as the logical block unit. The FAT file system (M3S-TFAT-Tiny) [6] manages the data in the USB memory as the file using the USB Host driver. The switch project only uses the two channel Ether driver and PTP Light driver. Figure 1.1 shows the typical structure and functional overview of the software.

[1] MAC address in this example.

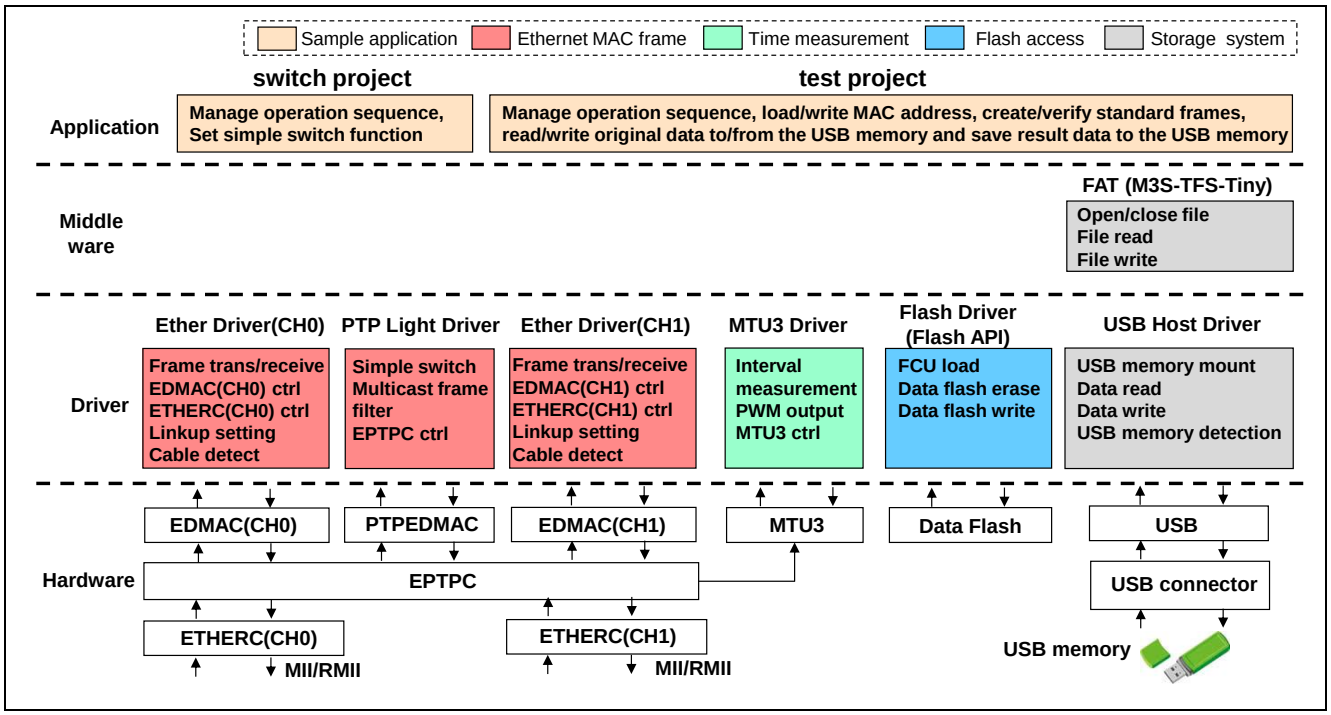

Figure 1.1 Software structure of this sample

1.5 File Structure

This sample codes are stored the “demo_src” and lower hierarchical folders. Figure 1.2 and Figure 1.3 show the source and header file structures of switch project and test project respectively. The sample data of the test project are stored in the USB memory. As for the detailed information of the FIT based modules (BSP, Ethernet Driver, Flash Driver, PTP Light Driver, FAT file system and USB Driver), please refer to the documentation of the each FIT module.

```
demo_src: main operation
|   sample_main.c
|   sample_main.h
|
+ --- usr: LED control
|       led.c
|       led.h
|
```

```
r_bsp: BSP (Board Support Package) FIT module

r_config: configuration setting of FIT modules
|       r_bsp_config.h
|       r_bsp_interrupt_config.h
|       r_ether_rx_config.h
|       r_ptp_light_rx_config.h

r_ether_rx: Ethernet Driver FIT module

r_ptp_light_rx: PTP Light Driver FIT module
```

Figure 1.2 File structure of switch project

```
demo_src: main operation
|  sample_main.c
|  sample_main.h
|
+ --- ether_if: Ethernet frame operation
|      ether_if.c
|      ether_if.h
|
+ --- flash_if: Data flash access operation
|      flash_if.c
|      flash_if.h
|
+ --- tfat_if: File system IF to USB driver
|      file_if.c
|      file_if.h
|      r_data_file.c
|      r_data_file.h
|      r_tfat_drv_if.c ;USB driver interface
|
+ --- tmr_if: Timer (MTU3) operation
|      tmr_if.c
|      tmr_if.h
|
+ --- usb_if: USB Host memory access control
|      r_usb_hmsc_defep.c
|      usb_memory_access.c
|
+ --- usr: LED control
|      led.c
|      led.h
|
+ --- usb_memory_sample: Sample of USB memory data (data parts of transfer frame)
|      + --- TEST: SRC_X.txt; Data files stored (X = 0,1,2,, 200)
```

```
r_bsp: BSP (Board Support Package) FIT module

r_config: configuration setting of FIT modules
|      r_bsp_config.h
|      r_bsp_interrupt_config.h
|      r_ether_rx_config.h
|      r_flash_rx_config.h
|      r_ptp_light_rx_config.h
|      r_usb_basic_config.h
|      r_usb_hmsc_config.h

r_ether_rx: Ethernet Driver FIT module

r_flash_rx: Flash Driver (Flash API) FIT module

r_ptp_light_rx: PTP Light Driver FIT module

r_tfat_rx: FAT file system (M3S-TFS-Tiny) FIT module
|      r_tfat_lib.h ;FAT library header file
|      + --- lib:  FAT library stored folder
|              r_mw_version.h ; middleware version information
|              r_stdint.h ; integer type definition
|              tfat_rx600_big.lib ; big endian
|              tfat_rx600_little.lib ; little endian
|

r_usb_basic: USB driver (USB basic operation) FIT module

r_usb_hmsc: USB driver (Host Mass Storage Class) FIT module
```

Figure 1.3 File structure of test project

2. Functional Information

This example is developed by the following principles.

2.1 Hardware Requirements

This driver requires your MCU supports the following feature:

- EPTPC
- ETHERC
- EDMAC
- MTU3[1]
- Data Flash[1]
- USB[1]

[1] test project only use.

2.2 Hardware Resource Requirements

This section details the hardware peripherals that this example requires. Unless explicitly stated, these resources must be reserved for the following driver, and the user cannot use them.

2.2.1 EPTPC Channel

This example uses the EPTPC. This resource needs to the simple switch function.

2.2.2 ETHERC Channel

This example uses the ETHEC (CH0) and ETHEC (CH1). Those resources need to the Ethernet MAC operations.

2.2.3 EDMAC Channel

This example uses the EDMAC (CH0) and EDMAC (CH1). Those resources need to the CPU Host interface of frame operations.

2.2.4 MTU3 Channel

This example uses the MTU3 (CH1) and MTU3 (CH2) for measurement the interval of the inter ports frame transfer applying to cascade connection.

2.2.5 Data Flash

This example uses the Data Flash to store the MAC address.

2.2.6 USB Channel

This example uses the USB 2.0 FS Host/Function Module to read test data from the USB memory and write transferred and result data to the USB memory.

2.3 Software Requirements

This example depends on the following packages (FIT modules):

- r_bsp
- r_ether_rx
- r_ptp_light_rx
- r_flash_rx[1]
- r_tfat_rx[1]
- r_usb_basic[1]
- r_usb_hmsc[1]

[1] test project only includes.

2.4 Limitations

There are following limitations in this example

- Not support PTP time synchronization.
- Cannot use the PTP driver (full version) [7] substitute for the PTP light driver.
- Cannot receive and process the PTP message frames[1] in the switch project.

[1]Relay control is possible.

2.5 Supported Toolchains

This example is tested and works with the following toolchain:

- Renesas RX Toolchain v2.05.00

2.6 Header Files

2.6.1 switch project

Each function call is accessed by including a single file, *r_ether_rx_if.h or r_ptp_light_rx_if.h* which is supplied with this project code.

2.6.2 test project

Each function call is accessed by including a single file, *r_ether_rx_if.h, r_ptp_light_rx_if.h, r_frash_rx_if.h, r_tfat_lib.h, r_usb_basic_if.h or r_usb_hmsc_if.h* which is supplied with this project code.

2.7 Integer Types

This project uses ANSI C99. These types are defined in *stdint.h*.

2.8 Configuration Overview

This section describes the configuration in this example.

2.8.1 switch project

The configuration options in this project are specified in *sample_main.h.* The option names and setting values are listed in the table below.

Configuration options	
#define LINK_CH **- Default value = 1**	Specify the Ethernet link channel at first. - When this is set to 0, Ethernet CH0 is selected. - When this is set to 1, Ethernet CH1 is selected.
#define NUM_CH **- Default value = 2**	Set the number of channels of the Ethernet controller. - Set 2 in this example.

2.8.2 test project

The configuration options in this project are specified in *sample_main.h and r_data_file.h.* The option names and setting values are listed in the table below.

Configuration options	
#define NUM_TEST **- Default value = 201**	Specify the total test times (frame transfer times). - Set 1 to 201 in this example.
#define LINK_CH **- Default value = 1**	Specify the Ethernet link channel at first. - When this is set to 0, Ethernet CH0 is selected. - When this is set to 1, Ethernet CH1 is selected.
#define NUM_CH **- Default value = 2**	Set the number of channels of the Ethernet controller. - Set 2 in this example.
#define SRC_IDX **- Default value = 0**	Set the frame transfer source channel. - Set 0 or 1, and different channel specified by "DST_IDX" in this

Configuration options	
	example.
#define DST_IDX **- Default value = 1**	Set the frame transfer destination channel. - Set 0 or 1, and different channel specified by "SRC_IDX" in this example.
#define MAC_ADDR_1H/2H **- Default value = 0x00007490**	Set the Ethernet MAC address upper 16 bits for port0/port1. The lower 16 bits of default value are set the upper 16bits of the Renesas vendor ID (=74-90-50). The upper 16 bits of default value are reserved field and should be set 00-00. If first time access or forcing erase option (=" FORCE_ERASE_PRM) defined, this address value is written to the data flash. Please change this value when users applied to this sample their own system.
#define MAC_ADDR_1L/2L **Case of SRC_IDX = 0,** **- Default value = 0x5000792C (port0)** **- Default value = 0x5000792D (port1)** **Case of SRC_IDX = 1,** **- Default value = 0x5000792E (port0)** **- Default value = 0x5000792F (port1)**	Set the Ethernet MAC address lower 32 bits for port0/port1. The upper 8 bits of default value are set the lower 8bits of the Renesas vendor ID (=74-90-50). The lower 24 bits of default value are set the unique value for this sample. If first time access or forcing erase option (=" FORCE_ERASE_PRM) defined, this address value is written to the data flash. Please change this value when users applied to this sample their own system.
#define FORCE_ERASE_PRM **- undefined**	Select whether erasing the MAC address written in the data flash after finishing operation or not? - If defined, the MAC address is erased from the data flash.
#define WAIT_MAX_CNT **- Default value = 1000000**	Set the loop maximum count value to wait read the reception frame. This value specifies the timeout of the frame reception.
#define RETRY_MAX_CNT **- Default value = 1000**	Set the read retry maximum times[1].
#define MAX_DAT_SIZE **- Default value = (1514 – 12)**	Set the maximum test file data size in byte unit to which equals the frame data field size. - Maximum value is 1502 byte.
#define READ_DIR **- Default value "TEST"**	Specify the directory name of the original test file stored.
#define READ_FILE **- Default value "SRC"**	Specify the file name of the original test file. This string is concatenated the file number assigned frame transmitted order and the file extension equal to ".txt". Ex. "SRC_0.txt", "SRC_1.txt", , "SRC_200.txt"
#define WRITE_DIR **- Default value "RCV"**	Specify the directory name of the received test file stored.
#define WRITE_FILE **- Default value "DST"**	Specify the file name of the received test file. This string is concatenated the file number assigned frame received order and the file extension equal to ".txt". Ex. "DST_0.txt", "DST_1.txt", , "DST_200.txt"
#define RESULT_FILE **- Default value "RESULT"**	Specify the file name of the result file to indicate file size and transfer interval time. This string is concatenated to the file extension equal to ".txt". Ex. "RESULT.txt"
#define FILESIZE **- Default value = 2048**	Specify the FAT file system data buffer size - Set 2048 in this sample.

[1] Read retry operation is need when received data was not transferred to the receive buffer by the EDMAC after frame reception interrupt detected.

2.9 Data Structures

This section details the data structures that are used with the functions of this example.

2.9.1 switch project

The simple switch function setting structure is only defined in *sample_main.c*.

```
/* TRNMR setting values table */
typedef struct
{
   RelEnabDir rel;
   TranMode trn;
} TrnTbl;
```

2.9.2 test project

Those data structure in this project are located in *sample_main.h and ether_if.h* as the prototype declaration.

```
/* Ether & USB access state */
typedef enum
{
   APL_START = 0, /* Operation start state */
   APL_READ,      /* USB memory read state */
   APL_COM,       /* Ether communication state */
   APL_WRITE,     /* USB memory write state */
   APL_STOP,      /* Operation stop state */
} APLState;
```

```
/* Data access information structure */
typedef struct
{
   uint16_t size[NUM_TEST]; /* Data size */
   int8_t *src;             /* Address of source data */
   int8_t *dst;             /* Address of destination data */
   uint32_t time[NUM_TEST]; /* Operation time */
} ACCInfo;
```

```
/* Ether communication state information */
typedef enum
{
   COM_ERR = -1, /* General error */
   COM_OK = 0,   /* No error */
   COM_TOUT,     /* Timeout occured */
   COM_OTH_SRC,  /* Received from other source */
   COM_OTH_CH,   /* Received from other channel */
   COM_ERR_FRM,  /* Error frame received */
} COMInfo;
```

2.10 Return Values

This section describes return values of the functions of this example. There is no return value in the switch project.

Those return values in the test project are located in *ether_if.h, flash_if.h and file_if.h* as the prototype declarations.

```
/* Ether access return value */
typedef enum
{
    ETHIF_ERR = -1, /* General error */
    ETHIF_OK = 0,
    ETHIF_TOUT,     /* Timeout occured */
} ethif_t;
```

```
/* Data flash access return value */
typedef enum
{
    FLSIF_ERR = -1, /* General error */
    FLSIF_OK = 0,
    FLSIF_ERASE_ERR,  /* Erase error */
    FLSIF_WRITE_ERR,  /* Write error */
    FLSIF_VERIFY_ERR, /* Verify error */
} flshif_t;
```

```
/* File access return value */
typedef enum
{
    FLIF_ERR = -1, /* General error */
    FLIF_OK = 0,
} flif_t;
```

3. Specification of This Example

3.1 Outline of Functions

The function of this example in switch project and test project show Table 3.1 and Table 3.2 respectively.

Table 3.1 Function of switch project

Item	Contents
main()	Main operation of this project.
wait_seq()	Wait operation of the sequence state transfer
EINT_Trig_isr()	Frame reception interrupt handler.
led_init()	Initialize user LED.
led_ctrl()	Update user LED pattern

Table 3.2 Function of test project

Item	Contents
main()	Main operation of this project.
led_init()	Initialize user LED.
led_ctrl()	Update user LED pattern
prm_init()	Initialize operation parameter.
usb_memory_start()	USB memory task start.
Sample_Task()	Sample application task.
EtherStart()	Ether start operation.
EtherCom()	Ether communication.
EINT_Trig_isr()	Frame reception interrupt handler.
EtherErr()	Ether error retrieval operation.
FlashInit()	Initialize flash API.
load_prm()	Load communication parameter (MAC address) from data flash.
upd_param()	Update communication parameter (MAC address) to data flash.
ers_param()	Erase communication parameter (MAC address) from data flash.
file_crt_dir()	Create received data directories.
file_read()	File reading operation.
file_write()	File writing operation.
file_stop()	File finalizing and operation result save operation.
file_err()	File error retrieval operation.
mtu3_dev_start()	Start MTU3 (The module-stop state is canceled).
mtu3_port_init()	Initial setting of the MTU3 ports.
mtu3_dev_stop()	Stop MTU3 (Transition to the module-stop state is made).
init_timer()	Initial setting of the MTU3 registers related to the measurement.
start_eval()	Start measurement of a communication interval.
stop_eval()	Stop measurement of a communication interval.
end_timer()	Finalizing of the MTU3 registers related to the measurement.
get_eval_cycle()	Get cycles of a communication interval.

3.2 Environment and Execution

Execution of this example needs two RX64M/71M RSK boards[1], an Ethernet cable, and a USB memory.

The outline of the execution sequence is following.

- Write the switch project execution code to one of the RX64M/71M RSK board (hereafter RSK board1). And then, write the test project execution code to the other RX64M/71M RSK board (hereafter RSK board2).
- Connect RSK board1 to RSK board2 by the Ethernet cable. The one edge and the other edge of the Ethernet cables are connected to identical channel (CH0 to CH0, CH1 to CH1)). Test frames are transferred between RSK board1 and RSK board2 via the Ethernet cable. The internal timer MTU3 of the RSK board2 measures the frame transverse time between two boards. The Data flash of the RSK board2 stores the MAC address to Block0 (Block1) when the direction of transfer is from CH0 to CH1 (from CH1 to CH0).
- Insert the USB memory to the USB port in the RSK board2. The USB memory stores the test and reception data to which transferred as the frames and the result data to indicate the performance of the frame transfer.
- The test data sample is prepared in the test project and it is stored the demo_src/usb_memory_sample folder. You can use it copying it to root layer of the USB memory.
- Power on the RSK board1 and the RSK board2.
- When the RSK board1 finishes Ethernet driver initialization and open process, the user LED composed of LED0, LED1, LED2 and LED3 shows the all-on pattern (LED0: ON, LED1: ON, LED2: ON, LED3: ON). The MAC addresses[2] assigned “74-90-50-00-79-34” for CH0 and “74-90-50-00-79-35” for CH1 in the source file in advance.
- When the RSK board2 finishes Ethernet, MTU3, Flash and USB Host driver initialization and open process, the user LED composed of LED0, LED1, LED2 and LED3 shows the “0x1” pattern (LED0: ON, LED1: OFF, LED2: OFF, LED3: OFF). In this initialization process, the MAC addresses programmed in the data flash are updated if appropriate values[3] were not set in advance.
- Push the SW1 switch of the RSK board1 and starts the switch project operation described in the Section 3.3.
- The user LED shows the times of pushing the SW1 switch. The times indicate setting pattern of the simple switch function. As for the setting pattern, please refer to the Section 3.3.
- If any error detected during the operation, the user LED shows the odd pattern (LED0: OFF, LED1: ON, LED2: OFF, LED3: ON).
- Push the SW1 switch of the RSK board2 and starts the test project operation described in the Section 3.3.
- When the test project operation finished without any error, the user LED shows the all-on pattern (LED0: ON, LED1: ON, LED2: ON, LED3: ON).
- If any error detected during the Ether communication, the user LED shows the “0x6” pattern (LED0: OFF, LED1: ON, LED2: ON, LED3: OFF).
- If any error detected during the USB communication or file access, the user LED shows the odd pattern (LED0: OFF, LED1: ON, LED2: OFF, LED3: ON).
- If any error detected during the erasing MAC address from the data flash in case of “FORCE_ERASE_PRM” option is defined, the user LED shows the “0xE” pattern (LED0: OFF, LED1: ON, LED2: ON, LED3: ON).

[1] Product name is a Renesas Starter Kit+ for RX64M [8] or a Renesas Starter Kit+ for RX71M [9].

[2] Please change this value when users applied to this sample their own system.

[3] Judging from the Renesas vendor ID (=74-90-50).

Figure 3.1 shows one of the environments during this example.

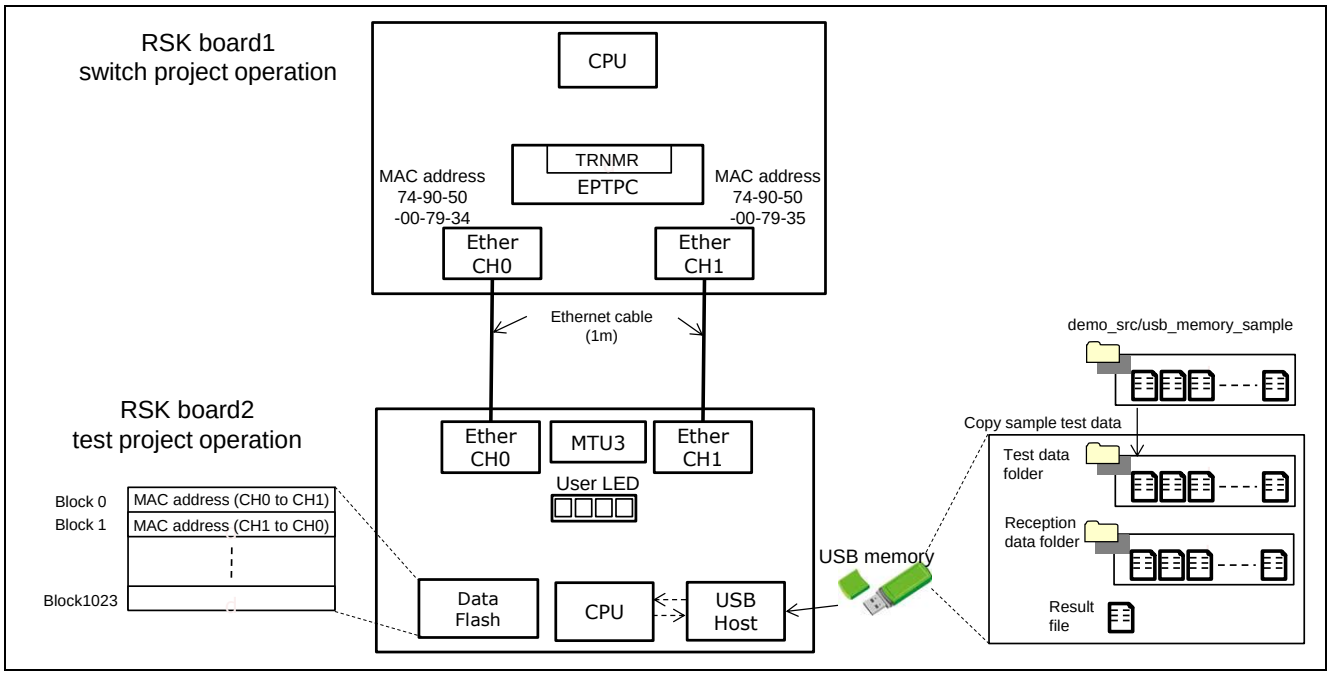

Figure 3.1 Environment

3.3 Operation Sequence

In this section, explain the operation sequence in this example when the direction of frame transfer is from CH0 to CH1 and the total test times are 201 specified to “NUM_TEST”.

Figure 3.2 shows the USB memory contents. There are TEST folder, RCV folder, and RESULT.txt file in the USB memory. The TEST folder stores the test data in the files from “SRC_0.txt” to “SRC_200.txt”. The RCV folder stores the reception data in the files from “DST_0.txt” to “DST_200.txt”. The test data and reception data mean the part of transmitting and receiving frame respectively. The contents of test data and reception one is identical if transfer operation finished without error. Figure 3.2 also shows the contents of SRC_2.txt and DST_2.txt as example. The RESULT.txt stores the number of the total test times, the frame transfer operation intervals and transferred frame sizes as the performance sample data.

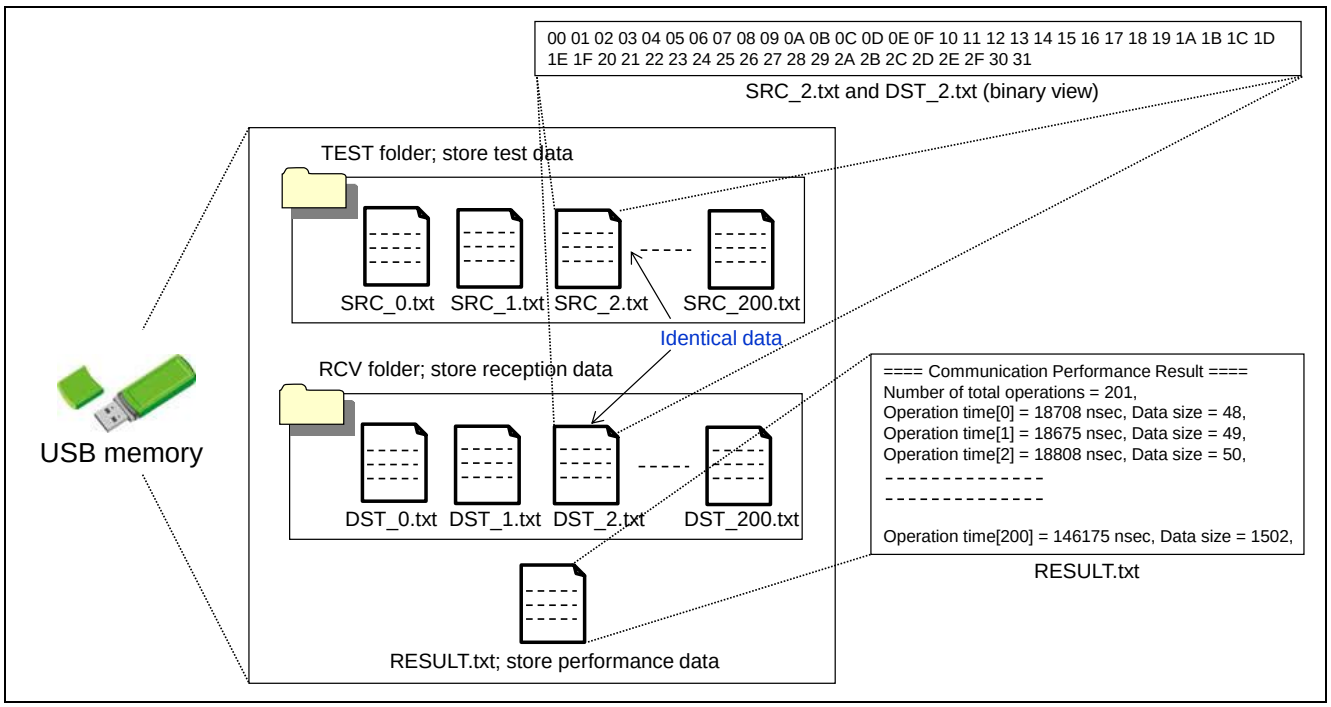

Figure 3.2 USB memory Contents

Figure 3.3 explains the test frame format in relation to the Mac addresses written in the data flash and the test data refer to the SRC_2.txt when the direction of frame transfer is from CH0 to CH1. The port0 MAC address of the RSK board2 is written beginning 6 byte in the Block0 whose address is “0x0010_0000 – 0x0010_0005” and the port1 MAC address of the RSK board2 is written next 6 byte in the Block0 whose address is “0x0010_0006 – 0x0010_000B”.

The destination MAC address is set the port1 MAC address loaded from the data flash. The source MAC address is set the port0 MAC address loaded from the data flash. After the source MAC address filed, that is start from type filed, the test data in the SRC_2.txt is concatenated. The total length of the frame becomes 62 byte (12 byte plus 50 byte).

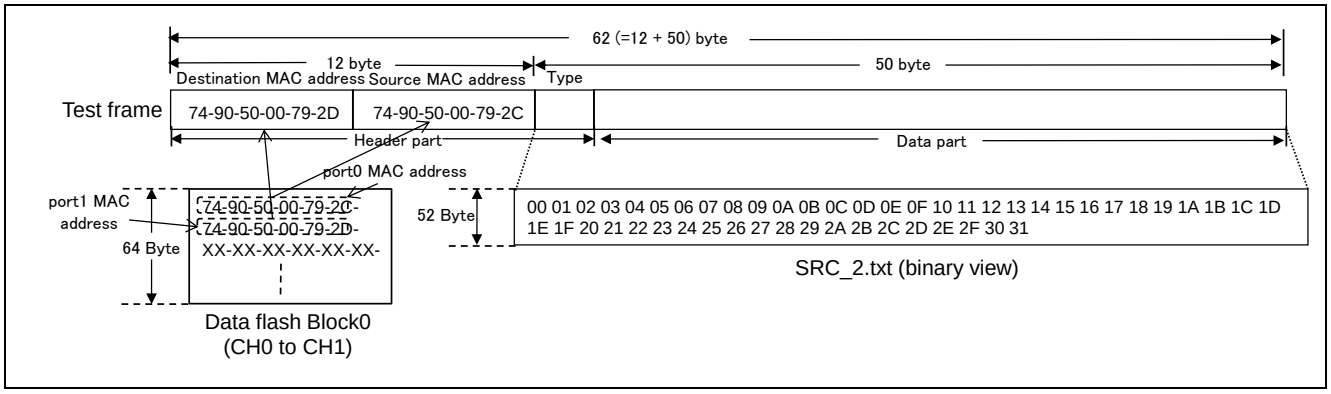

Figure 3.3 Frame format

Figure 3.4 shows the sequence over view in this example.

- RSK board1

(1) Set simple switch function.

- RSK board2

(2) Read the test data transferred as the frame from the USB memory via USB Host.

(3) Load MAC addresses from the data flash.

(4) Create the frame showed by the Figure 3.3.

(5) Transmit the frame from port0.

(6) Receive the relayed frame[1].

(7) Measure the frame transfer interval using the MTU3.

(8) Write the reception data transferred as the frame to the USB memory.

(9) Repeat the specified times of the USB read, frame transfer with interval measurement, and USB write. Thereafter, write the result of the frame transfer intervals to the USB memory.

(10) The MAC addresses in the data flash are erased if "FORCE_ERASE_PRM" option is defined.

[1] Only relayed frame by the simple switch of the RSK board2.

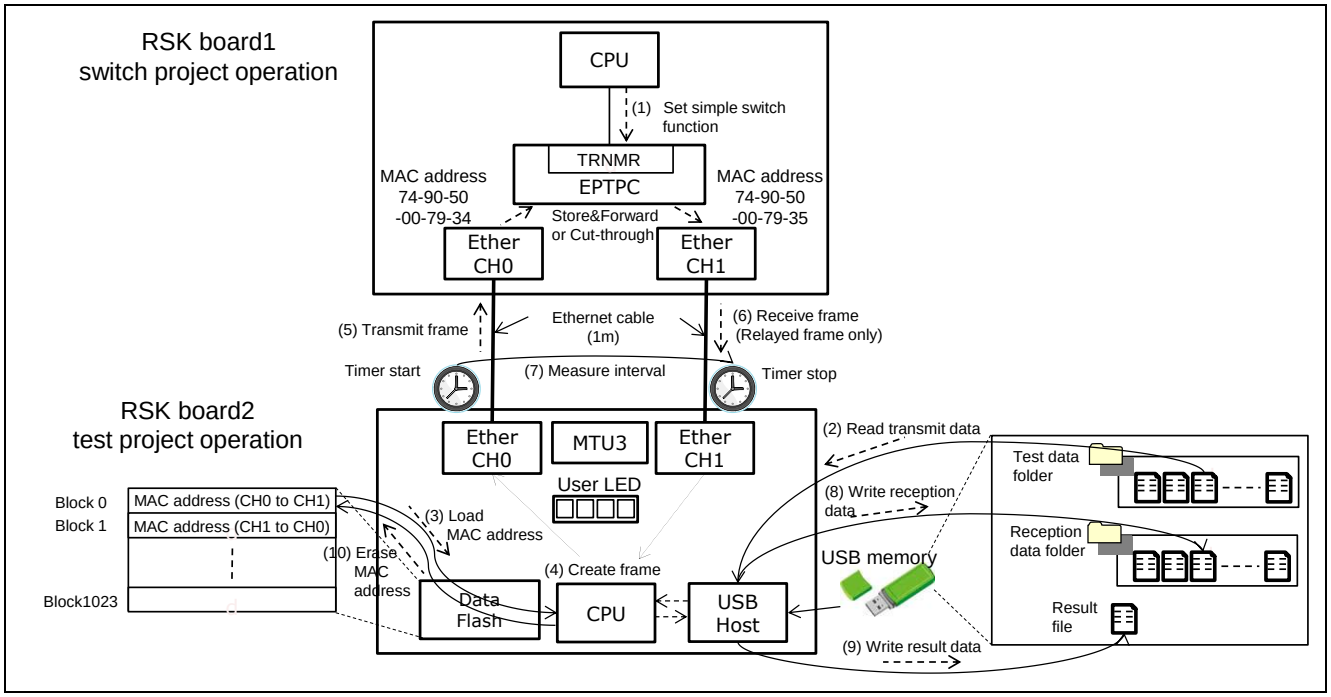

Figure 3.4 Sequence overview

3.4 Simple Switch Setting

Figure 3.5 shows the module structure and register specification related to the simple switch function. The simple switch function is implemented in the Packet Relation Control (PRC-TC) part of the Ethernet peripheral module. Store & forward or cut-throw transfer method is selectable. If cut-throw method is applied to the daisy chain topology which is common industrial network, it can reduce the inter ports transfer delay. It is also possible to use as two independent networks by the network isolating function.

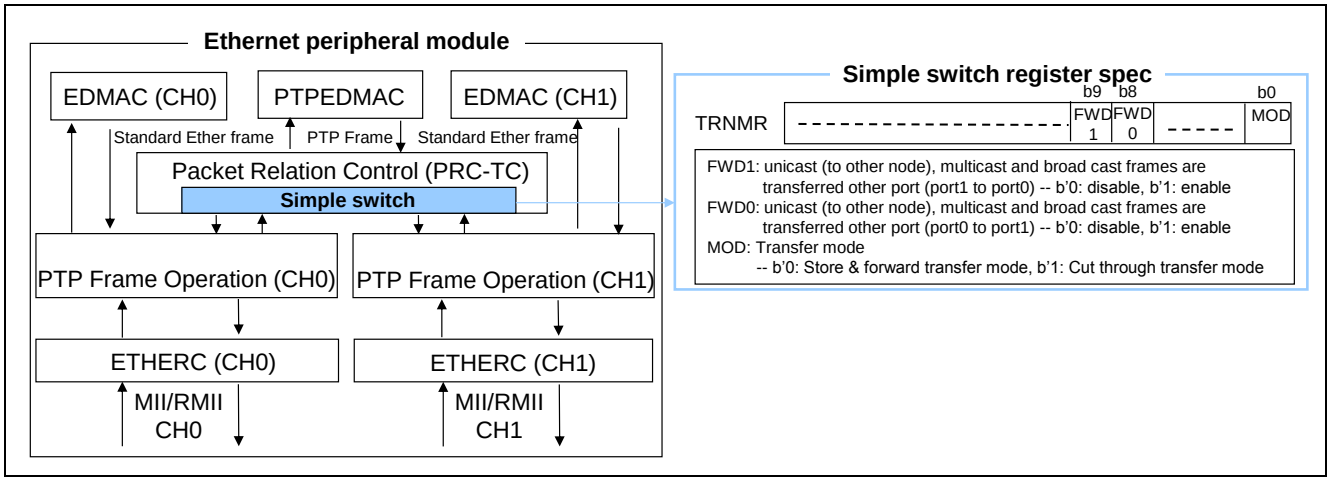

Figure 3.5 Module structure and register spec

The simple switch setting register is set by the times of pushing the SW1 switch of the RSK board1. Table 3.3 shows the simple switch setting and frame transfer result. 2nd row indicates the LED pattern depend on the SW1 pushing times after completed of Ethernet initialize sequence. 3rd row indicates TRNMR register value whose setting decides the simple switch function behavior. 8th row shows the expectation number of relayed frames if total number of the test frames is 201 specified to "NUM_TEST". In general, frame transfer performance applied to cut-through mode is faster than store & forward mode.

Table 3.3 Simple switch setting and frame transfer result

No.	LED pattern	TRNMR	FWD1 (b9)	FWD0 (b8)	MOD (b0)	Test frame Direction	Num of Frames	Comment
1	OFF-OFF-OFF-OFF	00-00-03-01	1:Enable CH1 to CH0	1:Enable CH0 to CH1	1:Cut-through	CH0 to CH1	201	Faster than No.3
2						CH1 to CH0	201	Faster than No.4
3	OFF-OFF-OFF-ON	00-00-03-00			0:Store & forward	CH0 to CH1	201	
4						CH1 to CH0	201	
5	OFF-OFF-ON-OFF	00-00-02-01		0:Disable CH0 to CH1	1:Cut-through	CH0 to CH1	0	Not relayed
6						CH1 to CH0	201	Faster than No.8
7	OFF-OFF-ON-ON	00-00-02-00			0:Store & forward	CH0 to CH1	0	Not relayed
8						CH1 to CH0	201	
9	OFF-ON-OFF-OFF	00-00-01-01	0:Disable CH1 to CH0	1:Enable CH0 to CH1	1:Cut-through	CH0 to CH1	201	Faster than No.11
10						CH1 to CH0	0	Not relayed
11	OFF-ON-OFF-ON	00-00-01-00			0:Store & forward	CH0 to CH1	201	
12						CH1 to CH0	0	Not relayed
13	OFF-ON-ON-OFF	00-00-00-01		0:Disable CH0 to CH1	1:Cut-through	CH0 to CH1	0	Not relayed
14						CH1 to CH0	0	Not relayed
15	OFF-ON-ON-ON	00-00-00-00			0:Store & forward	CH0 to CH1	0	Not relayed
16						CH1 to CH0	0	Not relayed

3.5 Software Operation Flow

In this section, describes the software operation flow of this sample.

3.5.1 switch project

Figure 3.6 shows the operation flow. The switch project set the Ethernet peripheral modules to do Ethernet communication. Thereafter, it set simple switch function and counts the number of frame reception using the frame received interrupt handler.

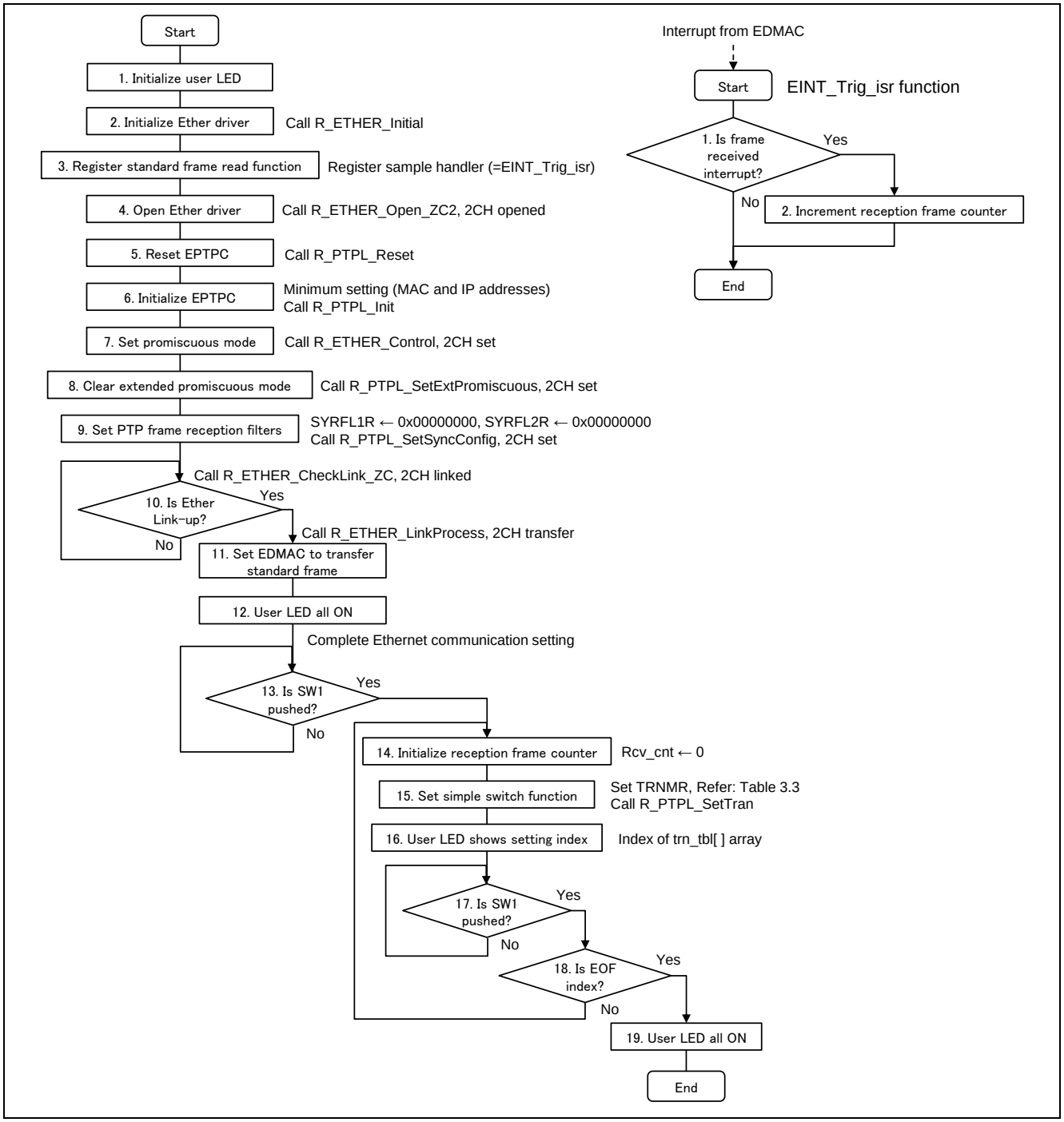

Figure 3.6 operation flow of the switch project

3.5.2 test project

Figure 3.7 shows the initialize operation and infinite loop. This sample task (Sample_Task function) is called via USB application task showed by Figure 3.8 within the infinite loop. Figure 3.9 shows the each task structure managed by the application state.

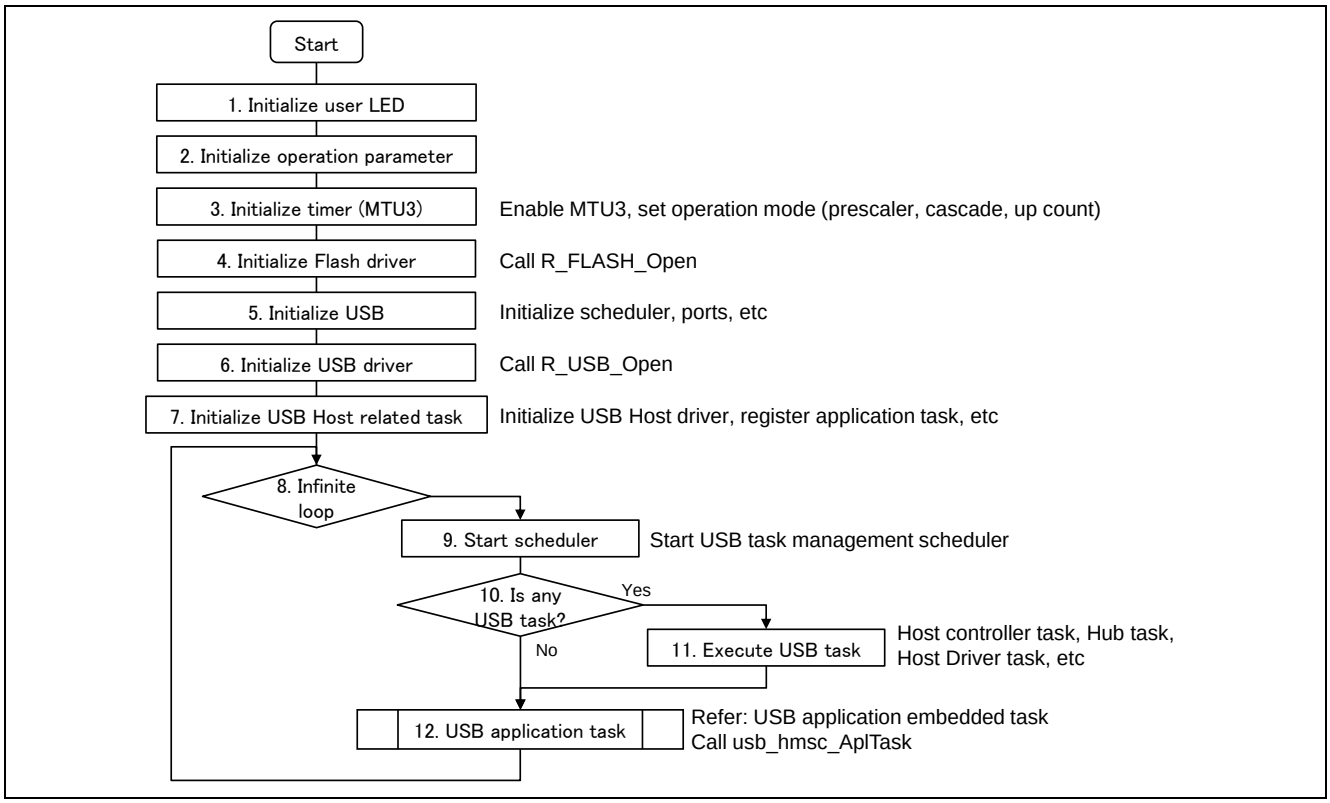

Figure 3.7 Initial operation

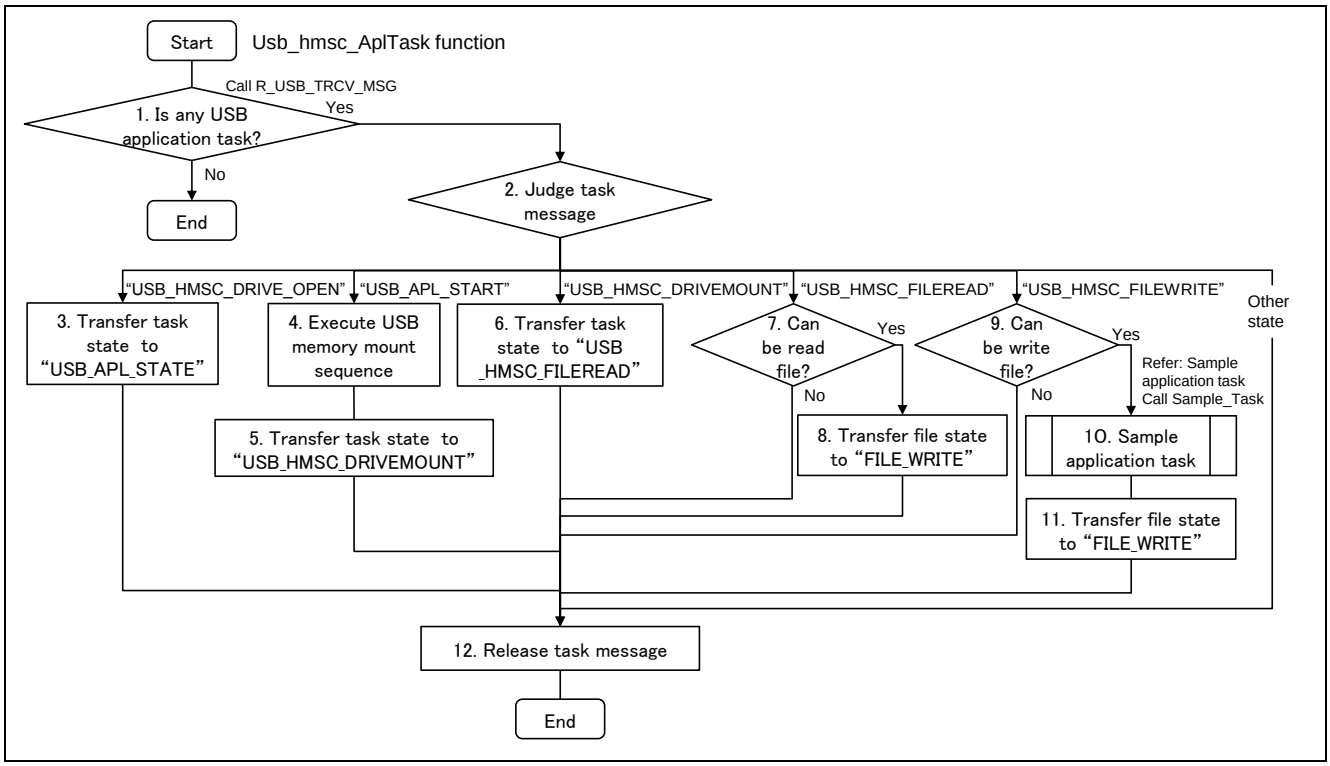

Figure 3.8 USB application embedded task

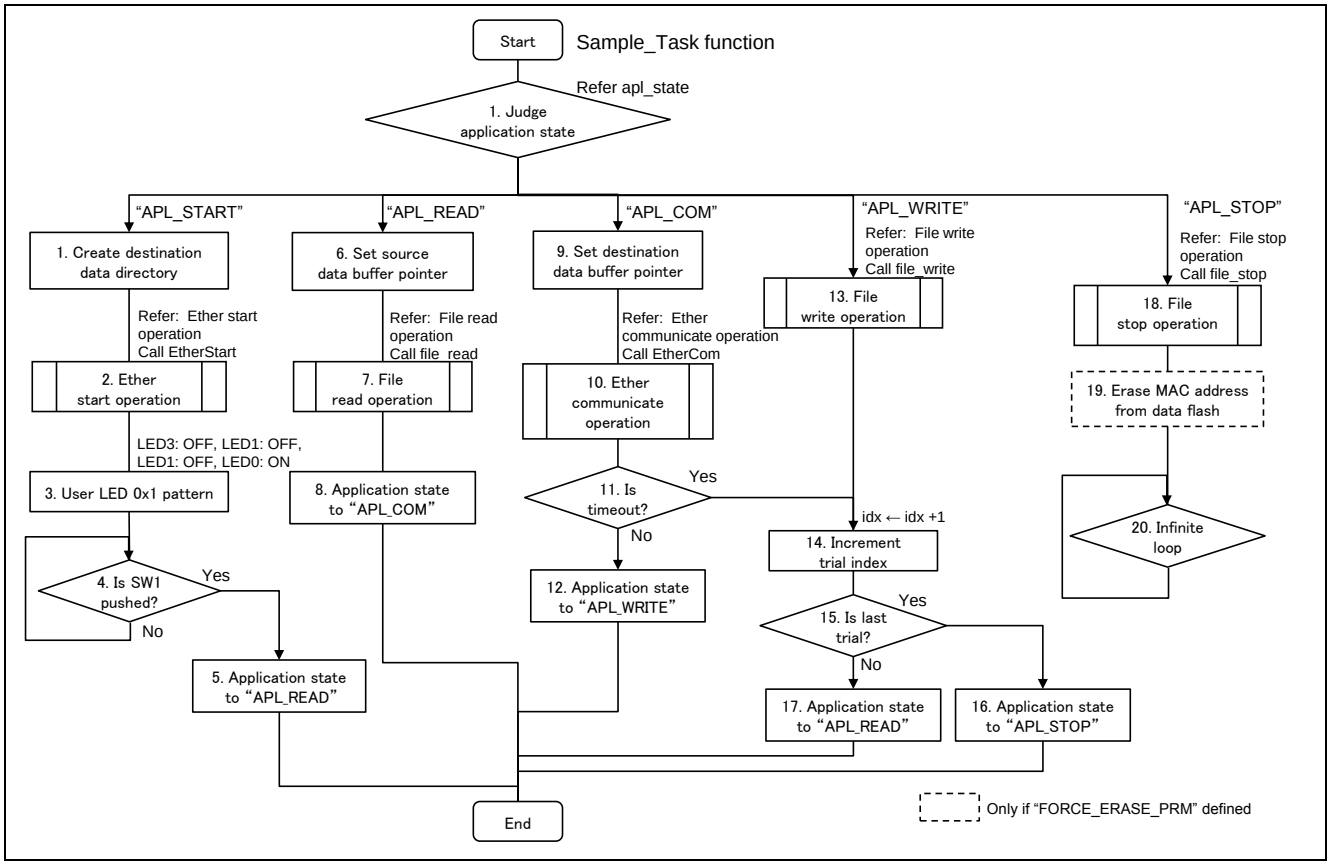

Figure 3.9 Sample application task

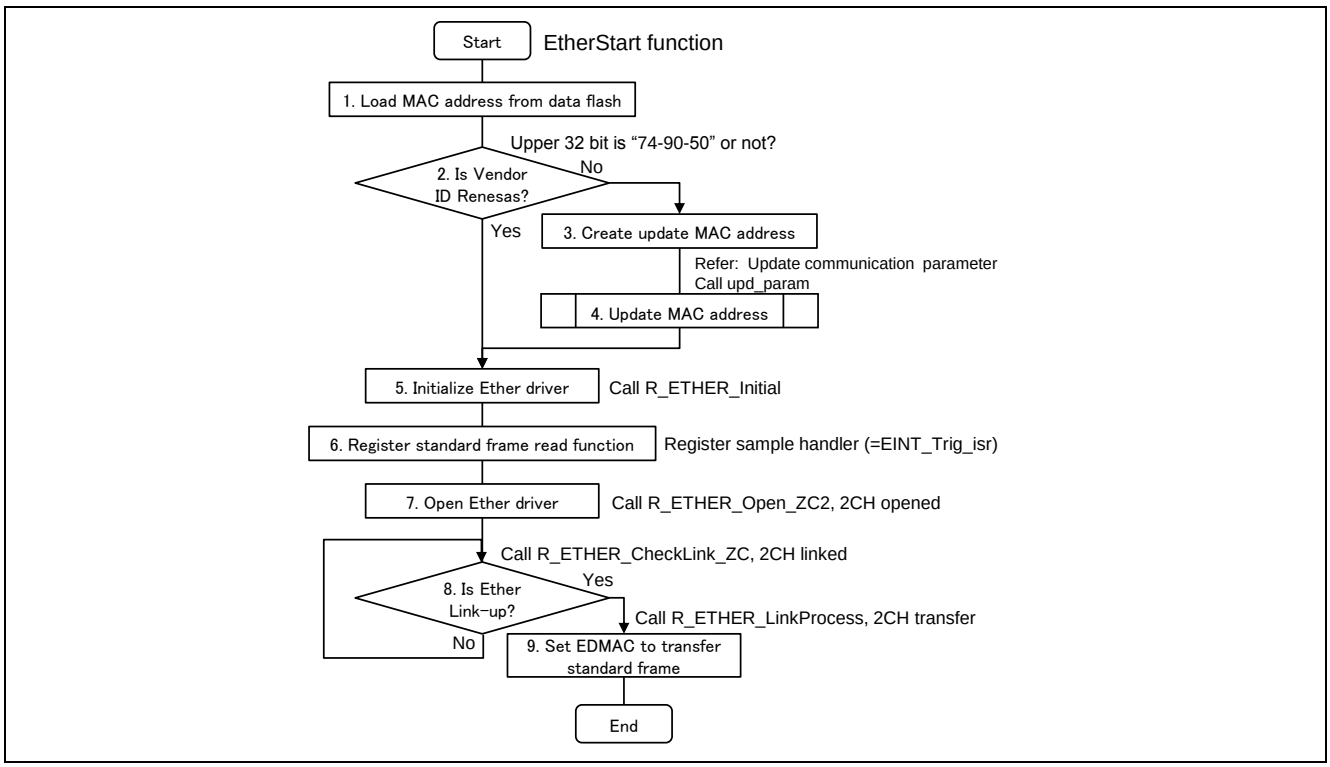

Figure 3.10 Ether start operation

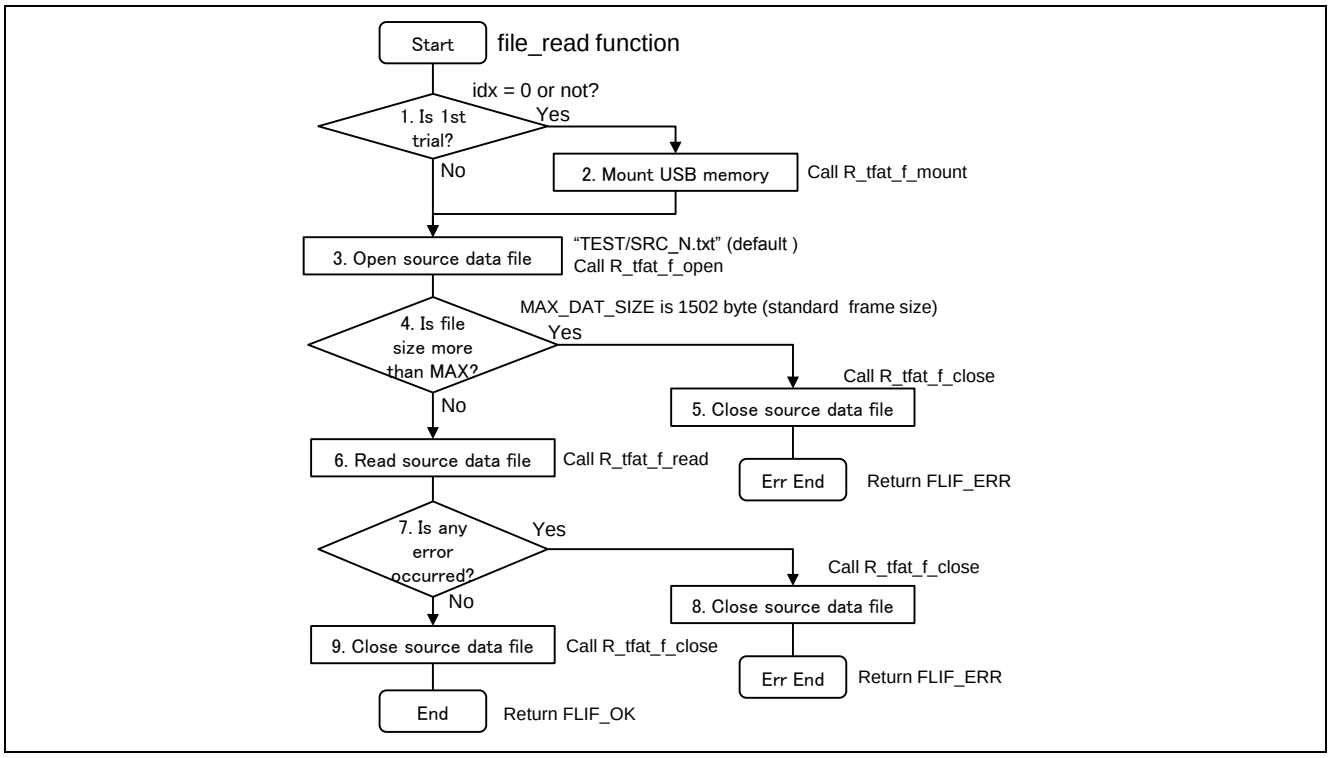

Figure 3.11 File read operation

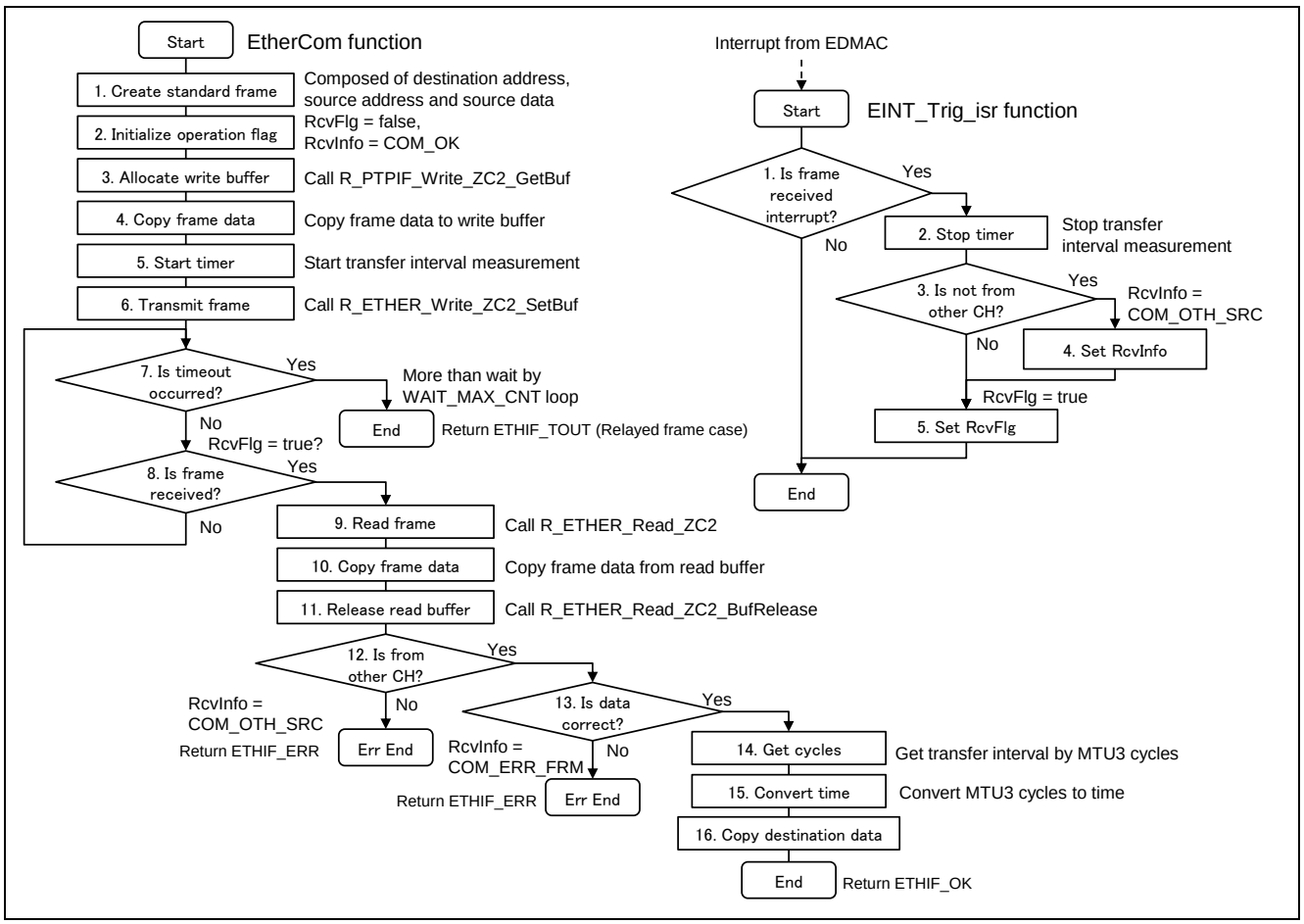

Figure 3.12 Ether communicate operation

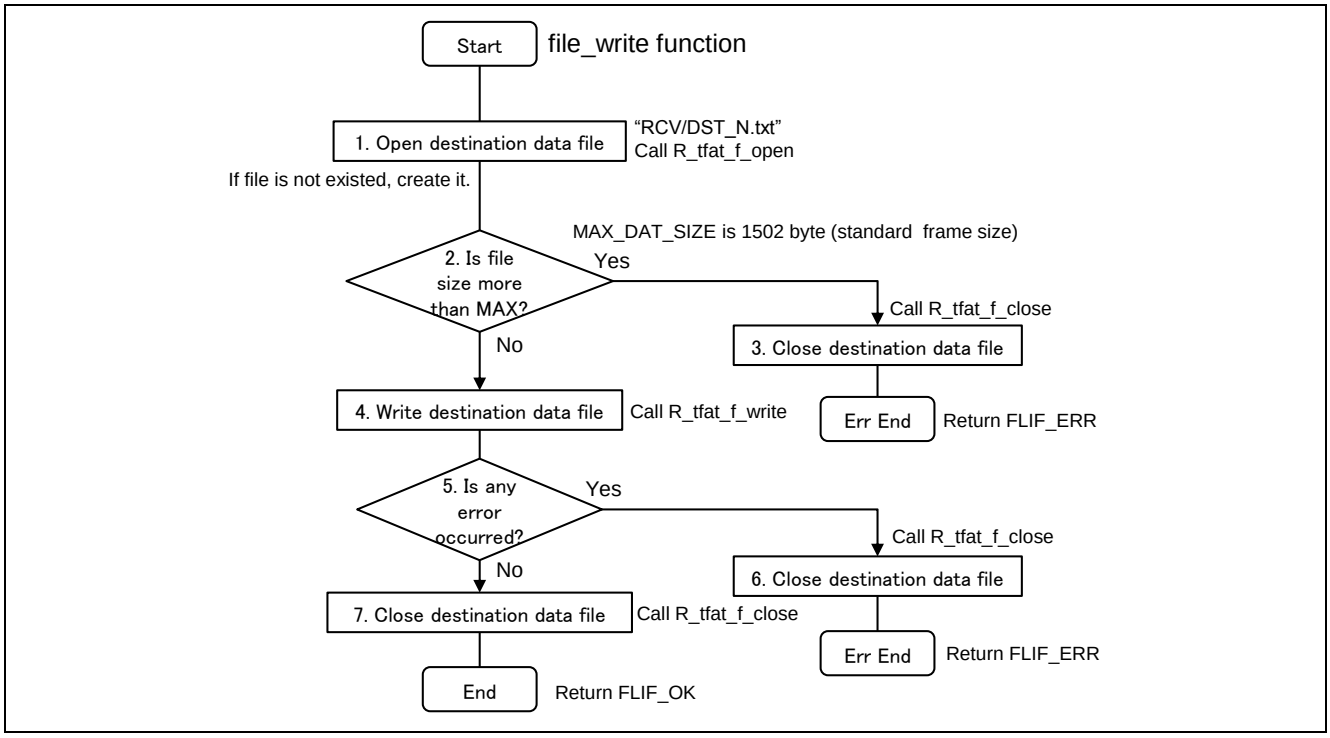

Figure 3.13 File write operation

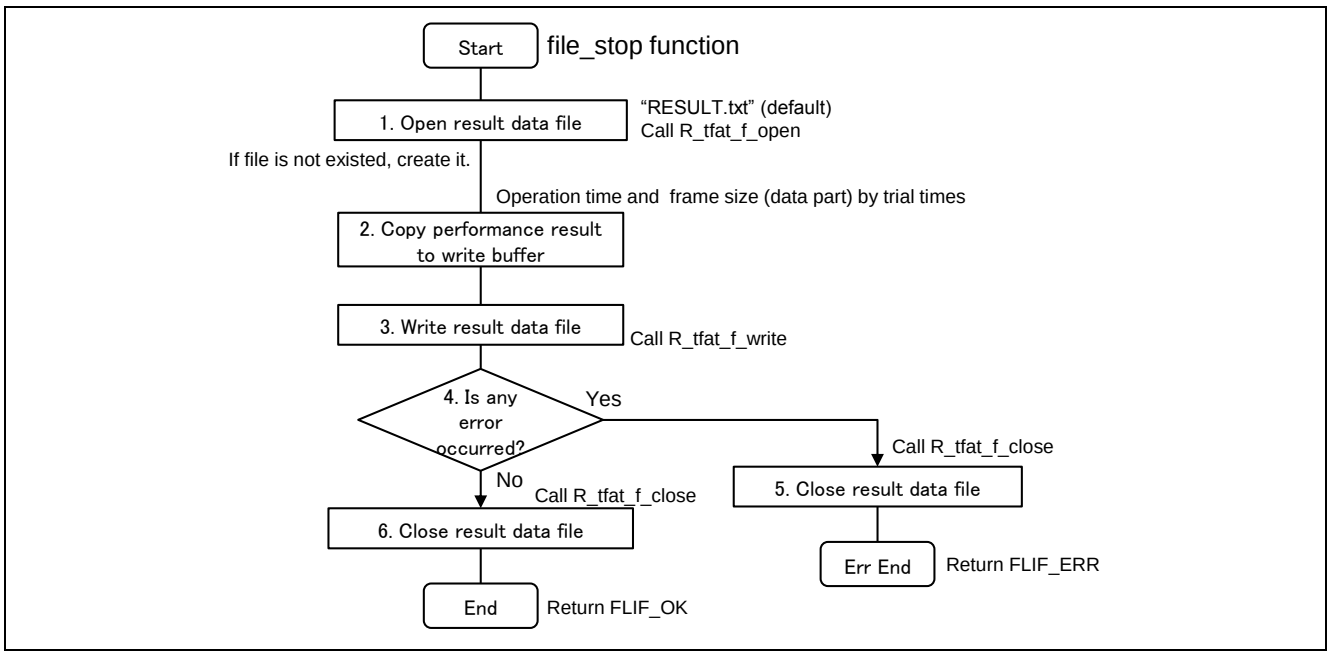

Figure 3.14 File stop operation

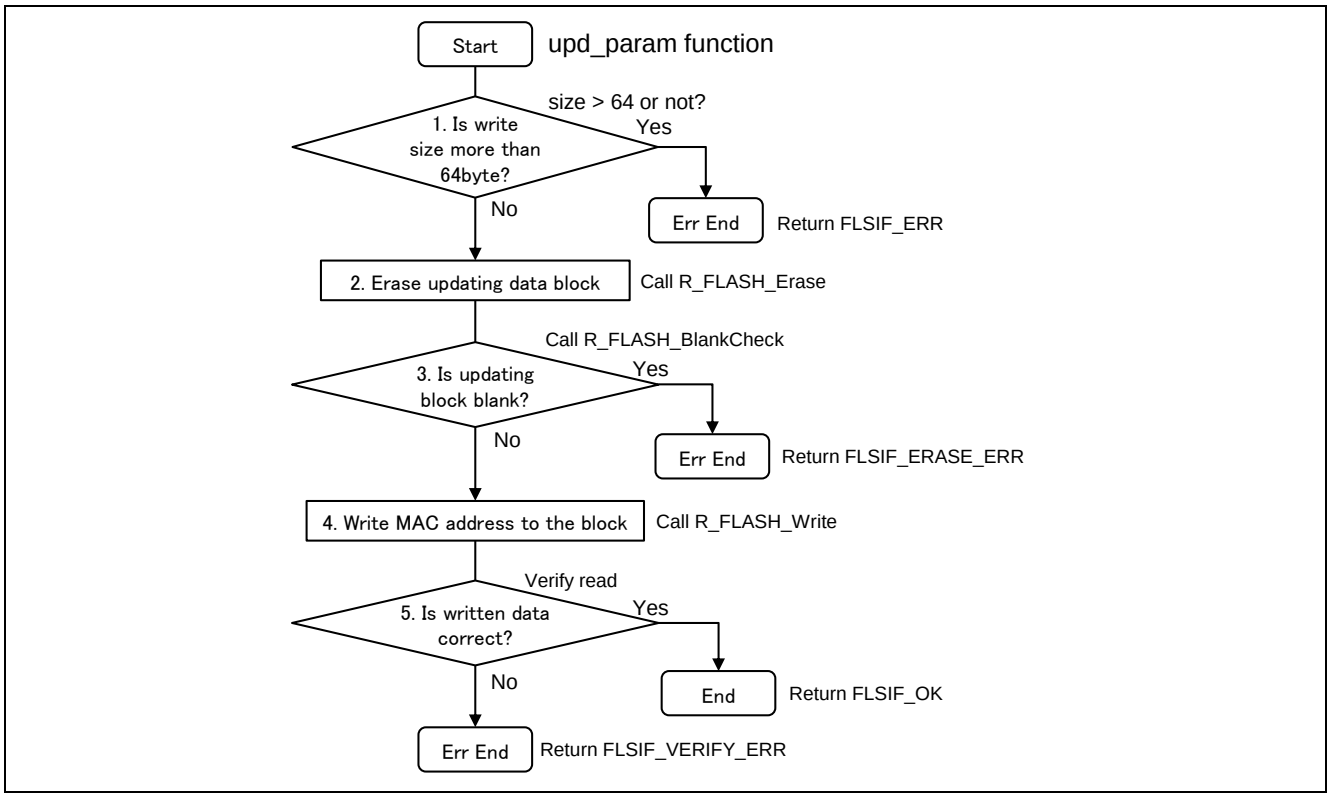

Figure 3.15 Update communication parameter

3.6 Board Setting

There are four jumpers changing from the default setting of the RX64M/71M RSK board to execute this example. The Ether PHY access channel is set consistent with the software configuration. The USB access setting[1] has to be changed from the board default setting. When the product name of the RX64M/71M RSK board is R0K50564MC001BR or R0K5RX71MC010BR, Figure 3.16 indicates their changing. And when the product name of the RX71M RSK board is R0K50571MC000BR, Figure 3.17 indicates their changing.

[1] USB setting only needs RSK board2 operated by the test project. (not need RSK board1 operated by the switch project)

- Ether PHY access setting

Jumper	LINK_CH = 1 (Default setting)	LINK_CH = 0	Functional use
J3	2-3	1-2	ETHERC ET0MDIO or ET1MDIO
J4	2-3	1-2	ETHERC ET0MDC or ET1MDC

- USB access setting

Jumper	Board default setting	This example	Functional use
J2	2-3	1-2	USB Enables Host Mode
J6	1-2	2-3	USB USB0VBUSEN

Figure 3.16 Jumper setting

- Ether PHY access setting

Jumper	LINK_CH = 1 (Default setting)	LINK_CH = 0	Functional use
J13	2-3	1-2	ETHERC ET0MDIO or ET1MDIO
J9	2-3	1-2	ETHERC ET0MDC or ET1MDC

- USB access setting

Jumper	Board default setting	This example	Functional use
J1	2-3	1-2	USB Enables Host Mode
J3	1-2	2-3	USB USB0VBUSEN

Figure 3.17 Jumper setting

4. Reference Documents

User's Manual: Hardware

RX64M Group User's Manual: Hardware Rev.1.00 (R01UH0377EJ)
RX71M Group User's Manual: Hardware Rev.1.00 (R01UH0493EJ)
The latest version can be downloaded from the Renesas Electronics website.

User's Manual: Software

RX Family RXv2 Instruction Set Architecture User's Manual: Hardware Rev.1.00 (R01US0071EJ)
The latest version can be downloaded from the Renesas Electronics website.

Technical Update/Technical News

The latest information 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

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	Jul 24, 2015	—	First edition issued.
1.10	Mar 31, 2016	—	Applied PTP light driver Rev.1.10 and Ethernet driver Rev.1.10.
1.11	Nov 11, 2016	—	Applied PTP light driver Rev.1.11 and Ethernet driver Rev.1.12.

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

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

1. Handling of Unused Pins

   Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.

   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of 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 a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of Microprocessing unit or Microcontroller unit products in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

RENESAS

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.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, 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 1207, 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 India Pvt. Ltd.
No.777C, 100 Feet Road, HALⅡ Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 5.0