RX Family

LPT Module Using Firmware Integration Technology

Abstract

This application note describes the LPT module using firmware integration technology (FIT). This module uses the low-power timer (LPT) to produce a signal to exit software standby mode. The MCU can periodically exit software standby mode using this module with software standby mode and ELC. Supported device can generate PWM waveform.

Hereinafter this module is referred to as the “LPT FIT module”.

Target Devices

The following is a list of devices that are currently supported by this API:

- RX113 Group
- RX130 Group
- RX140 Group
- RX231, RX230 Groups
- RX23W Group

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

Target Compilers

- Renesas Electronics C/C++ Compiler Package for RX Family (V2.05.00 or higher)
- GCC for Renesas RX
- IAR C/C++ Compiler for Renesas RX

For details of the confirmed operation contents of each compiler, refer to “7.1 Operation Confirmation Environment".

Related Documents

For additional information associated with this document, refer to the following application notes.

- Firmware Integration Technology User’s Manual (R01AN1833)

Contents

1. Specifications

The LPT FIT module supports the LPT which is the RX Family peripheral function and produces a signal to exit software standby mode.

The MCU can periodically exit software standby mode using this module with software standby mode and the event link controller (ELC). Supported device can generate PWM waveform.

1.1 LPT FIT Module

The LPT FIT module can be used by being implemented in a project as an API. See section 2.11 Adding the FIT Module to Your Project for details on methods to implement this FIT module into a project.

1.2 Overview of the LPT FIT Module

Settings for operating the LPT are configured by calling the R_LPT_Open function in this module. Also the LPT cycle is specified by calling R_LPT_Open function.

The compare match cycle is specified by calling R_LPT_InitChan function. Calling the R_LPT_InitChan function the selected compare match channel is enabled and the compare match cycle is set.

If the compare match cycle needs to be changed after the LPT has started operating, call the R_LPT_SetCMT function.

When starting LPT count, call the R_LPT_Control function using the LPT_CMD_START command.

When stopping LPT count, call the R_LPT_Control function using the LPT_CMD_STOP command.

When resetting LPT count, call the R_LPT_Control function using the LPT_CMD_COUNT_RESET command.

When starting PWM output, call the R_LPT_Control function using the LPT_CMD_PWM_START command.

When stopping PWM output, call the R_LPT_Control function using the LPT_CMD_PWM_STOP command.

Transition from software standby mode to the ELC operation enable state is triggered by the compare match 0 occurrence.

When compare match 1 occur, the interrupt request, the event output to ELC, the transition to snooze mode request, and the DMAC start request.

If the LPT module needs to be closed after the LPT has opend, call the R_LPT_Close.

When using this module, the LPT must not be controlled by any other modules.

1.3 API Overview

Table 1.1 lists the API functions included in the module.

Table 1.1 API Functions

Function	Description
R_LPT_Open	Initialize the LPT module and set LPT cycle.
R_LPT_InitChan	Enable the selected channel and set the compare match cycle.
R_LPT_SetCMT	Set the compare match cycle for the selected channel.
R_LPT_InitPWM	Set the PWM output configuration of the selected channel.
R_LPT_Control	Controls start/stop/reset of LPT count and start/stop of PWM output.
R_LPT_FinalChan	Disable the selected channel.
R_LPT_Close	Releases the LPT module.
R_LPT_GetVersion	Returns the version of this module.

1.4 Processing Example

Figure 1.1 shows an example of processing. The following is an example of the LPT FIT module.

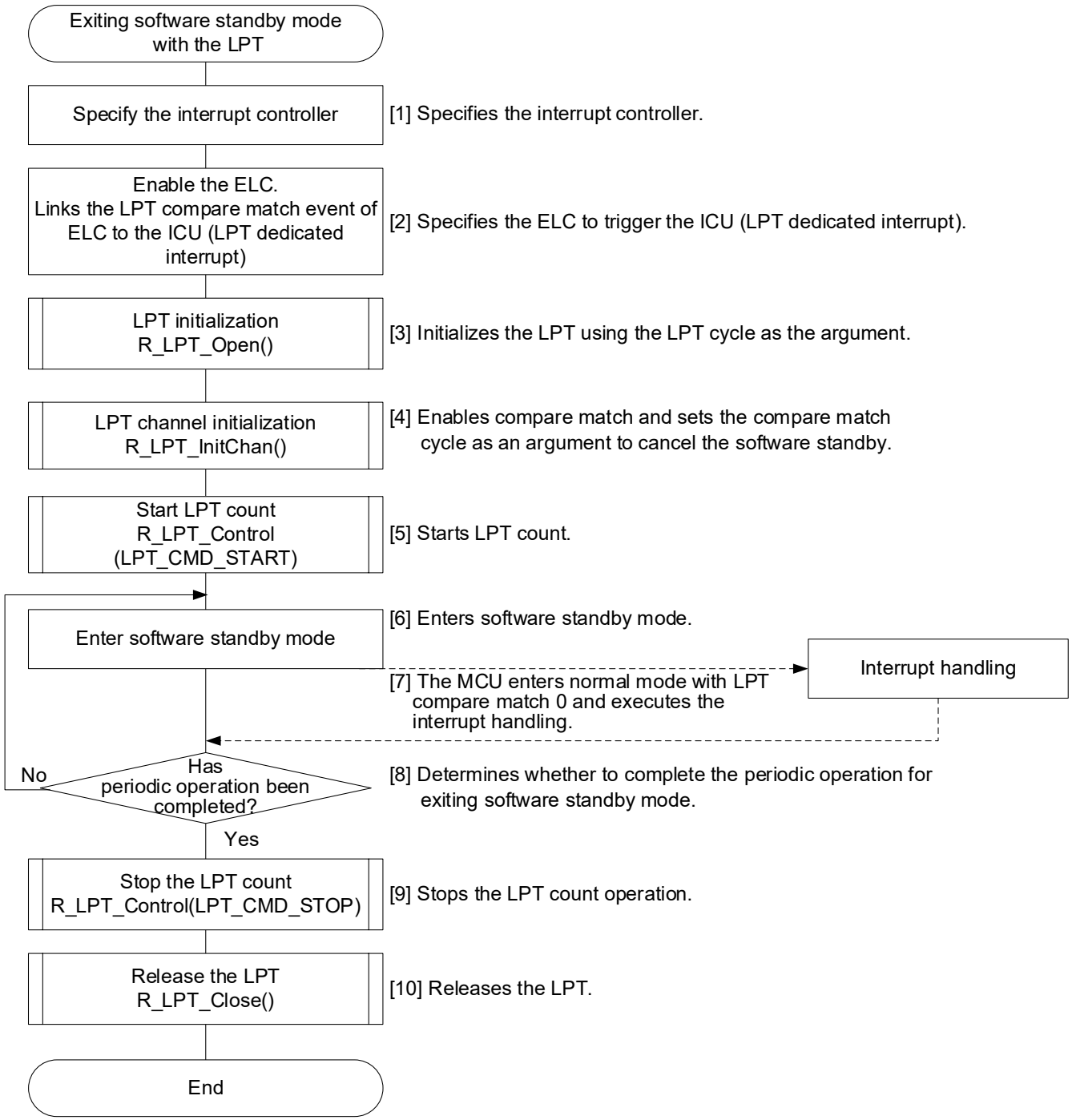


Figure 1.1 Processing Example of the LPT FIT Module

1.5 State Transition

Figure 1.2 shows the state transition diagram for this module.

The following is the state transition of the LPT FIT module as an example.

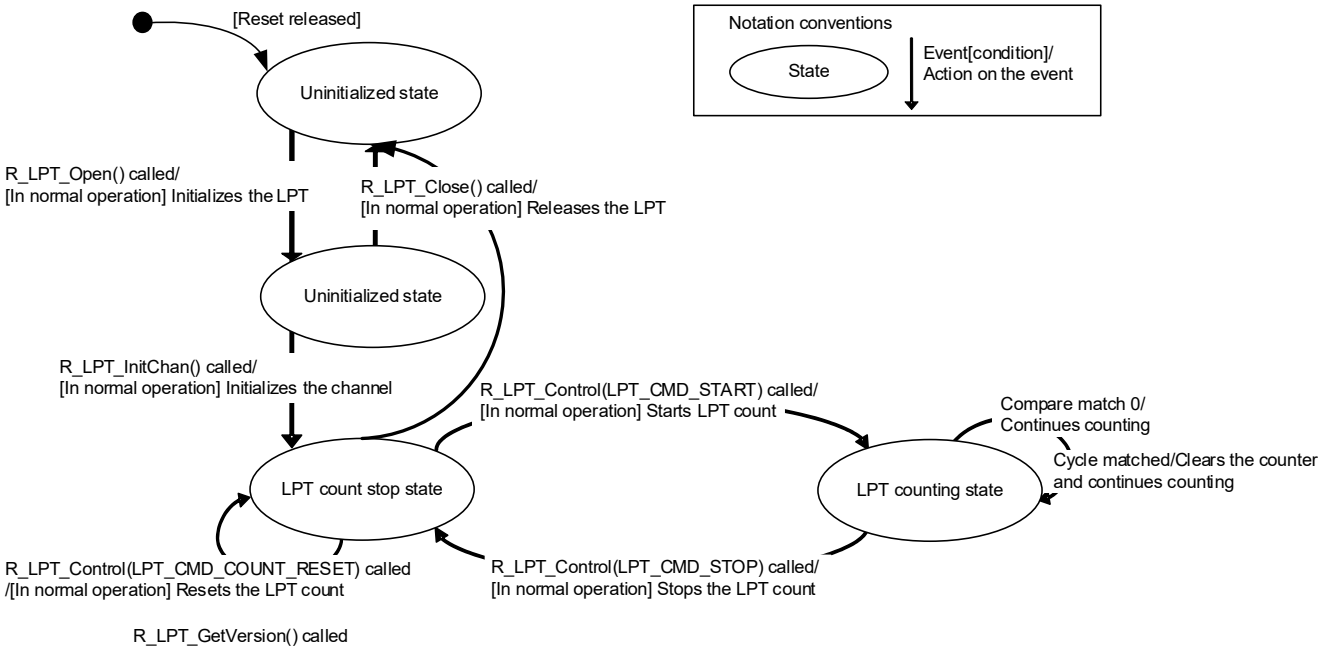


Figure 1.2 LPT FIT Module State Transition Diagram

2. API Information

This FIT module has been confirmed to operate under the following conditions.

2.1 Hardware Requirements

This driver requires your MCU support the following features:

- Low-power timer (LPT)
- Event link controller (ELC)
- Low power consumption (LPC)

2.2 Software Requirements

This driver is dependent upon the following packages:

- Renesas Board Support Package (r_bsp)

2.3 Supported Toolchains

This driver has been confirmed to work with the toolchain listed in 7.1, Operation Confirmation Environment.

2.4 Interrupt Vector

This FIT module does not use interrupt vectors.

2.5 Header Files

All API calls and their supporting interface definitions are located in r_lpt_rx_if.h.

2.6 Integer Types

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

2.7 Configuration Overview

The configuration options in this module are specified in r_lpt _rx_config.h. The option names and setting values are listed in the table below.

Configuration options in *r_lpt_rx_config.h*	
`#define LPT_CFG_PARAM_CHECKING (1)`	Selects whether to include parameter checking in the code. The parameter checking is processing to check parameters and is located in the beginning of each function. - When this is set to 0, code for parameter checking is not generated. - When this is set to 1, code for parameter checking is generated and executed. And, definition of “LPT_CFG_PARAM_CHECKING” is invalid when definition of “BSP_CFG_PARAM_CHECKING_ENABLE” is set 0. Then Parameter checking code is invalid.

`#define LPT_CFG_LPT_CLOCK_SOURCE (0)`
 	
`A default value is “BSP_CFG_PARAM_CHECKING_ENABLE” which is defined in r_bsp_config.h file.`
	Selects the clock source for the low-power timer.
	- When this is set to 0, the sub-clock oscillator is selected.
	- When this is set to 1, the IWDT-dedicated on-chip oscillator is selected.
	- When this is set to 1, the 4 divided low-speed on-chip oscillator is selected.

2.8 Code Size

The sizes of ROM, RAM and maximum stack usage associated with this module are listed below.

The ROM (code and constants) and RAM (global data) sizes are determined by the build-time configuration options described in 2.7, Configuration Overview.

The values in the table below are confirmed under the following conditions.

Module Revision: r_lpt_rx rev3.00

Compiler Version: Renesas Electronics C/C++ Compiler Package for RX Family V3.03.00

(The option of "-lang = c99" is added to the default settings of the integrated development environment.)

GCC for Renesas RX 4.8.4.201904

(The option of "-std=gnu99" is added to the default settings of the integrated development environment.)

IAR C/C++ Compiler for Renesas RX version 4.20.1

(The default settings of the integrated development environment.)

Configuration Options: Default settings

ROM, RAM and Stack Code Sizes

Device	Category	Memory Used					
		Renesas Compiler		GCC		IAR Compiler	
		With Parameter Checking	Without Parameter Checking	With Parameter Checking	Without Parameter Checking	With Parameter Checking	Without Parameter Checking
RX113 RX130 RX230 RX231 RX23W	ROM	**829 bytes**	**821 bytes**	**2,220 bytes**	**2,188 bytes**	**1,322 bytes**	**1,314 bytes**
	RAM	**0 bytes**		**0 bytes**		**0 bytes**	
	STACK	**104 bytes**	**100 bytes**	**-**		**80 bytes**	**76 bytes**
RX140	ROM	**1053 bytes**	**1038 bytes**	**1716 bytes**	**1700 bytes**	**1779 bytes**	**1757 bytes**
	RAM	**0 bytes**		**0 bytes**		**0 bytes**	
	STACK	**88 bytes**	**88 bytes**	–		**76 bytes**	**76 bytes**

2.9 Parameters

This section describes the parameter the enumeration used by the API functions in this module. The enumeration is located in r_lpt_rx_if.h as are the prototype declarations of API functions.

```
typedef enum e_lpt_ch
{
    LPT_CH0=0,     /* LPT channel 0 */
    LPT_CH1,       /* LPT channel 1 */
    LPT_NUM_CH
} lpt_ch_t;
```

```
typedef enum e_lpt_cmd
{
      LPT_CMD_START,             /* Start LPT count */
      LPT_CMD_STOP               /* Stop LPT count */
      LPT_CMD_COUNT_RESET,       /* Reset LPT count */
      LPT_CMD_PWM_START,         /* Start PWM output */
      LPT_CMD_PWM_STOP           /* Stop PWM output */
} lpt_cmd_t;
```

```
typedef struct st_lpt_pwm_cfg
{
    lpt_pwm_polarity_t output_polarity;    /* Output polarity */
    lpt_pwm_level_t output_level;          /* Output level */
}lpt_pwm_cfg_t;
```

2.10 Return Values

This section describes return values of API functions. This enumeration is located in r_lpt_rx_if.h as are the prototype declarations of API functions.

```
typedef enum /* Status codes for LPT APIs */
{
   LPT_SUCCESS:                /* Processing completed successfully. */
   LPT_ERR_LOCK_FUNC:          /* Operating. LPT has been used. */
   LPT_ERR_INVALID_ARG:        /* Argument has an invalid value. */
   LPT_ERR_CONDITION_NOT_MET: /* Condition not met. */
   LPT_ERR_INVALID_CH,         /* Channel is invalid. */
   LPT_ERR_NULL_PTR            /* Received null ptr. missing required argument */
} lpt_err_t;
```

2.11 Adding the FIT Module to Your Project

This module must be added to each project in which it is used. Renesas recommends the method using the Smart Configurator described in (1) or (3) or (5) below. However, the Smart Configurator only supports some RX devices. Please use the methods of (2) or (4) for RX devices that are not supported by the Smart Configurator.

(1) Adding the FIT module to your project using the Smart Configurator in e^2 studio
By using the Smart Configurator in e^2 studio, the FIT module is automatically added to your project. Refer to “RX Smart Configurator User’s Guide: e^2 studio (R20AN0451)” for details.

(2) Adding the FIT module to your project using the FIT Configurator in e^2 studio
By using the FIT Configurator in e^2 studio, the FIT module is automatically added to your project. Refer to “RX Family Adding Firmware Integration Technology Modules to Projects (R01AN1723)” for details.

(3) Adding the FIT module to your project using the Smart Configurator in CS+
By using the Smart Configurator Standalone version in CS+, the FIT module is automatically added to your project. Refer to “RX Smart Configurator User’s Guide: CS+ (R20AN0470)” for details.

(4) Adding the FIT module to your project in CS+
In CS+, please manually add the FIT module to your project. Refer to “RX Family Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)” for details.

(5) Adding the FIT module to your project using the Smart Configurator in IAREW
By using the Smart Configurator Standalone version, the FIT module is automatically added to your project. Refer to “RX Smart Configurator User’s Guide: IAREW (R20AN0535)” for details.

2.12 Adding FIT Modules to the IAR Project

This section describes how to add FIT modules to IAR projects.

2.12.1 Adding FIT Modules by using the Smart Configurator standalone version

In this explanation, IAR Embedded Workbench for Renesas RX 4.12.1 is used.

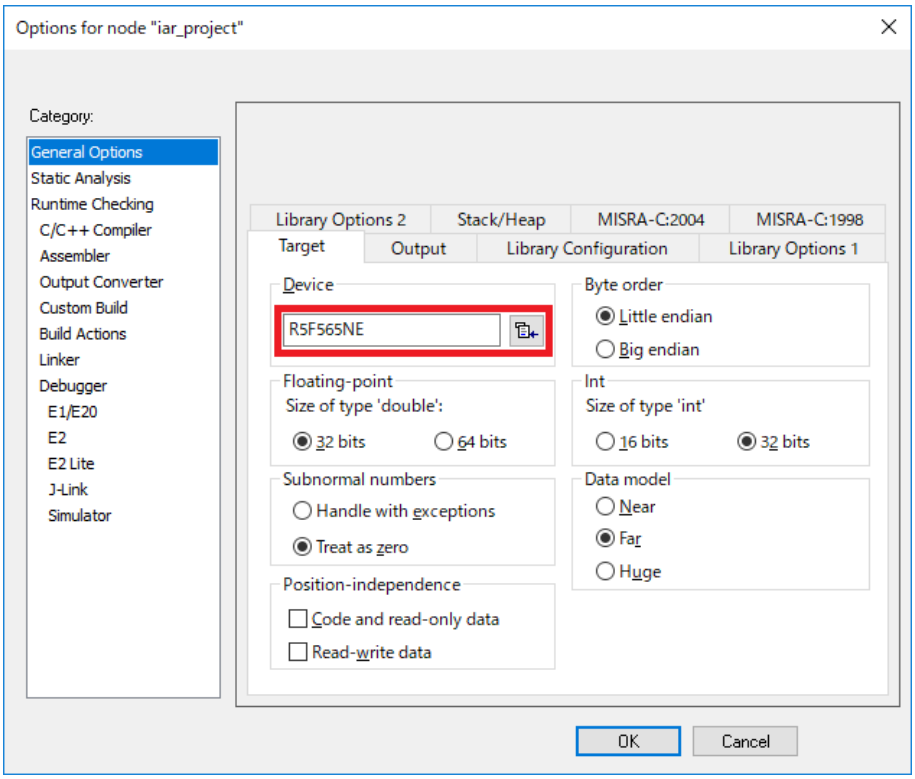
(1) Create a new project in IAREW.

(2) Adding FIT Modules to the IAR project by following the procedure in “7.1 Adding the FIT Module to YourProject”.

(3) Right-click on the project and click “Options…”.

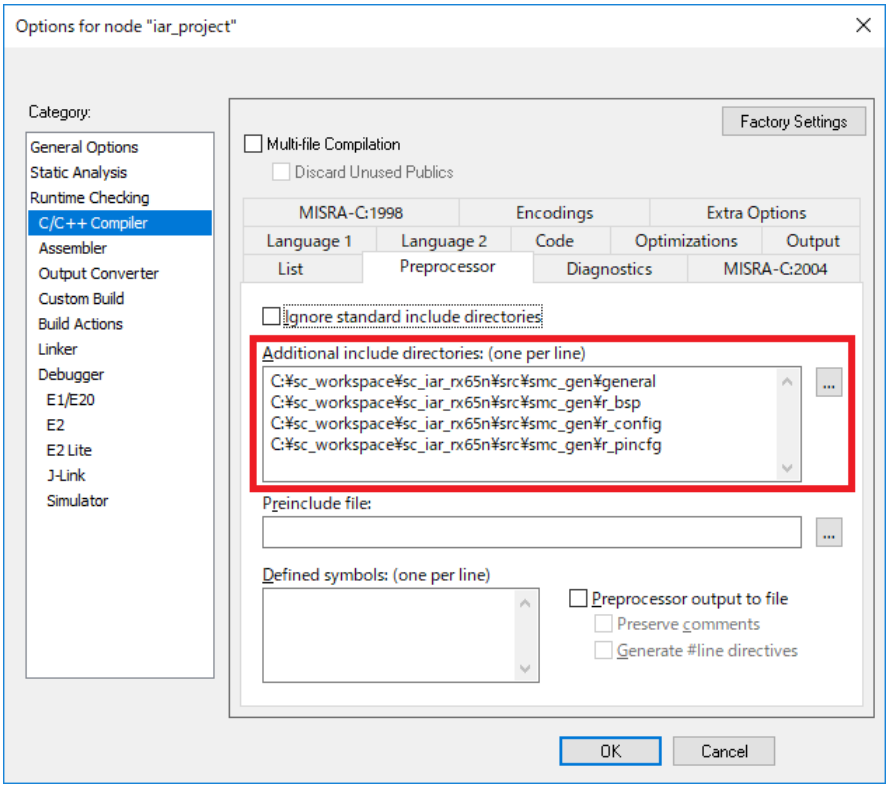
(4) Select “Target” on the General Options tab.

(5) For “Device”, select a device to use.



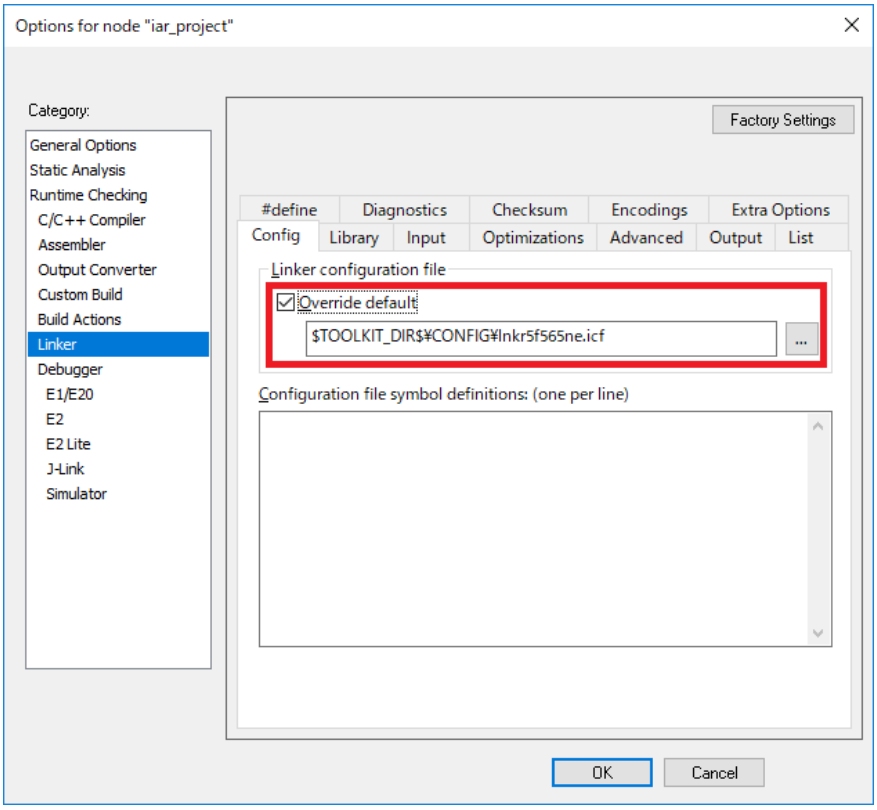
(6) Select “Preprocessor” on the C/C++ Compiler tab.

(7) Include path of the FIT modules for generated by the smart configurator standalone version is set.



(8) Select “Config” on the Linker tab.

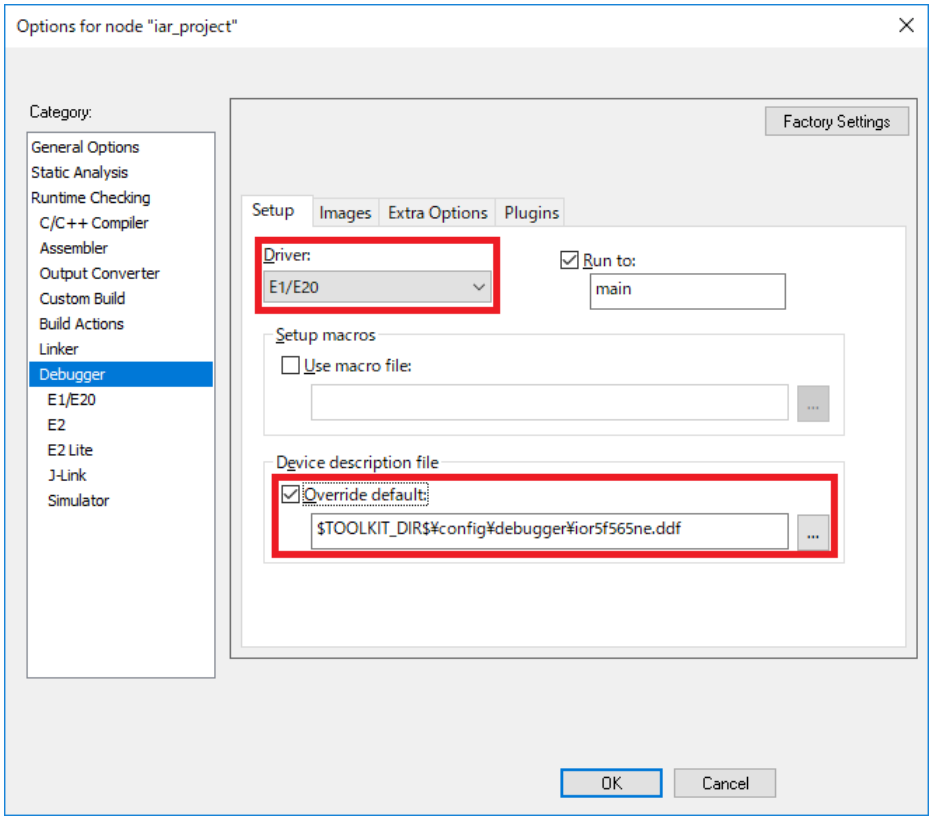
(9) For the linker configuration file, tick the “Override default” check box. Then, select “the target device.icffile”.



(10) Select “Setup” on the Debugger tab.

(11) For the driver, select “Emulator”.

(12) For the device description file, tick the “Override default” check box, and then select “the target device.dfffile”.



(13) Click “Project >> Rebuild All”.

(14) Click “E1/E20 Emulator >> Hardware Setup…”.

(15) On the hardware setup window, set “Debug Configurations” and press OK.
Click “Project >> Download and Debug”.

3. API Functions

R_LPT_Open ()

The function initializes the LPT FIT module. This function must be called before calling any other API functions.

Format

```
lpt_err_t R_LPT_Open (
    uint32_t const lpt_period
)
```

Parameters

uint32_t const lpt_period
LPT cycle (unit: μs)

Return Values

LPT_SUCCESS: */* Processing completed successfully. */*
LPT_ERR_LOCK_FUNC: */* Operating. LPT has been used. */*
LPT_ERR_INVALID_ARG: */* Argument has an invalid value. */*

Properties

Prototyped in r_lpt_rx_if.h.

Description

The initialization is performed to start LPT operation and then the LPT cycle specified with the argument is set.
Operations included in the initialization are as follows:

- Enables exiting software standby mode using the LPT.
- Sets the LPT clock source and the division ratio.
- Sets the LPT cycle.
- Provides the LPT clock.
- Resets the LPT.

Example

```
void main(void)
{
      lpt_err_t err;
      uint32_t lpt_period;

      lpt_period = 100000;
      err = R_LPT_Open(period);
      if (LPT_SUCCESS != err)
      {
            while(1) { };
      }
}
```

Special Notes

Call this function while the LPT clock source oscillation is stabilized.

When the sub-clock oscillator is selected as the LPT clock source, the LPT cycle must be specified from 92 to 64000488. But if using device that supported "no clock division", the LPT cycle must be specified from 46 to 64000488.
When the IWDT-dedicated on-chip oscillator is selected as the LPT clock source, the LPT cycle must be specified from 200 to 139811199. But if using device that supported "no clock division", the LPT cycle must be specified from 100 to 139811199.
When the LOCO is selected as the LPT clock source, the LPT cycle must be specified from 2 to 2097167.
When the IWDT-dedicated on-chip oscillator is selected as the LPT clock source, set the OFS0.IWDTSLCSTP bit to 0 (counting stop is disabled) in IWDT auto-start mode, and set the IWDTCSTPR.SLCSTP bit to 0 (counting stop is disabled) in other modes.
When the LOCO clock is selected as the LPT clock source , set the LFOCR.LOFXIN bit to 1 (counting stop is disabled) in other modes.
MCU executes the program after MCU waits for the stability time for Main Clock Oscillator Wait Control Register (SMOSCWTCR) when MCU is resumed from software standby mode.

R_LPT_InitChan ()

This function enable compare match and sets the value of LPT compare match.

Format

```
lpt_err_t R_LPT_InitChan (
    lpt_ch_t chan,
    uint32_t const cmt_period
)
```

Parameters

lpt_err_t chan
Channel to initialize.
uint32_t const cmt_period
LPT Compare match timer (unit: microsecond)

Return Values

LPT_SUCCESS:	*/* Processing completed successfully. */*
LPT_ERR_INVALID_ARG:	*/* Argument has an invalid value. */*
LPT_ERR_CONDITION_NOT_MET	*/* Condition not met. */*
LPT_ERR_INVALID_CH	*/* Selected channel is invalid */*

Properties

Prototyped in r_lpt_rx_if.h.

Description

This API function performed to enable compare match and sets the value of LPT compare match with the argument.

Example

```
void main(void)
{
        lpt_err_t err;
        uint32_t lpt_period;
       lpt_ch_t chan;
       uint32_t const cmt_period;

        lpt_period = 100000;
        err = R_LPT_Open(lpt_period);
        if (LPT_SUCCESS != err)
        {
                while(1) { };
        }

                            :
                            :

        chan = LPT_CH0;
        cmt_period = 100000;

        err = R_LPT_InitChan (chan, cmt_period);
        if (LPT_SUCCESS != err)
        {
                while(1) { };
        }

                            :
                            :

}
```

Special Notes:

When executing this function, set the value corresponding to the clock source for the compare match cycle of the argument as well as the LPT cycle. See R_LPT_Open () for details.

Set the value of Table 3.1 or higher for the compare match cycle of the argument according to the LPCNTPSSEL register set by R_LPT_Open. If the value of cmt_period is less than Table 3.1, it returns LPT_ERR_INVALID_ARG. For the LPCNTPSSEL register, refer to the hardware manual of each device.

Set the value of cmt_period to a value less than or equal the LPT period. If the value of cmt_period is greater than the LPT period, LPT_ERR_INVALID_ARG is returned.

This function must be called while LPT count stops. If this function is called during counting, LPT_ERR_CONDITION_NOT_MET is returned.

If the PWM output is enabled and the selected channel is 1, set the compare match cycle to the equal value as the LPT cycle. If the compare match cycle is set to a value different from the LPT cycle when the PWM output is enabled and the selected channel is 1, LPT_ERR_CONDITION_NOT_MET is returned.

Channel 1 can only be selected on some devices. If channel 1 is selected on unsupported device, LPT_ERR_INVALID_CH is returned.

Table 3.1 Compare match cycle setting minimum value

Clock source	LPCNTPSSEL					
	0	1	2	3	4	5
Sub-Clock	46	92	184	367	733	1465
IWDT	100	200	400	800	1600	3200
LOCO	2	3	6	12	24	48

R_LPT_SetCMT ()

This function set the value of LPT compare match.

Format

```
lpt_err_t R_LPT_SetCMT (
    lpt_ch_t chan,
    uint32_t const cmt_period
)
```

Parameters

lpt_err_t chan
Channel to initialize.
uint32_t const cmt_period
LPT Compare match timer (unit: microsecond)

Return Values

LPT_SUCCESS	*/* Argument has an invalid value */*
LPT_ERR_INVALID_ARG	*/* Argument has an invalid value */*
LPT_ERR_CONDITION_NOT_MET	*/* Condition not met */*
LPT_ERR_INVALID_CH	*/* Selected channel is invalid */*

Properties

Prototyped in r_lpt_rx_if.h.

Description

This API function performed to set the value of LPT compare match with the argument.

Example

```
void main(void)
{
        lpt_err_t err;
        uint32_t lpt_period;
       lpt_ch_t chan;
       uint32_t const cmt_period;

        lpt_period = 100000;
        err = R_LPT_Open(lpt_period);
        if (LPT_SUCCESS != err)
        {
                while(1) { };
        }

                          :
                          :

        chan = LPT_CH0;
        cmt_period = 100000;

        err = R_LPT_InitChan (chan, cmt_period);
        if (LPT_SUCCESS != err)
        {
                while(1) { };
        }

                          :
                          :

       cmt_period = 50000;
        err = R_LPT_SetCMT (chan, cmt_period);
        if (LPT_SUCCESS != err)
        {
                while(1) { };
        }

}
```

Special Notes:

Set cmt_period to the value corresponding to the clock source. Also, set a value equal to or greater than "[Minimum LPT period of selected clock source] x 2 LPCNTPSSEL register" for the compare match cycle of the argument. See the R_LPT_Open () for more information.

Set the value of cmt_period to a value less than or equal the LPT period. If the value of cmt_period is greater than the LPT period, LPT_ERR_INVALID_ARG is returned.

This function must be called while LPT count stops. If this function is called during counting, LPT_ERR_CONDITION_NOT_MET is returned. But if selected cannel 0 and PWM output is enable, this function can be called.

If the PWM output is enabled and the selected channel is 1, set the compare match cycle to the equal value as the LPT cycle. If the compare match cycle 1 is set to a value different from the LPT cycle when the PWM output is enabled, LPT_ERR_CONDITION_NOT_MET is returned.

Channel 1 can only be selected on some devices. If channel 1 is selected on an unsupported device, LPT_ERR_INVALID_CH is returned.

R_LPT_InitPWM ()

This function set the PWM configuration.

Format

```
lpt_err_t R_ R_LPT_InitPWM (
    lpt_ch_t chan,
    lpt_pwm_cfg_t * const p_config
)
```

Parameters

lpt_err_t chan
LPT channel.
*lpt_pwm_cfg_t * const p_config*
PWM configuration.

This module supports the following PWM settings: The PWM output characteristics are determined by setting the following structural elements in "p_config".

```
typedef enum e_lpt_pwm_polarity
{
    output_polarity_low=0,    /* Output polarity low */
    output_polarity_high      /* Output polarity high */
} lpt_pwm_polarity_t;

typedef enum e_lpt_pwm_level
{
    output_level_low=0,        /* Output level low */
    output_level_high          /* Output level high */
} lpt_pwm_level_t;
```

Return Values

LPT_SUCCESS / Processing completed successfully */*
LPT_ERR_CONDITION_NOT_MET / Condition not met */*
LPT_ERR_INVALID_CH / Selected channel is invalid */*
LPT_ERR_NULL_PTR / p_config is NULL */*

Properties

Prototyped in r_lpt_rx_if.h.

Description

This API function sets the PWM configuration.

Example

```
void main(void)
{
        lpt_err_t err;
        uint32_t lpt_period;
       lpt_ch_t chan;
       uint32_t const cmt_period;
       lpt_pwm_cfg_t const pwm_config;

        lpt_period = 100000;
        err = R_LPT_Open(lpt_period);
        if (LPT_SUCCESS != err)
        {
                while(1) { };
        }

                          :
                          :

        chan = LPT_CH0;
        cmt_period = 100000;

        err = R_LPT_InitChan (chan, cmt_period);
        if (LPT_SUCCESS != err)
        {
                while(1) { };
        }

                          :
                          :

       pwm_config.output_polarity = output_polarity_high;
       pwm_config.output_level = output_level_low;
        err = R_ R_LPT_InitPWM (chan, &pwm_config);
        if (LPT_SUCCESS != err)
        {
                while(1) { };
        }

}
```

Special Notes:

This function must be executed while LPT count stops. If this function is executed during counting, LPT_ERR_CONDITION_NOT_MET is returned.

PWM function can only executed on channel 0. If this function is executed with channel 1 selected, LPT_ERR_INVALID_CH is returned.

Set a valid value in the argument p_config. If this function is executed with an invalid value in p_config, LPT_ERR_NULL_PTR is returned.

R_LPT_Control ()

This function performs processing to start, stop, or reset LPT count and start, stop PWM output.

Format

```
lpt_err_t R_LPT_Control (
   lpt_cmd_t const cmd       /* Command */
)
```

Parameters

lpt_cmd_t const cmd
Command to be executed (see 2.9, Parameters).

Return Values

LPT_SUCCESS: / Processing completed successfully. */*
LPT_ ERR_INVALID_ARG: / Argument has an invalid value. */*
LPT_ERR_CONDITION_NOT_MET / Condition not met. */*

Properties

Prototyped in r_lpt_rx_if.h.

Description

This API function controls start/stop of LPT count and start and stop of PWM output.

The counter reset command (LPT_CMD_COUNT_RESET) must be executed while LPT count stops. If LPT_CMD_COUNT_RESET is executed during counting, LPT_ERR_CONDITION_NOT_MET is returned.

The PWM start/stop command (LPT_CMD_PWM_START/LPT_CMD_PWM_STOP) must be executed while LPT count stops. If LPT_CMD_PWM_START or LPT_CMD_PWM_STOP is executed during counting, LPT_ERR_CONDITION_NOT_MET is returned.

The LPT cycle and compare match 1 cycle must be equal when executing the PWM start command. If the values of LPT cycle and compare match 1 cycle are not equal, LPT_ERR_CONDITION_NOT_MET is returned.

LPT_CMD_PWM_START and LPT_CMD_PWM_STOP command can only execute on some device. If LPT_CMD_PWM_START or LPT_CMD_PWM_STOP is executed on unsupported device, LPT_ERR_INVALID_ARG is returned.

Example

```
void main(void)
{
        lpt_err_t err;
        uint32_t lpt_period;
       lpt_ch_t chan;
       uint32_t const cmt_period;

        lpt_period = 100000;
        err = R_LPT_Open(lpt_period);
        if (LPT_SUCCESS != err)
        {
                while(1) { };
        }
                            :
                            :

        chan = LPT_CH0;
        cmt_period = 100000;

        err = R_LPT_InitChan (chan, cmt_period);
        if (LPT_SUCCESS != err)
        {
                while(1) { };
        }

                            :
                            :

        err = R_LPT_Control(LPT_CMD_START);
        if (LPT_SUCCESS != err)
        {
                while(1) { };
        }

                            :
                            :

}
```

Special Notes

Call this function after the LPT have been configured in the R_LPT_Open and R_LPT_InitChan function.

R_LPT_FinalChan ()

This function finalizes compare match.

Format

```
lpt_err_t R_LPT_FinalChan (
    lpt_ch_t chan,
)
```

Parameters

lpt_err_t chan
Channel to finalize.

Return Values

LPT_SUCCESS	*/* Processing completed successfully */*
LPT_ERR_CONDITION_NOT_MET	*/* Condition not met */*
LPT_ERR_INVALID_CH	*/* Selected channel is invalid */*

Properties

Prototyped in r_lpt_rx_if.h.

Description

This API function performed to finalize compare match selected channel.

Example

```
void main(void)
{
        lpt_err_t err;
        uint32_t lpt_period;
       lpt_ch_t chan;
       uint32_t const cmt_period;

        lpt_period = 100000;
        err = R_LPT_Open(lpt_period);
        if (LPT_SUCCESS != err)
        {
                while(1) { };
        }

                          :
                          :

        chan = LPT_CH0;
        cmt_period = 100000;

        err = R_LPT_InitChan (chan, cmt_period);
        if (LPT_SUCCESS != err)
        {
                while(1) { };
        }

                          :
                          :

        err = R_LPT_FinalChan (chan);
        if (LPT_SUCCESS != err)
        {
                while(1) { };
        }

}
```

Special Notes:

This function must be called while LPT count stops. If this function is called during counting, LPT_ERR_CONDITION_NOT_MET is returned.

Channel 1 can only be selected on some devices. If channel 1 is selected on unsupported device, LPT_ERR_INVALID_CH is returned.

R_LPT_Close ()

This function performs processing to stop the LPT.

Format

```
lpt_err_t R_LPT_Close (
    void
)
```

Parameters

None

Return Values

LPT_SUCCESS: */* Processing completed successfully. */*

Properties

Prototyped in r_lpt_rx_if.h.

Description

The following operations are performed to stop the LPT.

- Stops the LPT.
- Resets the LPT if the LPT clock is provided.
- Stops the LPT clock.
- Resets the value of LPT compare match 0.
- Resets the value of LPT compare match 1.
- Resets the LPT cycle.
- Resets low-power timer control register 1.
- Disables exiting software standby mode using the LPT.

Example

```
void main(void)
{
      lpt_err_t err;
      uint32_t lpt_period;

      period = 100000;
      err = R_LPT_Open(period);
      if (LPT_SUCCESS != err)
      {
            while(1) { };
      }

      err = R_LPT_Close();
      if (LPT_SUCCESS != err)
      {
            while(1) { };
      }
}
```

Special Notes

Configure the LPT settings in the R_LPT_Open function first, wait one or more cycles of the LPT clock source, and then call this function.

R_LPT_GetVersion ()

This function returns the module version.

Format

```
uint32_t R_LPT_GetVersion (
  void
)
```

Parameters

None

Return Values

Version number

Properties

Prototyped in r_lpt_rx_if.h.

Description

Returns the module version number. The version number is encoded where the top 2 bytes are the major version number and the bottom 2 bytes are the minor version number.

Example

```
void main(void)
{
      uint32_t version;

      version = R_LPT_GetVersion();
      while(1) { };
}
```

Special Notes

None

4. Pin Setting

To use the PWM function with the low power timer FIT module, input/output signals of the peripheral function have to be allocated to pins with the multi-function pin controller (MPC). This pin allocation is referred to as “pin setting” in this document.

When performing the pin setting in the e^2 studio, the pin setting feature of the FIT configurator or the SmartConfigurator can be used. When using the pin setting feature, a source file is generated according to the option selected in the Pin Setting window in the FIT configurator or the Smart Configurator. Pins are configured by calling the function defined in the source file. Refer to Table 4.1 for details.

Table 4.1 Function Output by the FIT Configurator

MCU Used	Function to be Output	Remarks
RX140	R_LPT_PinSet_LPT()	When using the PWM function

5. Sample Code

This section describes the sample code for periodically exiting software standby mode using the LPT.

Operations are performed in the following order.

(1) Calls the R_LPC_LowPowerModeConfigure function to enable software standby mode.
(2) Cancels the module stop state for the ELC.
(3) Disables maskable interrupts.
(4) Disables DTC activation by the ELSR19I interrupt to set the communication target of the ELSR19I interrupt to the CPU.
(5) Disables the ELSR19I interrupt.
(6) Specifies the interrupt priority level of the ELSR19I interrupt higher than the processor interrupt priority level (IPL).
(7) Enables the ELSR19I interrupt.
(8) Enables the ELC function.
(9) Links the LPT compare match event of ELC to ICU (LPT dedicated interrupt).
(10) Calls the R_LPT_Open function to enable for the LPT FIT module to use the LPT.
(11) Calls the R_LPT_InitChan function to enable compare matching.
(12) Calls the R_LPT_Control function to start LPT count.
(13) Calls the R_LPC_LowPowerModeActivate function to enter software standby mode.
(14) With LPT compare match 0 occurrence, enters normal operation mode and then calls ELSR19I interrupt exception handling.
(15) Returns step 12 after executing user processing after exiting software standby mode.

- This sample code supports on RX113.
- This program uses the LPC FIT module. About the LPC FIT module, please refer to Renesas Electronics Website.

```
#include "platform.h"
#include "r_lpc_rx100_if.h"
#include "r_lpt_rx_if.h"

void main(void);

void main(void)
{
   lpt_err_t lpt_err;
   uint32_t lpt_period;
   lpc_err_t lpc_err;
   lpt_ch_t chan;
   uint32_t const cmt_period;

   /* ---- Disable to stop counting in sleep mode. ---- */
   IWDT.IWDTCSTPR.BIT.SLCSTP = 0;

   /* ---- Software standby mode setting ---- */
   lpc_err = R_LPC_LowPowerModeConfigure(LPC_LP_SW_STANDBY);
   if (LPC_SUCCESS != lpc_err)
   {
       while(1) { };
   }

   /* Cancel the module stop state for the ELC. */
   R_BSP_RegisterProtectDisable(BSP_REG_PROTECT_LPC_CGC_SWR);
   MSTP(ELC) = 0;
   R_BSP_RegisterProtectEnable(BSP_REG_PROTECT_LPC_CGC_SWR);

   /* ---- Disable maskable interrupts. ---- */
   R_BSP_InterruptsDisable();

   /* ---- ELSR19I interrupt settings ---- */
   ICU.DTCER[80].BIT.DTCE = 0;
   ICU.IER[0x0A].BIT.IEN0 = 0;
   ICU.IPR[80].BIT.IPR = 15;
   ICU.IER[0x0A].BIT.IEN0 = 1;
   while ( 1 != ICU.IER[0x0A].BIT.IEN0 )
   {
       /* Check if the written value is correctly reflected. */
   }

   /* ---- ELC settings ---- */
   ELC.ELCR.BIT.ELCON = 1;        /* ELC function is enabled. */
   ELC.ELSR[19].BIT.ELS = 0x5D;   /* Links the LPT compare match event of ELC (5Dh)
                                     to ICU (LPT dedicated interrupt). */

   /* ---- LPT initialization ---- */
   Lpt_period = 100000;               /* LPT cycle = 100000[µs] */
   lpt_err = R_LPT_Open(lpt_period);
   if (LPT_SUCCESS != lpt_err)
   {
       while(1) { };
   }

   /* ---- LPT compare match 0 initialization  ---- */
   cmt_period = 100000;               /* compare match cycle = 100000[µs] */
   chan       = LPT_CH0;              /* compare match channel 0 */
   err = R_LPT_InitChan (chan, cmt_period);
   if (LPT_SUCCESS != err)
   {
       while(1) { };
   }

   /* ---- Start LPT count. ---- */
   lpt_err = R_LPT_Control(LPT_CMD_START);
   if (LPT_SUCCESS != lpt_err)
   {
       while(1) { };
   }

   while(1)
   {
       /* ---- Enter software standby mode. ---- */
       lpc_err = R_LPC_LowPowerModeActivate(NULL);
       if (LPC_SUCCESS != lpc_err)
       {
```

When the IWDT-dedicated on-chip oscillator is selected as the LPT clock source, the setting not to stop the clock during software standby is required.

Transition from the ELC operation enable state to normal operation mode is made using the ELSR19I interrupt.

The ELSR19I interrupt is triggered by completion of the transition from software standby mode to the ELC operation enable state.

```
            while(1) { };
        }
        /* ---- Write user processing after exiting software standby mode. ---- */
    }
}
```

Figure 5.1 Example of Processing for Periodically Exiting Software Standby Mode Using the LPT

6. Demo Projects

Demo projects are complete stand-alone programs. They include function main() that utilizes the module and its dependent modules (e.g. r_bsp). This FIT module has the following demo projects:

The following demo project works only on Ver 3.00 or later.

6.1 lpt_demo_rskrx231

This section describes the sample code for periodically exiting software standby mode using the LPT.

The demo will be executed through these steps:

(1) Configured LPC module for Software standby mode

(2) Configure IWDT clock not to stop counting in Software standby mode

(3) Cancel module stop-state for ELC

(4) Disable DTC activation so the interrupt request will send directly to CPU

(5) Configure ELC module Low-power timer count down match event

(6) Start ELC and LPT operation

(7) Main loop: Enter software standby mode in 3 seconds and then return back to normal operation mode

Notes:

- This program uses the LPC FIT module. About the LPC FIT module, please refer to Renesas Electronics Website
- When debugging with e^2 studio, to know if the program is in Software Standby mode or Normal mode, user can observe the program running status on the bottom left corner of e^2 studio window:
  - **Standby**: the program is in Software Standby mode.
  - **Running**: the program is in Normal mode.
- To enable the LPT clock source, the LTP clock source (BSP_CFG_LPT_CLOCK_SOURCE) of BSP is changed from the initial value to the IWDT dedicated on-chip oscillator .

How to confirm:

- Program will print out a message on the console notify about entering software standby mode.
- Program will enter software standby mode in 3 seconds, CPU and all peripheral functions without setting of module stop state will stop in this mode.
- After 3 seconds, program will return back to normal mode, print out a notify message on the console and blink LED 0 in 1 second.

6.2 lpt_demo_rskrx113

The lpt_demo_rskrx113 program is identical to lpt_demo_rskrx231.

Note: To enable the LPT clock source, the LPT clock source (BSP_CFG_LPT_CLOCK_SOURCE) of BSP is changed from the initial value to the IWDT dedicated on-chip oscillator.

6.3 lpt_demo_tbkrx140

This section describes the sample code for PWM output using the LPT.

The demo will be executed through these steps:

(1) Configured LPT module

(2) Set the PWM waveform output

(3) Set the pins for PWM output

(4) Start PWM operation

(5) Main loop: Start LPT operation and the PWM waveform is output at a duty ratio of 50% (1 second interval)

Notes:

- To enable the LPT clock source, the LTP clock source (BSP_CFG_LPT_CLOCK_SOURCE) of BSP is changed from the initial value to the IWDT dedicated on-chip oscillator.

How to confirm:

- Attach the terminal of a measuring device such as an oscilloscope to P26 (CN16).
- When you run the program, you can see the PWM output with a duty ratio of 50% (1 second interval).

6.4 Adding a Demo to a Workspace

Demo projects are found in the FIT Demos subdirectory of the distribution file for this application note. To add a demo project to a workspace, select *File >> Import >> General >> Existing Projects into Workspace* , then click “Next”. From the Import Projects dialog, choose the “Select archive file” radio button. “Browse” to the FIT Demos subdirectory, select the desired demo zip file, then click “Finish”.

6.5 Downloading Demo Projects

Demo projects are not included in the RX Driver Package. When using the demo project, the FIT module needs to be downloaded. To download the FIT module, right click on the required application note and select “Sample Code (download)” from the context menu in the *Smart Browser >> Application Notes* tab.

7. Appendices

7.1 Operation Confirmation Environment

This section describes operation confirmation environment for the LPT FIT module.

Table 7.1 Operation Confirmation Environment (Rev. 3.00)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio 2021-07 (21.1.0)
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.03.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
	GCC for Renesas RX 8.3.0.202004 Compiler option: The following option is added to the default settings of theintegrated development environment. -std=gnu99
	IAR C/C++ Compiler for Renesas RX version 4.20.1 Compiler option: The default settings of the integrated development environment.
Endian	Big endian/little endian
Revision of the module	Rev.3.00
Board used	Renesas Starter Kit for RX113（product No: R0K505113SxxxBE） Renesas Starter Kit for RX130（product No: R0K5051SxxxBE） Target board for RX140（product No: RTK5RX140xxxxxxxxx） Renesas Starter Kit for RX231（product No: R0K505231SxxxBE）

Table 7.2 Operation Confirmation Environment (Rev. 2.01)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio 2020-10 (20.10.0)
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.02.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian	Big endian/little endian
Revision of the module	Rev.2.01
Board used	Renesas Starter Kit for RX113 (product No: R0K505113SxxxBE) Renesas Starter Kit for RX231 (product No: R0K505231SxxxBE)

Table 7.3 Operation Confirmation Environment (Rev. 2.00)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 7.8.0 IAR Embedded Workbench for Renesas 4.14.01
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.02.00 Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
	GCC for Renesas RX 8.03.00.201904 Compiler option: The following option is added to the default settings of the integrated development environment. -std=gnu99
	IAR C/C++ Compiler for Renesas RX version 4.14.01 Compiler option: The default settings of the integrated development environment.
Endian	Big endian/little endian
Revision of the module	Rev.2.00
Board used	Renesas Solution Starter Kit for RX23W (product No.: RTK5523Wxxxxxxxxx) Renesas Starter Kit for RX231 (product No.: R0K505231xxxxxx)

Table 7.4 Operation Confirmation Environment (Rev. 1.23)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 7.3.0
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00
	Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian	Big endian/little endian
Revision of the module	Rev.1.23

Table 7.5 Operation Confirmation Environment (Rev. 1.22)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 7.3.0
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V3.01.00
	Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian	Big endian/little endian
Revision of the module	Rev.1.22
Board used	Renesas Starter Kit+ for RX113 (product No: R0K505113SxxxBE)

Table 7.6 Operation Confirmation Environment (Rev. 1.21)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 6.0.0
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V2.07.00
	Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian	Big endian/little endian
Revision of the module	Rev.1.21
Board used	Renesas Starter Kit+ for RX130-512KB (product No: RTK5051308SxxxxxBE)

Table 7.7 Operation Confirmation Environment (Rev. 1.20)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 6.0.0.001
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V2.07.00
	Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian	Big endian/little endian
Revision of the module	Rev.1.20
Board used	Renesas Starter Kit+ for RX130-512KB (product No: RTK5051308SxxxxxBE)

Table 7.8 Operation Confirmation Environment (Rev. 1.11)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 5.0.1.005
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V2.05.00
	Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian	Big endian/little endian
Revision of the module	Rev.1.11
Board used	Renesas Starter Kit+ for RX113 (product No: R0K505113SxxxBE) Renesas Starter Kit+ for RX231 (product No: R0K505231SxxxBE) Renesas Starter Kit+ for RX130 (product No: RTK5005130SxxxxxBE)

Table 7.9 Operation Confirmation Environment (Rev. 1.10)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 5.0.0.043
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V2.04.01
	Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian	Big endian/little endian
Revision of the module	Rev.1.10
Board used	Renesas Starter Kit+ for RX113 (product No: R0K505113SxxxBE) Renesas Starter Kit+ for RX231 (product No: R0K505231SxxxBE) Renesas Starter Kit+ for RX130 (product No: RTK5005130SxxxxxBE)

Table 7.10 Operation Confirmation Environment (Rev. 1.00)

Item	Contents
Integrated development environment	Renesas Electronics e^2 studio Version 4.2.0.012
C compiler	Renesas Electronics C/C++ Compiler Package for RX Family V2.04.01
	Compiler option: The following option is added to the default settings of the integrated development environment. -lang = c99
Endian	Big endian/little endian
Revision of the module	Rev.1.00
Board used	Renesas Starter Kit+ for RX113 (product No: R0K505113SxxxBE)

7.2 Troubleshooting

(1) Q: I have added the FIT module to the project and built it. Then I got the error: Could not open source file “platform.h”.

A: The FIT module may not be added to the project properly. Check if the method for adding FIT modules is correct with the following documents:

- Using CS+:

  Application note “Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826)”

- Using e^2 studio:

  Application note “Adding Firmware Integration Technology Modules to Projects (R01AN1723)”

When using a FIT module, the board support package FIT module (BSP module) must also be added to the project. Refer to the application note “Board Support Package Module Using Firmware Integration Technology (R01AN1685)”.

(2) Q: I have added the FIT module to the project and built it. Then I got the error: This MCU is not supported by the current r_lpt_rx module.

A: The FIT module you added may not support the target device chosen in your project. Check the supported devices of added FIT modules.

(3) Q: I have added the FIT module to the project and built it. Then I got an error for when the configuration setting is wrong.

A: The setting in the file “r_lpt_rx_config.h” may be wrong. Check the file “r_lpt_rx_config.h”. If there is a wrong setting, set the correct value for that. Refer to 2.7, Configuration Overview for details.

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Mar. 1, 2016	-	First edition issued
1.10	July. 1, 2016	all	Added supported devices for RX130, RX231 and RX230.
		All	Improved the Accuracy of the LPT periodic time when using the IWDT-dedicated on-chip oscillator to LPT clock source.
		10	Changed the periodic range for the argument of the API when LPT using the IWDT oscillator by improved influence to the Accuracy of the LPT periodic time.
		10	Added the note for the LPT periodic time on special note of section 3.1.
1.11	Oct. 1, 2016	All	Added the LPT counter reset command (LPT_CMD_COUNT_RESET) for the R_LPT_Control function.
		8, 13	Added LPT_ERR_CONDITION_NOT_MET to the Return Values.
		Program	Added the LPT counter reset command (LPT_CMD_COUNT_RESET) for the R_LPT_Control function.
		Program	Added LPT_ERR_CONDITION_NOT_MET as the return value.
1.20	Oct. 1, 2017	all	Added supported devices for RX130-512KB.
		1	Added the following document to Related Documents. “Renesas e2 studio Smart Configurator User Guide (R20AN0451)”
		3, 7	Move "Required memory size" in Chapter 1.2 to chapter 2.7.
		8	Revised 2.11 Adding the FIT Module to Your Project.
		17, 18	Added 5.1 Operation Confirmation Environment.
		19	Added 5.2 Troubleshooting.
1.21	Oct. 31, 2017	17	Added 5. Demo Projects
		18	6.1 Operation Confirmation Environment: Added table for Rev.1.21
		20	6.2 Troubleshooting: Added 2 more questions
1.22	Nov 16, 2018	–	Added document number in XML
		8	Updated “2.11 Adding the FIT Module to Your Project” Added “2.12 “for”, “while” and “do while” statements”
		18	Added table for Rev.1.22
1.23	Apr 01, 2019	–	Changes associated with functions: Added support setting function of configuration option Using GUI on Smart Configurator. [Description] Added a setting file to support configuration option setting function by GUI.
		3	Moved 1.3 API Overview.
		4	Changed 1.4 Processing Example.
		5	Changed 1.5 State Transition.
		6	Changed 2 API Information. Added 2.4 Interrupt Vector.
		7	Changed 2.8 Code Size.
		8	Changed 2.9 Parameters. Deleted Callback Function.
		9	Changed 2.12 “for”, “while” and “do while” statements.
		15	Changed R_LPT_GetVersion.

Rev.	Date	Description	
		Page	Summary
1.23	Apr 01, 2019	19	6.1 Operation Confirmation Environment: Added table for Rev.1.23.
2.00	Jun 10, 2020	-	Added support for RX23W Modified comment of API function to Doxygen style. Update the following compilers - GCC for Renesas RX - IAR C/C++ Compiler for Renesas RX.
		1	Added Target Compilers.
		1	Related Documents: Deleted the following documents Firmware Integration Technology User's Manual (R01AN1833) RX Family Adding Firmware Integration Technology Modules to Projects (R01AN1723) RX Family Adding Firmware Integration Technology Modules to CS+ Projects (R01AN1826) Renesas e^2 studio Smart Configurator User Guide (R20AN0451)
		7	Changed 2.8 Code Size.
		8	Changed 2.11 Adding the FIT Module to Your Project.
		9	Deleted Target devices describing "WAIT_LOOP".
		10..15	Deleted the Reentrant for each API in 3. API Functions.
		19	Changed 5.1 lpt_demo_rskrx231. Changed 5.2 lpt_demo_rskrx113.
		21	6.1 Operation Confirmation Environment: Added table for Rev.2.00.
2.01	Nov.30.20	-	Updated the sample code project due to the upgrade of the development environment.
3.00	Jul.31.21	-	Added support for RX140. Added function R_LPT_InitChan, R_LPT_SetCMT, R_LPT_FinalChan, R_LPT_InitPWM. Added the PWM start/stop command (LPT_CMD_PWM_START/LPT_CMD_PWM_STOP) for the R_LPT_Control function.

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

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

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

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 or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving 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, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. 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 shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: “Standard” and “High Quality”. The intended 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; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or 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 damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user’s manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user’s manuals, application notes, “General Notes for Handling and Using Semiconductor Devices” in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, 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 and impractical, you are responsible for evaluating 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. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall 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 shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, 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.

(Note1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/.