R01AN1854EG0100
Rev.1.00
Jun 17, 2014

RX Family

Event Link Controller Module Using Firmware Integration Technology

Introduction

The event link controller (ELC) connects (links) the events generated by various peripheral modules to different modules. Event linking allows direct cooperation between the modules without CPU intervention.

This FIT component provides support for the ELC controller provided on the target device.

Target Device

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

- **RX111 Group**

Related Documents

- Firmware Integration Technology User's Manual (R01AN1833EU0100)
- Board Support Package FIT Module (R01AN1685EU0220)
- Adding FIT Modules to Projects (R01AN1723EU0100)

Contents

1. Overview

1.1 Using this feature of the MCU

The event link controller (ELC) connects (links) the events generated by various peripheral modules to different modules. Event linking allows direct cooperation between the modules without CPU intervention. This software provides a driver to operate the ELC module.
To provide the required level of functionality it has been necessary for this software module to interact with the ELC peripheral, additionally this module operates a low-power mode that disables the ELC peripheral when not in use. To provide this power control feature the module uses the relevant low power module that controls the ELC.

1.2 How we do this

A simple interface has been provided allowing another software component to create a link between two available modules. The calling software simply has to specify which type module is to be controlled, which signal shall be used to trigger the target module and populate the appropriate configuration structure for the specified type of link.

1.3 Example of Middleware in Action

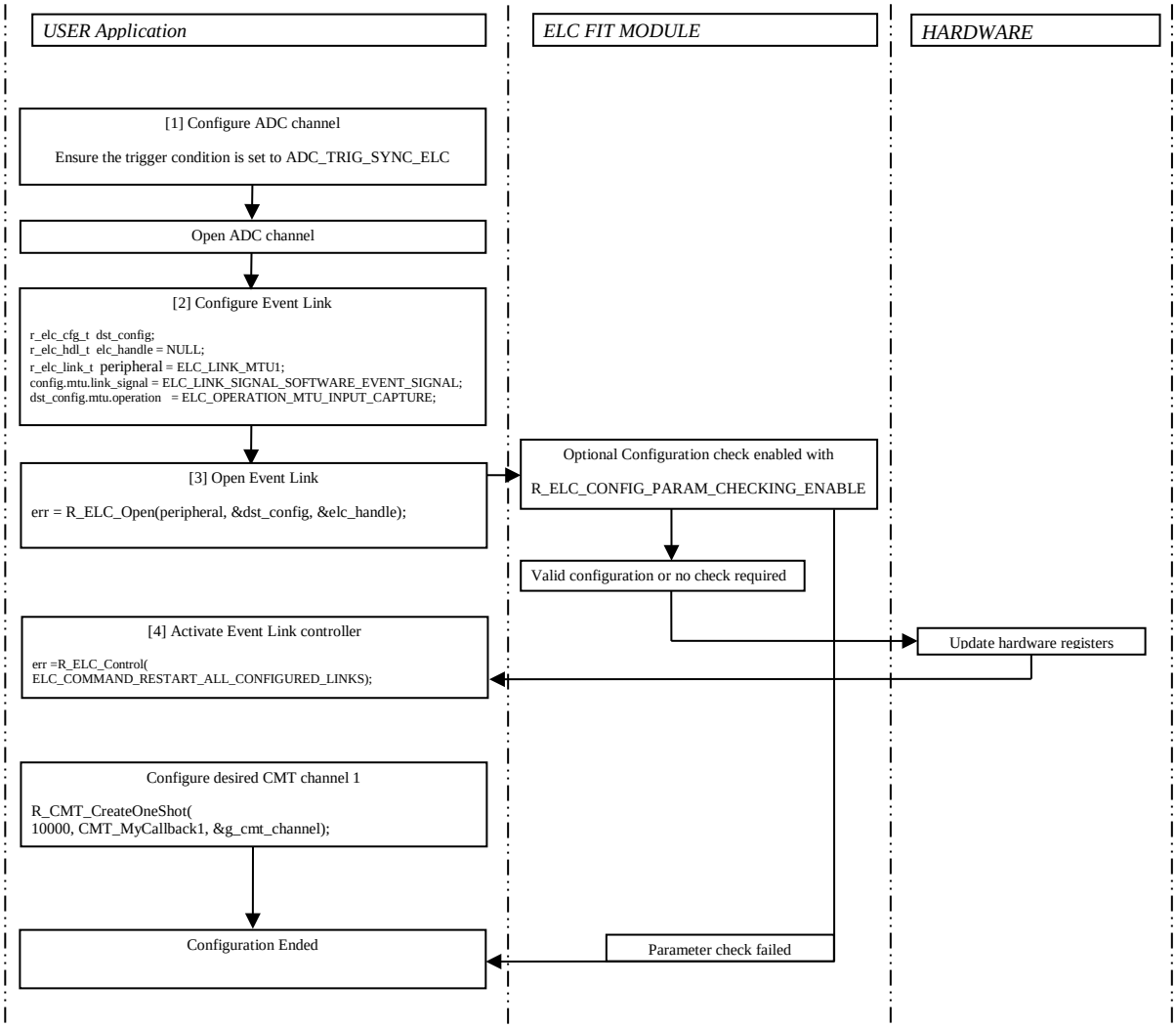


Figure 1 : Open and configure event link

What is happening?

This example shows how to use the ELC module to link the CMT channel 1 timer event to initiate the ADC module.

It will be necessary to configure the ADC and CMT modules prior to configuring the event.

Instructions for operation

[1] Configure the desired ADC operation

[2] Configure the proposed ELC configuration

[3] Open the relevant ELC configuration channel using *R_ELC_Open()* function.

[4] If channel opens successfully (indicated by *R_ELC_Open* command returning the code R_ELC_SUCCESS)

Configure any other desired Links

Activate the Event link controller using the command

R_ELC_Control(ELC_COMMAND_RESTART_ALL_CONFIGURED_LINKS);

[5] Configure the desired CMT operation

[6] Start the CMT timer

[7] The timer will trigger the ADC to start

2. API Information

This Middleware API follows the Renesas API naming standards.

2.1 Hardware Requirements

This middleware requires your MCU support the following features:

- RX111 BSP Version 2.2 or greater

2.2 Hardware Resource Requirements

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

2.2.1 ELC Peripheral

This driver makes use of the ELC peripheral, control of the driver software can be achieved by using the settings available in the configuration file r_elc_config.h which is located in the r_config folder.

2.3 Software Requirements

This middleware is dependent upon the following packages:

- r_bsp Version 2.2 or higher

2.4 Supported Toolchains

This middleware is tested and working with the following toolchain(s):

- Renesas RX Toolchain v1.02

2.5 Header Files

All API calls are accessed by including a single file *r_elc.h* which is supplied with this middleware's project code.

The elc API id configured via the single file *r_elc_config.h* which is supplied with this middleware's project code.

By default the parameter checking is enabled in the configuration file, this is useful when creating new links between modules but can be safely disabled once links have been developed to save space and increase performance.

2.6 Integer Types

If your toolchain supports C99 then *stdint.h* should be described as shown below. If not, then there should be *typedefs.h* file that is included with your project as defined by the Renesas Coding Standards document.

This project uses ANSI C99 "Exact width integer types" in order to make the code clearer and more portable. These types are defined in *stdint.h*.

2.7 Configuration Overview

2.7.1 Common Configuration

All FIT components shall use the appropriate config file to control any optional component of the module.

All controls available for this module use a binary ENABLED/DISABLED state.

The ELC config file shall be called "`r_elc_config.h`" and contain the following controls:

Table describing configuration options in this middleware	
`R_ELC_CONFIG_PARAM_CHECKING_ENABLE`	This configuration option allows the FIT component to check the specified channel configuration prior to committing the requested configuration to the actual ELC registers. With this mode ENABLED The ELC component will check the proposed configuration. If the configuration specified creates an error state then the function under use will be set to the appropriate `r_elc_err_t` value. Note if more than 1 error state is present only the first error is reported. With this mode DISABLED No checking of the proposed configuration is made and the desired configuration is applied to the actual ELC registers. This module cannot guarantee operation if the proposed configuration is invalid.

Table 1 : Info about the configuration

2.8 API Data Structures

This section details the data structures that are used with the middleware's API functions.

2.8.1 Common Structures

The following structures are common links and all Event links make use of these structures

r_elc_link_signal_id is used to represent which of the possible signals to link as an input to the desired peripheral. A list of the available commands is as follows:

```
/* Supported link signal sources for this module */
typedef enum r_elc_link_signal_id
{
   ELC_LINK_SIGNAL_MTU1_COMPARE_MATCH_1A_SIGNAL,
   ELC_LINK_SIGNAL_MTU1_COMPARE_MATCH_1B_SIGNAL,
   ELC_LINK_SIGNAL_MTU1_OVERFLOW_SIGNAL,
   ELC_LINK_SIGNAL_MTU1_UNDERFLOW_SIGNAL,
   ELC_LINK_SIGNAL_MTU2_COMPARE_MATCH_2A_SIGNAL,
   ELC_LINK_SIGNAL_MTU2_COMPARE_MATCH_2B_SIGNAL,
   ELC_LINK_SIGNAL_MTU2_OVERFLOW_SIGNAL,
   ELC_LINK_SIGNAL_MTU2_UNDERFLOW_SIGNAL,
   ELC_LINK_SIGNAL_MTU3_COMPARE_MATCH_3A_SIGNAL,
   ELC_LINK_SIGNAL_MTU3_COMPARE_MATCH_3B_SIGNAL,
   ELC_LINK_SIGNAL_MTU3_COMPARE_MATCH_3C_SIGNAL,
   ELC_LINK_SIGNAL_MTU3_COMPARE_MATCH_3D_SIGNAL,
   ELC_LINK_SIGNAL_MTU3_OVERFLOW_SIGNAL,
   ELC_LINK_SIGNAL_MTU4_COMPARE_MATCH_4A_SIGNAL,
   ELC_LINK_SIGNAL_MTU4_COMPARE_MATCH_4B_SIGNAL,
   ELC_LINK_SIGNAL_MTU4_COMPARE_MATCH_4C_SIGNAL,
   ELC_LINK_SIGNAL_MTU4_COMPARE_MATCH_4D_SIGNAL,
   ELC_LINK_SIGNAL_MTU4_OVERFLOW_SIGNAL,
   ELC_LINK_SIGNAL_MTU4_UNDERFLOW_SIGNAL,
   ELC_LINK_SIGNAL_CMT1_COMPARE_MATCH_1_SIGNAL,
   ELC_LINK_SIGNAL_SCI5_ERROR_RECEIVE_ERROR_OR_ERROR_SIGNAL_DETECTION_SIGNAL,
   ELC_LINK_SIGNAL_SCI5_RECEIVE_DATA_FULL_SIGNAL,
   ELC_LINK_SIGNAL_SCI5_TRANSMIT_DATA_EMPTY_SIGNAL,
   ELC_LINK_SIGNAL_SCI5_TRANSMIT_END_SIGNAL,
   ELC_LINK_SIGNAL_RIIC0_COMMUNICATION_ERROR_OR_EVENT_GENERATION_SIGNAL,
   ELC_LINK_SIGNAL_RIIC0_RECEIVE_DATA_FULL_SIGNAL,
   ELC_LINK_SIGNAL_RIIC0_TRANSMIT_DATA_EMPTY_SIGNAL,
   ELC_LINK_SIGNAL_RIIC0_TRANSMIT_END_SIGNAL,
   ELC_LINK_SIGNAL_AD_CONVERSION_END_SIGNAL_OF_12BIT_AD_CONVERTER,
   ELC_LINK_SIGNAL_LVD1_VOLTAGE_DETECTION_SIGNAL,
   ELC_LINK_SIGNAL_DTC_TRANSFER_END_SIGNAL,
   ELC_LINK_SIGNAL_INPUT_EDGE_DETECTION_SIGNAL_OF_INPUT_PORT_GROUP_1,
   ELC_LINK_SIGNAL_INPUT_EDGE_DETECTION_SIGNAL_OF_SINGLE_INPUT_PORT_0,
   ELC_LINK_SIGNAL_INPUT_EDGE_DETECTION_SIGNAL_OF_SINGLE_INPUT_PORT_1,
   ELC_LINK_SIGNAL_SOFTWARE_EVENT_SIGNAL,
   ELC_LINK_SIGNAL_DOC_DATA_CONDITION_MET_SIGNAL,
} r_elc_link_signal_id_t;
```

r_elc_link_t is used to represent which of the possible peripherals to configure, list of the available peripherals are as follows:

```
/* Supported peripherals */
typedef enum r_elc_link
{
    ELC_LINK_MTU1              = 0x01,
    ELC_LINK_MTU2              = 0x02,
    ELC_LINK_MTU3              = 0x03,
    ELC_LINK_MTU4              = 0x04,
    ELC_LINK_CMT1              = 0x07,
    ELC_LINK_ADC               = 0x0f,
    ELC_LINK_DAC               = 0x10,
    ELC_LINK_INT1              = 0x12,
    ELC_LINK_OUTPUT_BITGROUP_1 = 0x14,
    ELC_LINK_INPUT_BITGROUP_1  = 0x16,
    ELC_LINK_BIT_A             = 0x18,
    ELC_LINK_BIT_B             = 0x19
} r_elc_link_t;
```

The R_ELC_Command function is used to specify which command is to be sent to the driver. A list of the available commands is as follows:

```
/* Supported Control Commands for this module */
typedef enum r_elc_command_type
{
    ELC_COMMAND_HALT_ALL_CONFIGURED_LINKS,
    ELC_COMMAND_START_LINK,
    ELC_COMMAND_HALT_LINK,
    ELC_COMMAND_GENERATE_SOFTWARE_INTERRUPT
    ELC_COMMAND_START_ALL_CONFIGURED_LINKS,

} r_elc_command_t;
```

The following structures represent the configuration objects associated with each of the 7 types of Event links available in the r_elc_configure_type_t structure.

2.8.2 ELC_CFG module configuration structures

This structure is used to configure the elc module. Specific sub-structures (mtu,cmt.cvt etc.) are used to configure the associated modules, as the sub-structures overlap only one should be used.

Description of config structure:

```
/* configure signal for required events */
typedef union
{
       r_elc_config_mtu_t       mtu; /* ELC_LINK_MTU1, ELC_LINK_MTU2,
                                        ELC_LINK_MTU3, ELC_LINK_MTU4 events */
       r_elc_config_cmt_t       cmt; /* ELC_LINK_CMT1 events */
       r_elc_config_converter_t cvt; /* ELC_LINK_ADC & ELC_LINK_DAC events */
       r_elc_config_interrupt_t isr; /* ELC_LINK_INT1 events */
       r_elc_config_pgr_t       pgr; /* Port group events */
       r_elc_config_lkb_t       lkb; /* ELC_LINK_BIT_A, ELC_LINK_BIT_B events */
} r_elc_cfg_t;
```

Details information on the of the sub-structures follows:

2.8.3 ELC_CONFIGURATION_MTU Event link structures

Configuration Link r_elc_configure_type_t element ELC_CONFIGURATION_MTU is used to configure MTU events.

Description of config structure:

```
/* configuration for mtu category of elc event links */
typedef struct r_elc_mtu
{
    /* restricted list of signals available */
    r_elc_link_signal_id_t link_signal;

    /* restricted list of available operations for this configuration */
    r_elc_mtu_elos_t       operation;
} r_elc_config_mtu_t;
```

The `r_elc_link_signal_id_t` type can be set from all 36 available `r_elc_link_signal_id_t` types to act as the trigger for the link.

The `r_elc_mtu_elos_t` type can be configured to operate the linked peripheral in one of four modes:

ELC_OPERATION_MTU_COUNT_START – The counter is started

ELC_OPERATION_MTU_COUNT_RESTART – The counter is restarted

ELC_OPERATION_MTU_INPUT_CAPTURE – The relevant TCNT is captured into the appropriate TGRA register

Note - if ELC_FUNCTION_MTU_CH1 is configured MTU1.TCNT is captured into MTU1.TGRA

Note - if ELC_FUNCTION_MTU_CH2 is configured MTU2.TCNT is captured into MTU2.TGRA

Note - if ELC_FUNCTION_MTU_CH3 is configured MTU3.TCNT is captured into MTU3.TGRA

Note - if ELC_FUNCTION_MTU_CH4 is configured MTU4.TCNT is captured into MTU4.TGRA

ELC_OPERATION_MTU_DISABLE – The event is disabled

2.8.4 ELC_CONFIGURATION_CMT Event link structures

Configuration Link r_elc_configure_type_t element ELC_CONFIGURATION_CMT is used to configure CMT channel 1 events.

Description of config structure:

```
/* configuration for cmt category of elc event links */
typedef struct r_elc_cmt
{
    /* restricted list of signals available */
    r_elc_link_signal_id_t link_signal;

    /* restricted list of available operations for this configuration */
    r_elc_cmt_elos_t       operation;
} r_elc_config_cmt_t;
```

The `r_elc_link_signal_id_t` type can be set from all 36 available `r_elc_link_signal_id_t` types to act as the trigger for the link.

The `r_elc_cmt_elos_t` type can be configured to operate the linked peripheral in one of four modes:

ELC_OPERATION_CMT_COUNT_START – The counter is started

ELC_OPERATION_CMT_COUNT_RESTART – The counter is restarted

ELC_OPERATION_CMT_EVENT_CAPTURE – The relevant TCNT is captured into the appropriate TGRA register

ELC_OPERATION_CMT_DISABLE – The event is disabled

2.8.5 ELC_CONFIGURATION_CONVERTER Event link structures

Configuration Link r_elc_configure_type_t element ELC_CONFIGURATION_CONVERTER is used to configure the converter peripheral events.

Description of config structure:

```
/* configuration for converter category of elc event links */
typedef struct r_elc_converter
{
    /* restricted list of signals available */
    r_elc_link_signal_id_t link_signal;
} r_elc_config_converter_t;
```

The `r_elc_link_signal_id_t` type can be set from all 36 available `r_elc_link_signal_id_t` types to act as the trigger for the link.

The converter peripherals can only be started by a link event so there is no need to specify any control options for this link.

2.8.6 ELC_CONFIGURATION_INTERRUPT Event link structures

Configuration Link r_elc_configure_type_t element ELC_CONFIGURATION_INTERRUPT is used to configure the interrupt peripheral events.

Description of config structure:

```
/* configuration for interrupt category of elc event links */
typedef struct r_elc_interrupt
{
    /* restricted list of signals available */
    r_elc_link_signal_id_t link_signal;
} r_elc_config_interrupt_t;
```

The `r_elc_link_signal_id_t` type can be set from all 36 available `r_elc_link_signal_id_t` types to act as the trigger for the link.

The interrupt peripherals can only be started by a link event so there is no need to specify any control options for this link.

2.8.7 ELC_CONFIGURATION_PGR Event link structures

Configuration Link r_elc_configure_type_t element ELC_CONFIGURATION_PGR is used to configure the link events for the associated I/O port bits that can be linked as outputs. The PGR setting specifies each port bit in the same 8-bit I/O port as the member of a group. One to eight port bits can be specified as the members of the same group as required.

Description of config structure:

```
/* configuration for pgr category of elc event links */
typedef struct r_elc_pgr
{
    /* restricted list of available */
    r_elc_link_signal_id_t link_signal;

    /* restricted list of available operations for this configuration */
    r_elc_link_signal_id_t    link_signal;
    r_elc_cfg_input_bitgroup_t  input_cgf;
    r_elc_cfg_output_bitgroup_t output_cgf;
} r_elc_config_pgr_t;
```

The `r_elc_link_signal_id_t` type can be set from all 36 available `r_elc_link_signal_id_t` types to act as the trigger for the link.

Output configuration structure.

```
/* configuration for cfg output bitgroup category of pgr elc event links */
typedef struct r_elc_cfg_output_bitgroup
{
   /* port configuration bit mask */
   uint8_t                 group_mask;

   /* port operation upon event input */
   r_elc_bitgroup_output_t action;
} r_elc_cfg_output_bitgroup _t;
```

The `group_mask` for each I/O port specifies each port bit in the same 8-bit I/O port as the member of a group. One to eight port bits can be specified as the members of the same group as required.

0: The port bit is not specified as a member of the same group.

1: The port bit is specified as a member of the same group.

The `r_elc_cfg_output_bitgroup_t` type for each group can be configured on input to operate the following action:

ELC_BITGROUP_ACTION_OUTPUT_0 – The port bit shall output 0 when the event is input..

ELC_BITGROUP_ACTION_OUTPUT_1 – The port bit shall output 1 when the event is input..

ELC_BITGROUP_ACTION_OUTPUT_TOGGLE – The port bit shall toggle (invert) current value when the event is input.

ELC_BITGROUP_ACTION_OUTPUT_BUFFER – The port bit shall output the buffer value when the event is input.

ELC_BITGROUP_ACTION_OUTPUT_ROTATE – The port bit value is rotated out in the group (from MSB to LSB) when the event is input.

Input configuration structure.

```
/* configuration for cfg input bitgroup category of pgr elc event links */
typedef struct r_elc_cfg_input_bitgroup
{
   /* port configuration bit mask */
   uint8_t                  group_mask;

   /* port operation upon event output */
   r_elc_bitgroup_output_t action;
} r_elc_cfg_output_bitgroup _t;
```

Input configuration structure.

```
/* configuration for cfg input bitgroup category of pgr elc event links */
typedef struct r_elc_cfg_input_bitgroup
{
   /* port configuration bit mask */
   uint8_t                  group_mask;

   /* port operation upon event input */
   r_elc_edge_detect_t     edge_detection;
   r_elc_enabled_t         port_buf_overwrite;
} r_elc_cfg_input_bitgroup _t;
```

The `group_mask` for each I/O port specifies each port bit in the same 8-bit I/O port as the member of a group. One to eight port bits can be specified as the members of the same group as required.

0: The port bit is not specified as a member of the same group.

1: The port bit is specified as a member of the same group.

The `r_elc_edge_detect_t` type for each group can be configured on input to generate an event upon the following action:

ELC_EDGE_DETECT_RISING_EDGE – Event upon detection of the rising edge of the external input signal.

ELC_EDGE_DETECT_FALLING_EDGE – Event upon detection of the falling edge of the external input signal.

ELC_EDGE_DETECT_BOTH_EDGES – Event upon detection of the both edges of the external input signal.

The `r_elc_buffer_enabled _t` type for each group the buffer overwrite can be controlled:

ELC_BUFF_ENABLED – Overwriting enabled.

ELC_BUFF_DISABLED – Overwriting disabled.

2.8.8 ELC_CONFIGURATION_PGC Event link structures

Configuration Link r_elc_configure_type_t element ELC_CONFIGURATION_PGC is used to configure the link events for the associated I/O port bits.

Control between input and output is managed by the group operation setting (r_elc_pgc_port_group_operation_select_t).

For the output port group, PGC specifies the form of outputting the signal externally via the port when the event signal set as input (r_elc_pgc_event_output_select_t).

For the input port group, PGC enables/disables overwriting of PDBF and specifies the conditions of event generation (r_elc_buffer_overwrite_select_t).
Description of config structure:

```
/* configuration for pgc category of elc event links */
typedef struct r_elc_pgc
{
    /* restricted list of available */
    r_elc_link_signal_id_t link_signal;

    /* restricted list of available operations for this configuration */
    r_elc_pgc_port_group_operation_select_t group_operation;
    r_elc_pgc_event_output_select_t         output_select;
    r_elc_pgc_buffer_overwrite_select_t     buffer_overwrite;
} r_elc_config_pgc_t;
```

The `r_elc_link_signal_id_t` type can be set from all 36 available `r_elc_link_signal_id_t` types to act as the trigger for the link.

The `r_elc_pgc_port_group_operation_select_t` type for each group can be configured to operate the linked peripheral in one of five modes:

when the event is input

ELC_OPERATION_PGC_GROUP_OUTPUT_0 – 0 is output.

ELC_OPERATION_PGC_GROUP_OUTPUT_1 – 1 is output.

when the event is output

ELC_OPERATION_PGC_GROUP_OUTPUT_TOGGLE – the toggled (inverted) value is output.

ELC_OPERATION_PGC_GROUP_OUTPUT_BUFFER – the buffer value is output.

ELC_OPERATION_PGC_GROUP_OUTPUT_ROTATE – the bit value is rotated out in the group (from MSB to LSB).

The `r_elc_pgc_event_output_select_t` type for each group can be configured to operate the linked peripheral in one of three modes:

The Event is generated upon the following condition

ELC_EVENT_OUTPUT_PGC_DETECT_RISING_EDGE – detection of the rising edge of the external input signal.

ELC_EVENT_OUTPUT_PGC_DETECT_FALLING_EDGE – detection of the falling edge of the external input signal.

ELC_EVENT_OUTPUT_PGC_DETECT_BOTH_EDGES – detection of both rising and falling edges of the external input signal.

The `r_elc_pgc_buffer_overwrite_select_t` type for each group can be configured to operate the linked peripheral in one of three modes:

The Event is generated upon the following condition

ELC_BUFFER_PGC_OVERWRITE_DISABLED – overwriting PDBF register is disabled.

ELC_BUFFER_PGC_OVERWRITE_ENABLED – overwriting PDBF register is enabled.

2.8.9 ELC_CONFIGURATION_PEL Event link structures

Configuration Link r_elc_configure_type_t element ELC_CONFIGURATION_PEL is used to configure the event link port peripheral events.

Description of config structure:

```
/* configuration for event link port category of elc event links */
typedef struct r_elc_pel
{
    /* restricted list of available */
    r_elc_link_signal_id_t link_signal;

    /* restricted list of available operations for this configuration */
    r_elc_pel_bit_spec_t   bit_number_specification;
    r_elc_pel_link_spec_t  event_link_specification;
} r_elc_config_pel_t;
```

The `r_elc_link_signal_id_t` type can be set from all 36 available `r_elc_link_signal_id_t` types to act as the trigger for the link.

The `r_elc_pel_bit_spec` type can be configured to operate the linked peripheral in one of 8 bits:

ELC_PEL_BIT_0 – Select bit 0 of the port

ELC_PEL_BIT_1 – Select bit 1 of the port

ELC_PEL_BIT_2 – Select bit 2 of the port

ELC_PEL_BIT_3 – Select bit 3 of the port

ELC_PEL_BIT_4 – Select bit 4 of the port

ELC_PEL_BIT_5 – Select bit 5 of the port

ELC_PEL_BIT_6 – Select bit 6 of the port

ELC_PEL_BIT_7 – Select bit 7 of the port

The `r_elc_pel_link_spec` type can be configured to operate the linked peripheral in one of 3 modes:

ELC_PEL_LINK_OUTPUT_0_OR_DETECT_RISING_EDGE – For output port set 0, for input port detect rising edge.

ELC_PEL_LINK_OUTPUT_1_OR_DETECT_FALLING_EDGE – For output port set 1, for input port detect falling edge.

ELC_PEL_LINK_OUTPUT_TOGGLE_OR_DETECT_BOTH_EDGES – For output port is toggled (inverted), for input port detect rising and falling edges.

2.9 Return Values

elc_error_code
This section details the error codes that are used with the event link controller API functions.
With the exception of the `R_ELC_GetVersion`() interface function all other interface functions have a return type of r_elc_err.

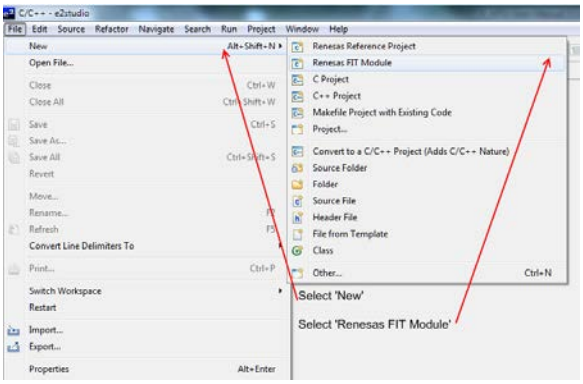
When completing its requested action the interface function returns one of the error codes shown below. If the function fails for multiple reasons then only the first issue shall be given, address this issue and re-run the function to determine the next issue.

```
/* Supported Error Controls for this Module */
typedef enum r_elc_err
{
   R_ELC_SUCCESS=0,                // Operation completed without error
   R_ELC_ERR_HW_LOCK_UNAVAILABLE,  // peripheral in use locking not available
   R_ELC_ERR_INVALID_ARG,          // argument is not valid for parameter
   R_ELC_ERR_INVALID_TYPE,         // Incorrect type specified
   R_ELC_ERR_INVALID_FUNCTION,     // Incorrect function specified for type
   R_ELC_ERR_LINK_EVENT_IN_USE,    // target event is in use
   R_ELC_ERR_INVALID_OPERATION,    // incorrect operation of specified link
   R_ELC_ERR_UNCONFIGURED_LINK,    // action cannot be performed link not open
   R_ELC_ERR_INVALID_LINK_SIGNAL,  // link signal cannot be specified
   R_ELC_ERR_INVALID_COMMAND       // command specified not recognized
} r_elc_err_t;
```

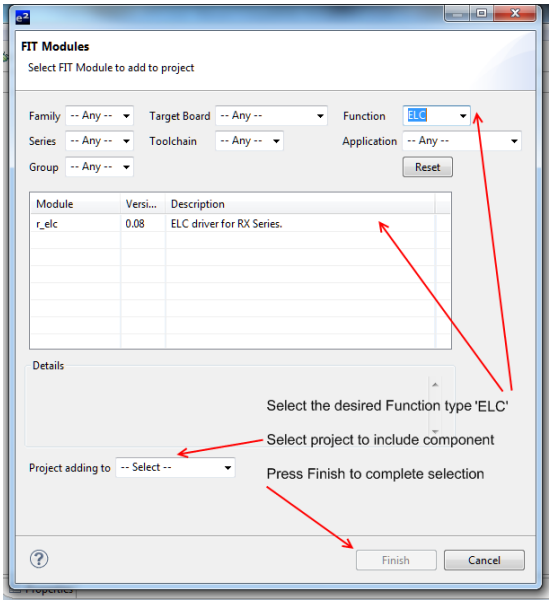
2.10 Adding Middleware to Your Project

Follow the steps below to add the middleware's code to your project when using e2studio.

1. Add new R_ELC FIT module to component to your workspace.



2. Choose 'r_elc' component and select project in which component is to be added.



3. Complete selection by pressing Finish.
4. Please read any messages that follow your decision, as this module has dependencies upon the version of r_bsp module.

Follow the steps below to add the middleware's code to your project when not using e2studio.

5. Copy the 'r_elc' directory (packaged with this application note) to your project directory.
6. Add a project include path for the "r_config" directory.
7. Add a project include path for the "r_elc" directory.
8. Add a project include path for the "r_elc\src" directory.
9. Open "r_config\r_elc_config.h" file and configure the driver for your project.

Adding r_elc support to your source code

1. Add the interface file to your source code i.e.

```
#include "r_elc_if.h"
```

The module should now be installed into your project.

3. API Functions

3.1 R_ELC_GetVersion

This function contains the FIT compliant version information MACRO for component.

Format

```
uint32_t R_ELC_GetVersion(void)
```

Parameters

none

Return Values

uint32_t The module internal version number

Properties

Prototyped in file “r_elc_if.h”

Description

Version information MACRO for component. 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. For example, Version 4.25 would be returned as 0x00040019.

Reentrant

This function does not modify data so it can be accessed at any time by any calling function.

Example

The following shows and example of the function in use.

```
#define NEEDS_VERSION_1_9 (0x00010009)
int main()
{
    uint32_t        version = R_ELC_GetVersion();
    if (version > NEEDS_VERSION_1_9)
    {
        /* Continue with product use */
    }
    else
    {
        printf(“Error minimum version of ELC FIT interface not available\n”);
    }
}
```

3.2 R_ELC_Open

This function contains the FIT compliant function required to open access to specified elc peripheral.

Format

```
bool R_ELC_Open(r_elc_link_t peripheral,
                r_elc_cfg_t * const p_config,
                r_elc_hdl_t * const p_hdl)
```

Parameters

peripheral – target to configure

p_config – desired configuration

p_hdl – handle created by FIT module to be used to access the configuration with other functions

Return Values

ELC_SUCCESS:	Successful, function has been applied to selected peripheral
ELC_ERROR_XXX:	Error while attempting to apply command. When R_ELC_CONFIG_PARAM_CHECKING_ENABLE is ENABLED the more detailed error handling is in use. The error code table elc_error_code_t details specific error available in this module. When R_ELC_CONFIG_PARAM_CHECKING_ENABLE is DISABLED function shall not return the code ELC_SUCCESS.

Properties

Prototyped in file “r_elc_if.h”

Description

This function uses the supplied configuration to set up the desired event link.

The active links configured by the R_ELC_Open command shall not become active until the start command (ELC_COMMAND_START_ALL_CONFIGURED_LINKS)

Reentrant

No function modifies channel settings and is not protected against access while executing.

Example

The following shows and example of the R_ELC_Open function is in use.

```
int main()
{
    r_elc_cfg_t  dst_config;
    r_elc_hdl_t elc_handle     = NULL;
    r_elc_link_t src_signal    = ELC_LINK_MTU2;
    dst_config.mtu.link_signal = ELC_LINK_SIGNAL_SOFTWARE_EVENT_SIGNAL;
    dst_config.mtu.operation   = ELC_OPERATION_MTU_DISABLE;

    r_elc_err_t err = R_ELC_Open(src_signal, &dst_config, &elc_handle);
    if (R_ELC_SUCCESS == err)
    {
        /* Continue code */
    }
}
```

3.3 R_ELC_Close

This function contains the FIT compliant function required to terminate access to specified link.

Format

```
r_elc_err_t R_ELC_Close (r_elc_hdl_t * const p_hdl)
```

Parameters

p_hdl - handle created by FIT open function to configuration

Return Values

ELC_SUCCESS: Successful, function has been applied to selected configuration

Properties

Prototyped in file “r_elc_if.h”

Description

This function closes an open event link. The link must have been opened by the R_ELC_Open command, though it need not be active.

Reentrant

No function modifies channel settings and is not protected against access while executing.

Example

The following shows and example of the R_ELC_Close function is in use.

```
int main()
{
    r_elc_cfg_t  dst_config;
    r_elc_hdl_t elc_handle     = NULL;
    r_elc_link_t src_signal    = ELC_LINK_MTU2;
    dst_config.mtu.link_signal = ELC_LINK_SIGNAL_SOFTWARE_EVENT_SIGNAL;
    dst_config.mtu.operation   = ELC_OPERATION_MTU_DISABLE;

    r_elc_err_t err = R_ELC_Open(src_signal, &dst_config, &elc_handle);
    if (R_ELC_SUCCESS == err)
    {
        /* Close active link */
       R_ELC_Close(elc_handle);
    }
}
```

3.4 R_ELC_Read

This function contains the FIT compliant function required to read data from I/O port

Format

```
r_elc_err_t R_ELC_Read (r_elc_data_command_t * const cmd,
                        uint8_t * data)
```

Parameters

cmd – specify which port to read

data – allocated memory to store result from read function

Command list

ELC_ACCESS_DATA_PDBF1 – Port B

Return Values

ELC_SUCCESS:	Successful, data had been read
ELC_ERROR_XXX:	Error while attempting to apply command. When R_ELC_CONFIG_PARAM_CHECKING_ENABLE is ENABLED the more detailed error handling is in use. The error code table elc_error_code_t details specific error available in this module. When R_ELC_CONFIG_PARAM_CHECKING_ENABLE is DISABLED function shall not return the code ELC_SUCCESS.

Properties

Prototyped in file “r_elc_if.h”

Description

This function reads data from the data port connected to the elc . Any link must have been opened by the R_ELC_Open command, though it need not be active.

Reentrant

No function modifies I/O port and is not protected against access while executing.

Example

The following shows and example of the R_ELC_Open function is in use.

```
int main()
{
    r_elc_hdl_t elc_handle = NULL;
    r_elc_cfg_t  dst_config;
    r_elc_link_t src_signal = ELC_LINK_MTU2;
    dst_config.mtu.link_signal = ELC_LINK_SIGNAL_SOFTWARE_EVENT_SIGNAL;
    dst_config.mtu.operation   = ELC_OPERATION_MTU_DISABLE;
    uint8_t data = 0;

    r_elc_err_t err = R_ELC_Open(src_signal, &dst_config, &elc_handle);
    if (R_ELC_SUCCESS == err)
    {
        /* Close active link */
       R_ELC_Read(ELC_ACCESS_DATA_PDBF1, &data);
    }
```

```
}
```

3.5 R_ELC_Write

This function contains the FIT compliant function required to write data to I/O port

Format

```
r_elc_err_t R_ELC_Write (r_elc_data_command_t * const cmd,
                         uint8_t * data)
```

Parameters

cmd – specify which port to write

data – data to write to I/O port

Command list

ELC_ACCESS_DATA_PDBF1 – Port B

Return Values

ELC_SUCCESS:	Successful, data had been written
ELC_ERROR_XXX:	Error while attempting to apply command. When R_ELC_CONFIG_PARAM_CHECKING_ENABLE is ENABLED the more detailed error handling is in use. The error code table elc_error_code_t details specific error available in this module. When R_ELC_CONFIG_PARAM_CHECKING_ENABLE is DISABLED function shall not return the code ELC_SUCCESS.

Properties

Prototyped in file “r_elc_if.h”

Description

This function writes data to the data port connected to the elc . Any link must have been opened by the R_ELC_Open command, though it need not be active.

Reentrant

No function modifies I/O port and is not protected against access while executing.

Example

The following shows and example of the R_ELC_Open function is in use.

```
int main()
{
    r_elc_hdl_t elc_handle = NULL;
    r_elc_cfg_t  dst_config;
    r_elc_link_t src_signal = ELC_LINK_MTU2;
    dst_config.mtu.link_signal = ELC_LINK_SIGNAL_SOFTWARE_EVENT_SIGNAL;
    dst_config.mtu.operation   = ELC_OPERATION_MTU_DISABLE;
    uint8_t data = 0xFF;

    r_elc_err_t err = R_ELC_Open(src_signal, &dst_config, &elc_handle);
    if (R_ELC_SUCCESS == err)
    {
        /* Close active link */
       R_ELC_Write(ELC_ACCESS_DATA_PDBF1, &data);
    }
```

```
}
```

3.6 R_ELC_Control

This function contains the FIT compliant function required to send control commands to the event link driver.

Format

```
r_elc_err_t R_ELC_Control (r_elc_hdl_t     const p_hdl,
                           r_elc_command_t const cmd)
```

Parameters

p_hdl – handle created by FIT open function to configuration

cmd – specify which port to write

Command list

ELC_COMMAND_HALT_ALL_CONFIGURED_LINKS – Linkage of all the configured events is disabled.

ELC_COMMAND_START_LINK – Linkage of configured event identified by the p_hdl parameter is enabled.

ELC_COMMAND_HALT_LINK – Linkage of configured event identified by the p_hdl parameter is disabled.

ELC_COMMAND_GENERATE_SOFTWARE_INTERRUPT – Software event is generated.

ELC_COMMAND_START_ALL_CONFIGURED_LINKS – Linkage of all the configured events is enabled.

Return Values

ELC_SUCCESS:	Successful, command completed without reported error
ELC_ERROR_XXX:	Error while attempting to apply command. When R_ELC_CONFIG_PARAM_CHECKING_ENABLE is ENABLED the more detailed error handling is in use. The error code table elc_error_code_t details specific error available in this module. When R_ELC_CONFIG_PARAM_CHECKING_ENABLE is DISABLED function shall not return the code ELC_SUCCESS.

Properties

Prototyped in file “r_elc_if.h”

Description

This function is used to send a

Reentrant

No function modifies elc controller and is not protected against access while executing.

Example

The following shows and example of the `R_ELC_Control` function is in use.

```
int main()
{
    r_elc_hdl_t elc_handle = NULL;
    r_elc_cfg_t  dst_config;
    r_elc_link_t src_signal = ELC_LINK_MTU2;
```

```
    dst_config.mtu.link_signal = ELC_LINK_SIGNAL_SOFTWARE_EVENT_SIGNAL;
    dst_config.mtu.operation   = ELC_OPERATION_MTU_DISABLE;

    r_elc_err_t err = R_ELC_Open(src_signal, &dst_config, &elc_handle);
    if (R_ELC_SUCCESS == err)
    {
        /* start the active links */
       R_ELC_Control(ELC_COMMAND_START_ALL_CONFIGURED_LINKS);
    }
}
```

4. Provided Modules

The modules provided can be downloaded from the Renesas Electronics website.

5. Reference Documents

User's Manual: Hardware

RX111 User's Manual: Hardware Rev.1.00 (R01UH0365EJ)
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.

User's Manual: Development Tools

RX Family C/C++ Compiler Package V.1.01 User's Manual Rev.1.00 (R20UT0570EJ)
The latest version can be downloaded from the Renesas Electronics website.

Website and Support

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

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

Revision History

		Description	
Rev.	**Date**	**Page**	**Summary**
1.00	June 17, 2014	--	First edition issued

General Precautions in the Handling of MPU/MCU Products

The following usage notes are applicable to all MPU/MCU products from Renesas. For detailed usage notes on the products covered by this 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 an MPU or MCU 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.

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

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, 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 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022/9044

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

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Ku, Seoul, 135-920, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 4.0